

Decomposition of complex two-dimensional shapes into simple convex shapes

Z van Rensburg



orcid.org/0000-0001-5647-000X

Dissertation submitted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Prof PA van Vuuren

Graduation ceremony: October 2019

Student number: 24181773

Abstract:

Decomposing a complex shape into visually significant parts comes naturally for humans, and turns out to be very useful in areas such as shape analysis, shape matching, recognition, topology extraction, collision detection and other geometric processing methods [1]. After analysis it was found that the Minimum Near-Convex Decomposition (MNCD) method [2] is one of the most promising algorithms currently available that shows room for improvement.

The focus of this dissertation is to make an improvement on the time it takes to decompose a complex shape, while keeping the decomposition (number of parts) results the same. One improvement that was implemented was to neglect the Morse function, as this takes a long time to execute. Another improvement was to make use of Delaunay Triangulation (DT) instead of considering all of the vertices, as no overlapping will take place and the need for the non-overlapping matrix is no longer necessary. Experimental results show that an average time reduction of 58%, but an increase in the number of parts. Thus there is an improvement made on the duration of the algorithm, but there is room to improve on the total amount of parts obtained after decomposition.

Keywords:

shape decomposition, complex shapes, simple shapes, convex shapes, Delaunay Triangulation (DT), Minimum Near-Convex Decomposition (MNCD), Discrete Contour Evolution (DCE), shape simplification, time complexity, optimization, parts

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Short literature survey	3
1.3	Research question	4
1.4	Method	4
1.5	Validation	5
1.6	Dissertation overview	5
2	Literature study	7
2.1	Background research	7
2.1.1	Terminology	7
2.1.2	Shape descriptors	11
2.1.3	Shape decomposition	15
2.1.4	Different types of shape decomposition	16
2.1.5	<i>Discussion</i>	18
2.2	Literature review	18
2.2.1	Quality of the solution	19
2.2.2	Shape decomposition methods	26
2.2.3	Choosing the method to improve on	31
2.3	Conclusion	33
3	The MNCD method	35
3.1	Introduction	35
3.2	Background	35
3.2.1	<i>Mutex Pairs</i>	35
3.2.2	<i>Morse Theory</i>	36
3.2.3	Binary Integer Linear Programming (BILP)	38
3.3	The Minimum Near-Convex Decomposition method	39
3.3.1	Overview	42
3.3.2	Decomposing a shape	46
3.3.3	Visually naturalness condition ($\mathbf{w}^T \mathbf{x}$)	47
3.3.4	Convexity constraint ($\mathbf{Ax} \geq \mathbf{1}$)	48
3.3.5	Non overlapping constraint ($\mathbf{Bx} \leq \mathbf{1}$)	48
3.3.6	Selection of variables	48
3.3.7	Binary Integer Linear Programming (BILP)	49

3.4	Conclusion	49
4	Improved implementation	50
4.1	Introduction	50
4.2	Algorithm overview	50
4.3	Background	54
4.3.1	Harris Corner Detection	55
4.3.2	Discrete Contour Evolution (DCE)	57
4.3.3	Delaunay Triangulation (DT)	60
4.4	Improved MNCD-method	61
4.4.1	Reduce the cut set $C(S)$	62
4.4.2	Morse function removal	68
4.4.3	Preprocessing	68
4.4.4	Determining the convexity matrix (A -matrix)	75
4.4.5	Identifying simpler shapes	77
4.5	Conclusion	78
5	Experimental results	79
5.1	Formulating hypothesis	80
5.2	Experiments	81
5.2.1	Accuracy evaluation	81
5.2.2	Time evaluation	83
5.2.3	Number of parts evaluation	84
5.2.4	Invariance evaluation	85
5.2.5	Parameter evaluation	87
5.2.6	Comparison to other algorithms	97
5.3	Simple shape output results	99
5.3.1	Conclusion	101
6	Conclusion	102
6.1	Conclusion	102
6.2	Future improvements	102
A	Article	110
B	Human Perception Experiment	129
B.1	Human Perception Experiment	130
B.1.1	The problem	130
B.1.2	Investigative question	130
B.1.3	Hypothesis	130
B.1.4	Experiments	130
B.1.5	Capture of results	131
B.1.6	Conclusion	138
C	Questionnaires	139

List of Figures

1.1	Picture demonstrating the concept of shape decomposition.	2
2.1	Figure illustrating the difference between convex (a) and concave (b), with the curves at the top, and the polygons at the bottom. The green lines indicate interior angles less than 180° , where the red line indicates interior angle greater than 180°	8
2.2	Figure illustrating the difference between computational decomposition (a) and human perceptual decomposition (b). This image is used to demonstrate visually naturalness.	8
2.3	Figure illustrating a vertex (a) and a convex and concave angle (b). Concave angles are also referred to as notches or reflex.	9
2.4	Picture showing how the different concave parts fit into a concavity tree of the shape [3].	9
2.5	Figure illustrating the concept of mutex pairs, where $(p_1; p_2)$ locates completely outside of the contour and $(p_1; p_3)$ intercepts with the contour, and are thus mutex pairs. However $(p_1; p_4)$ are completely inside the contours and is thus not a mutex pair.	10
2.6	Figure illustrating the concept of a Morse function $f(p)$ where the green line is the Morse function of the half doughnut.	10
2.7	Figure illustrating the concept of near-convex. As can be seen in (a) the pentagon is strictly convex, in (b) there is a rather large angle (green) to indicate that it is concave. In (c) there is a smaller angle which if allowed can be classified as a near-convex shape	11
2.8	Picture to show the graph representation process: (a) original image, (b) contours of the original image, (c) convert contours into vectors, (d) obtain primitive quadrilaterals, (e) obtain number of nodes and (f) use vector length for node and angle for graph [4].	12
2.9	Figure showing the shape at the left, with a yellow arrow indicating the distance to the boundary. As this arrow moves along the shape boundary, the distance is mapped against the angle that this 'arrow' forms with the centroid of the shape (red dot). The results are shown in the graph next to the shape [5].	12
2.10	Picture showing (a) the convex hull of a shape and its concavities and (b) the concavity tree of the shape [3].	13
2.11	Picture showing the medial axis (red) of the shape (black).	14
2.12	Picture showing an example of shape decomposition [6].	14
2.13	Graph explaining the different types of Shape decomposition [2].	16
2.14	Picture to demonstrate the minima rule (red dots), the short-cut rule (purple lines) and the neck- and-limb rule (green circles and lines) [7].	17
2.15	Picture to demonstrate the use of geometrical methods to decompose the object. In this case morphological operations were used to determine disk of most importance [8].	18
2.16	The outlines of the most commonly used shapes found in different articles [2, 9, 10, 11, 12].	19

2.17	Picture illustrating the results of shape decomposition done by humans through the use of questionnaires.	19
2.18	Picture to demonstrate how to determine the minimal number of parts.(a) shows that there exist one concave vertex, while (b) shows that the shape is indeed concave. (c) demonstrates that with one concave vertex, at least two parts can be formed [13].	21
2.19	Picture to demonstrate translation invariance - (a) the original position with the blue dot as the origin, (b) shows a translation of 10px left, while (c) shows a translation of 7px right and 5px up.	23
2.20	Picture to demonstrate rotation invariance - (a) the original position with the pink dot as the origin, (b) shows a rotation of 90° , while (c) shows a translation 180° in an anti-clockwise direction.	23
2.21	Picture to demonstrate size invariance - (a) the original size with the blue line to indicate size, (b) shows a resize of 75%, while (c) shows a translation of 125% of the original size.	24
2.22	Picture to demonstrate noise invariance - (a) the original image, (b) shows slight distortion in the image (c) shows noise in the original image.	24
2.23	Some of the decomposition results of Wang <i>et al.</i> (PFSD) [14].	27
2.24	the shape decomposition steps of the MNCD algorithm [2].	28
2.25	Some of the decomposition steps followed by the CSD method [15].	29
2.26	Some of the decomposition results of Lien <i>et al.</i> (ACD) [16].	30
2.27	Steps summarizing the AD algorithm [1].	31
2.28	The decomposition results of the above mentioned algorithms [1].	32
3.1	Picture demonstrating the concepts of mutex pairs.	36
3.2	Picture demonstrating the Morse function of a half circle shown as the height function f in blue.	37
3.3	Picture demonstrating the Reeb graph of a half circle shown as the changes in the number of connected components in the function f^{-1} in blue.	38
3.4	Picture demonstrating that near-convex decomposition does not decompose a complex shape into strict convex parts. As can be seen in the decomposed picture on the right, the red circle indicates a concave vertex, thus the green part is not convex, but near-convex.	40
3.5	Picture demonstrating the definition of a decomposed shape.	41
3.6	A figure demonstrating that any two vertices pq or v_1v_3 with a line connecting them that lies completely within the contour, is considered a cut (shown in green). The line v_1v_2 is found outside the concave part of the contour and line v_2, v_3 intersect with the contour, which forms mutex pairs (shown in blue).	42
3.7	A figure demonstrating all of the possible cut in the cut set $C(S)$	43
3.8	A figure demonstrating the cut set $C(S)$ that has now been shrunk to only contain cuts with at least one concave vertex.	44
3.9	A figure demonstrating mutex areas identified by using multiple Morse functions.	44
3.10	A figure demonstrating the cuts that separate the mutex pairs and the A matrix that shows which cuts separates which mutex pairs.	45
3.11	A figure showing the cost of each cut and the cost vector obtained w	45
3.12	A figure showing the intersection of each cut and the intersection matrix B obtained.	46
3.13	A figure showing the final selected cuts and the selection vector $\mathbf{x}$	46
4.1	First step of the algorithm: Obtain corners using HCD.	51

4.2	Second step of the algorithm: Obtain vertices using DCE.	52
4.3	Third step of the algorithm: Obtain cut set using DT.	52
4.4	Fourth step of the algorithm: Determine ψ -mutex set.	53
4.5	Fifth step of the algorithm: Determine convexity matrix $\mathbf{A}$	53
4.6	Sixth step of the algorithm: Determine cost vector $\mathbf{w}$	53
4.7	Seventh step of the algorithm: Solve the BILP and obtain selected cuts vector $\mathbf{x}$	54
4.8	Eighth step of the algorithm: Obtain primitive shapes.	54
4.9	Harris corner detection: a simple image to show the criteria required to be classified as a corner. When assuming that the rectangle is the derivative of the original image, and that it is in grayscale. The colours are inserted for it to makes sense. The green area shown high intensity, the pink shows lower intensities and yellow region with very low, or zero intensity [18].	56
4.10	Process showing how the DCE process works.	58
4.11	DCE: an image to demonstrate how the relevance value is determined.	59
4.12	Image illustrating Delaunay triangulation where (a) is the input points drawn, (b) is the Delaunay triangulation of the input points and (c) showing an example of the circum-circle properties [19].	61
4.13	Algorithm to obtain optimal DCE explained step-by-step.	63
4.14	Plot of $J(I)$ vs. Number of parts removed. The pink circle and green circle indicates the DCE process of figure 4.13 between (d-e) and (g-i) respectively. The green circle also indicates the stopping criteria, which is 9 vertices.	64
4.15	These two graphs show the effect of filling $\mathbf{J}$ with zeros (a) and ones(b). As can be seen the minimum values for (a) is zero and the index is equal to the original length where the index in (b) is not zero and thus the index is not equal the original length.	65
4.16	Figure demonstrating the concept of (a) overlapping and (b) non-overlapping cuts.	66
4.17	(a)A figure that shows the DT of an arrow when a set of points (blue) is given. Yellow lines (b) are to indicate that these lines aren't part of the contour, green lines (c) show duplicate lines, red lines (d) show desired triangulation.	67
4.18	(a) shows all of the mutex pairs and how the Morse function is no longer used.	68
4.19	(a) shows the corner that has been detected and a zoomed section showing that two points approximately ten pixels away are also used to determine the angle of the corner(A) by using the cosine rule.	69
4.20	(a) shows the corner that has been detected and a zoomed section showing that two points approximately ten pixels away are also used to determine the curvature of the corner(A).	72
4.21	a) Part shown with one possible cut, in magenta. b) shows the one part that is cut and the green circle corner points that lie on the cut part, the blue square corner points that form part of the cut coordinates and the magenta diamond point that is outside the cut part.	75
4.22	Figure showing (a) final selected cut and (b) two simpler shapes identified	77
5.1	Figures that form part of the MPEG-7 shape dataset and that will be used in the experiments.	79
5.2	Figure showing several images used from the MPEG-7 dataset. Each picture on the left represents the ground truth. On the right the picture of the algorithm cuts is shown with thick red lines indicating cuts that agree, while the thin red lines shows cuts that do not agree.	82
5.3	Figure showing the results of the MNCD algorithm (middle) and the improve algorithm (right) decomposition results after different distortion were applied to an object.	86

5.4	Figure showing the results of the MNCD algorithm (middle) and the improved algorithm (right) decomposition after an object has been rotated.	87
5.5	Human shape decomposition result, also referred to as human ground truth, of a camel. . .	88
5.6	Decomposition and accuracy in % results of Ψ -parameter, with (a) $\Psi = 0.01R$, (b) $\Psi = 0.3R$, (c) $\Psi = 0.5R$, (d) $\Psi = 0.75R$ and (e) $\Psi = 1R$, where R is the radius of the enclosing circle that surrounds the object and parameters $\beta = 1$, and $\lambda = 1$	89
5.7	Decomposition results and accuracy in % of λ -parameter, with (a) $\lambda = 0$, (b) $\lambda = 0.25$, (c) $\lambda = 0.5$, (d) $\lambda = 0.75$ and (e) $\lambda = 1$ with $\beta = 1$, and $\Psi = 0.5R$	92
5.8	Decomposition results and accuracy in % of β -parameter with (a) $\beta = 0$, (b) $\beta = 0.25$, (c) $\beta = 0.5$, (d) $\beta = 1$ and (e) $\beta = 3$ with $\lambda = 0.5$, and $\Psi = 0.5R$	94
5.9	Decomposition results of multi-parameters	96
5.10	Decomposition results of parameters $\Psi = 0.15R$, $\lambda = 25 \times 10^{-7}$, and $\beta = 1$	97
5.11	Decomposition comparison with methods ACD, CSD, MNCD and our method. Parameters are set to $\Psi = 0.15R$, $\lambda = 0.25$, and $\beta = 1$	98
5.12	Approximate contour results with changes in σ . In (a) $\sigma = 10\%$, (b) $\sigma = 5\%$ and (c) $\sigma = 1\%$ of the perimeter of the shape.	99
5.13	Simple shape output after σ has been changed to: (a) $\sigma = 0.006$, (b) $\sigma = 0.01$, (c) $\sigma = 0.025$, (d) $\sigma = 0.03$, (e) $\sigma = 0.06$ and (f) $\sigma = 0.1$	100
B.1	Picture showing the questionnaire to evaluate human perception.	131
B.2	Picture showing the questionnaire to evaluate human perception.	132
B.3	Picture showing the questionnaire to evaluate human perception.	133
B.4	Picture showing the questionnaire to evaluate human perception.	133
B.5	Picture showing the questionnaire to evaluate human perception.	134
B.6	Picture showing the questionnaire to evaluate human perception.	134
B.7	Picture showing the questionnaire to evaluate human perception.	135
B.8	Picture showing the questionnaire to evaluate human perception.	135
B.9	Picture showing the questionnaire to evaluate human perception.	136
B.10	Picture showing the questionnaire to evaluate human perception.	137

List of Tables

2.1	Table showing a summary of all the criteria that will be used to choose a shape descriptor to make use of.	15
2.2	Table showing a summary of all the criteria that will be used to rank methods from highest to lowest in terms of improvement required.	25
2.3	Table showing the conversion of scores determined by the different criteria.	26
2.4	Table showing a summary of all the decomposition methods.	33
2.5	Table showing a summary of all the decomposition methods.	33
5.1	Table showing the average % deviation for the MNCD and the improved method from the ground truth experiment for the pictures of the MPEG-7 dataset.	83
5.2	Table showing the average % deviation(%) for the MNCD and the improved method from the ground truth experiment for the pictures of the MPEG-7 dataset.	84
5.3	Table showing the number of parts that the same shapes are decomposed into for the different methods. The last column showing the percentage improvement.	85
5.4	Table summarizing the results of changing the <i>psi</i> -parameter value and how the accuracy results are influenced by changes of the ψ value	89
5.5	Table showing the average reduction (%) in time for the pictures of the MPEG-7 dataset with $\lambda = 1$ and $\beta = 1$	90
5.6	Table showing the total number of parts (NOP) for the pictures of the MPEG-7 dataset with $\lambda = 1$ and $\beta = 1$	91
5.7	Table showing the average reduction (%) in time for the pictures of the MPEG-7 dataset with $\Psi = 0.5R$ and $\beta = 1$	93
5.8	Table showing the total number of parts (NOP) for the pictures of the MPEG-7 dataset while changing the λ parameter, with $\Psi = 0.1R$ and $\beta = 1$	93
5.9	Table showing the average reduction (%) in time for the pictures of the MPEG-7 dataset with $\Psi = 0.5R$ and $\lambda = 0.5$	95
5.10	Table showing the total number of parts (NOP) for the pictures of the MPEG-7 dataset while changing the β parameter, with $\Psi = 0.5R$ and $\lambda = 0.5$	95
5.11	Table showing the parameter values for each decomposed picture in figure 5.9.	97
5.12	Table showing the evaluation results of the different shape decomposition algorithms 5.9.	98
B.1	Table showing the average and standard deviation of the number of parts after decomposition of human perception	137

Chapter 1

Introduction

1.1 Introduction

Imagine you are a bird lover, and you have recently discovered a smart phone application that can identify different animals. While hiking you see a bird sitting on a branch with its back to the sun. You grab your phone to take a picture, and notice that the sun in the background makes only the silhouette appear of the bird. You take a blurry photo and open up the application. The application loads and you submit the photo. You wait for some time to pass. Eventually, after thinking that the phone froze, you get a result. To your utter disappointment, the application shows: *"Unable to identify. Please take another photo"*. By this time the bird is no longer there and the sun has almost set. Imagine you sit in the same scenario as mentioned above, but instead of waiting for the day to pass, you get an instant result where the beautiful bird has been identified as a rabbit.

As can be seen in this situation, a speedy response and an accurate result would have been very helpful. This is only one of many examples where a fast and accurate shape decomposition application will be appreciated. Some examples include classifying plants on the basis of the shape of the leaves of that plant, being able to identify motor vehicles by the shape and size thereof or classifying birds in terms of the different shapes put together to identify them to mention only a few applications. As can be seen in these examples mentioned, it is very important that fast and accurate classification be achieved in practice, as time is money and speed and accuracy it is critical in saving time and therefore saving money.

The three basic parts of image processing and pattern recognition include pre-processing, feature extraction and classification [20]. This dissertation however does not deal with the classification of objects, but that rather speeding up the process thereof - hence we will be looking at feature extraction. In the above mentioned scenario, shapes are of interest and there exists many different ways and means of extracting shapes from an image.

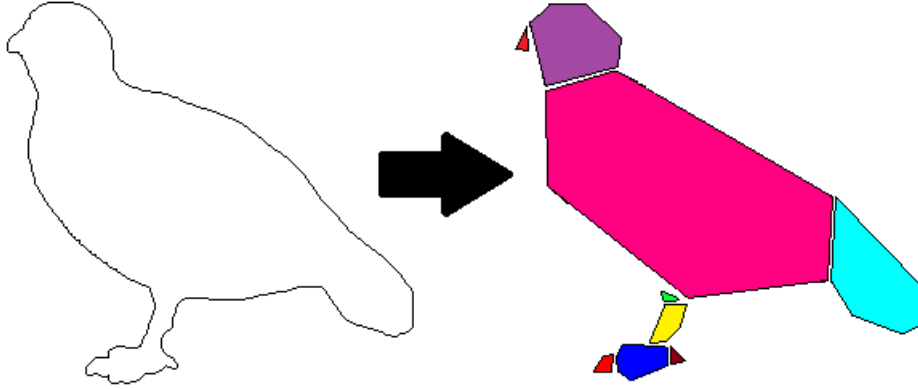


Figure 1.1: *Picture demonstrating the concept of shape decomposition.*

Upon investigating features that can be extracted, it was found that shape decomposition is one of the best ways to approach object recognition. An example of bird's shape decomposition is shown in figure 1.1. This is because if one investigates how humans identify complex shapes, it was found that humans tend to decompose such shapes into simpler, or more recognisable ones [7, 21]. These may include shapes among other like triangles, rectangles and circles. Thus, shape decomposition will be a good feature to make use of, but the human cognitive system, is very complex, and trying to portray this to a computational system is not an easy task [15].

The goal of this dissertation is to find a shape decomposition algorithm that can be improved in terms of speed and accuracy. Accuracy in this dissertation will refer to how closely the system relates to human-perception, while speed will depend on the time-complexity of the algorithm that is used. In order to ensure that the method is independent of colour, size or good quality pictures, shape-decomposition methods are looked at. The reasons for his choice will be discussed in Chapter 2.

Perceptual-based shape decomposition, like its name suggests, decomposes a shape into parts that do not overlap and that is consistent to human perception [22]. Decomposing a complex shape into visually significant parts comes naturally for humans, and turns out to be very useful in areas such as shape analysis, shape matching, recognition, topology extraction, collision detection and other geometric processing methods [1, 15]. Thus technology has been developed in order to better recognize objects.

After comparing several different shape decomposition algorithms which can be seen in chapter 2, it is found that the Minimum Near-Convex Decomposition (MNCD) [2] method is the most accurate according to human perception; has the most room for speed improvement; produces minimal, yet accurate decomposed parts; is invariant to noise, rotation, translation and scale and makes use of perceptual and geometric rules to obtain shape decomposition. Each of these concepts will be discussed in more detail in chapter 2.

The aim of this dissertation is to improve the MNCD method in such a way, that the time it takes to complete the shape decomposition is less, while improving the accuracy or keeping it the same. Furthermore the idea is to decompose a complex shape of an object into simpler more primitive shapes. This is done in order to make the classification process shorter, and thus also the whole recognition process. This then leads to an investigative question, which after some literature

has been surveyed, will help us shape a concrete investigative question.

1.2 Short literature survey

Different methods of shape decomposition can generally be categorized into two classes. The first class is motivated by psychological studies [23], while the second is by geometric constraints [24].

In psychological studies, a complex shape is decomposed into natural parts [25, 26]. The definition of natural parts is dependent on human conceptuality and can therefore be determined by investigating the way humans decompose different complex shapes. There does, however, exist several fundamental rules of perception that have been developed from cognitive science principles. Some of the most well-known rules include the *minima*-, *short-cut*- and *limbs-and-neck rules* [27, 28].

The geometric studies aim to decompose shapes into geometrically related parts [29]. A very simple example of this would be when a compound shape like a trapezium, knowing its properties, is broken down into triangles and a rectangle. The most popular geometric device that is used is convexity due to the fact that most convex parts have decent geometrical and topological properties. It is also an important constraint, as convexity plays a role in human perception [30]. The significant difference between the perceptual and geometrical properties are that one makes use of the shape properties and calculations to determine the decomposition results, whilst the other makes use of more visual properties and calculations related to the visual properties to determine the decomposition results.

In order to generally measure the execution of a category, two indexes are looked at namely, time complexity and the number of decomposed parts.

Time complexity is defined as the computational complexity that is used to describe the amount of time it takes for an algorithm to execute [31]. Keil [32] proved that the time it takes for convex decomposition can be written as: $O(n + r^2 \min(r^2, n))$. Here the number of vertices is represented by n and the number of *reflexes* by r - reflexes in their paper refers to any interior angle of a vertex with an angle greater than π . In [33], Rom and Medioni uses a Hierarchical Decomposition and Axial Shape Description (HDASD) method to recognise parts. For their algorithm, the time complexity is $O(n \log n)$, where n is the number of boundary points/vertices. Furthermore Ren, Yaun and Liu proposed a Minimum Near-Convex Decomposition (MNCD) method in [2]. In their method they break down complex shapes into a minimal number of "near-convex" components. In their work it was found that the time complexity can be written as $O(n^2)$.

The second execution measure is the number of parts that a shape has been decomposed into. Here Lien [9] proposed a method where a 2D-shape is decomposed into the minimal number of strictly convex parts. It is crucial to note that strictly convex methods produce a huge number of decomposed shapes [12]. In order to overcome this problem, Lien and Amato [16] proposed an approximate convex decomposition method. Their algorithm is designed to be more efficient since strictly convex decomposition produces a lot of unnecessary parts, and takes a longer time to decompose. Despite the improvement, there still remains two unsolved problems: redundant parts are produced and it is difficult to obtain visually naturalness [2]. To solve these problems, Ren *et al.* in their MNCD method [2] break down complex shapes into a minimal number of "near-convex" components as mentioned earlier.

Many methods make use of a combination of geometric and psychological techniques to decompose a shape, and some of the examples are the Minimum Near Convex Decomposition (MNCD)

[2], Perception-based Shape Decomposition (PSD) [17], Weighted Skeleton and Fixed-share Decomposition (WSFD) [34], etc.

Although there has been a lot of work done on improving the visually naturalness of shape decomposition, there is still room for improvement on the amount of time it takes to do so. This might be problematic in areas where real-time recognition is required. For example, in Ren *et al.* [2], it takes on average 3.97 seconds to decompose a hand. This might be due to some preprocessing being done, but that can still be considered a long time for real-time recognition, especially because this is only the time it takes to decompose, and not recognize as well.

1.3 Research question

In general, shape decomposition methods tend to try and obtain the least number of decomposed parts after decomposition has taken place [35]. Adding to this, the trend is also to try and get the decomposition as close as possible to the way humans decompose shapes [35]. Therefore, if one wants to improve the time it takes to decompose a complex shape into primitive shapes, the final question can be:

Which decomposition method will decompose a complex shape into the least number of simple shapes in the shortest amount of time?

1.4 Method

In this section the method that will be followed to answer the research question will be discussed. The steps that will be followed are listed below and will be discussed shortly thereafter:

1. Do a literature review.
2. Do an overview on the approach that will be followed to improve a method.
3. Implement the improved method.
4. Do experiments and obtain results.
5. Conclude by answering investigative question.

The first step would be do a literature study, which includes doing some background research on the proposed solution - shape decomposition. This will also include doing a literature review on different methods that have been used before. It is important to identify method with areas for improvement, and to focus on these methods.

Once the method that will answer the investigative question the best has been identified, a few improvements will be suggested. With these suggestions in mind, an overview will be done on these suggestions to better understand each them and to implement them to the best of their capabilities.

These improvements will then be implemented and a discussion on the ways it has been implemented will be done.

This is then followed by experiments and the results will be tabled and graphed. The experiments will include testing different parameters, and also comparing their outcomes. Lastly, the results of the improved method will be compared to the selected method to evaluate the outcome of the improvements.

This then lead to the conclusion, where the results and the future work will be discussed. This is also where the investigation question will be answered.

1.5 Validation

In general, to validate output one must be able to confirm that the results obtained are correct by measuring it against previous research, and the same output must be found when the same method is repeated by somebody else (it is repeatable). It is also important to go back to the investigative question and see if the improvements that were made was as intended.

Thus during the experiments, the results must be tabled and graphed. After this, the desired results must be drawn up and then these results needs to be compared to that of other research. In this case, to validate the results found in the proposed improved method, it will be compared with the results of other well known decomposition methods. For this specific case, the time that the decomposition takes place needs to be recorded, as well as the output results of the decomposition (in this case the number of parts). This will then be validated against the recorded information of previous decomposition methods.

In order to ensure that this improved method is repeatable, the implementation methods will be discussed in detail, as to be able to repeat these improvements. Furthermore, the experimental set-up must be explained in detail for the experiments to produce the same results. It is important to note the factors that might have an influence on the results, for example in this case it might be required to produce the computer hardware details, as this will have an influence on the speed of decomposition.

Finally, in order to check if the investigative question is answered, the results of the improved method will be compared to the parameters that was specified in the investigative question. That is in this case the results should shown an improvement in time, and the accuracy should be more or less the same.

1.6 Dissertation overview

To conclude this chapter, an overview of the rest of the dissertation will be given. In the next chapter, a background study will be done on shape decomposition, or specifically, the different types of shape decompositions that exist. This will then be followed by a study done on different shape descriptors, to give the reader an idea of what a shape descriptor is, and how it can be implemented to solve the problem at hand. After that a literature review is done on different shape decomposition methods. The main discussion will be on the time complexity of different methods, which will be compared to each other in order to find the best method to improve on. Next a discussion on the number of parts produced after decomposition of different methods is done. After that a comparison to identify the method that has the most room for improvement is done.

Chapter 3 then consists of an overview of the different algorithms that will be used in order to improve the method that was identified in Chapter 2. These algorithms include the Discrete Contour Evolution (DCE), Corner detection, Delaunay Triangulation (DT), and the Shape Decomposition algorithm that is going to be improved.

Chapter 4 will then be a discussion on how the improvements were implemented. This includes a discussion on determining a stopping criteria of the DCE, the concavity, curvature and moving of the corner points detected, discrete lines, how the cut set will be determined, how the mutex pairs

will be determined, the Ψ -concavity, the determination of the $\mathbf{A}$ and $\mathbf{w}$ matrices, how the Binary Linear Integer Programming (BILP) will be implemented and lastly how the simple shapes are to be identified.

In chapter 5 the experiments will be done and the results will be obtained. Experiments will be done on the time reduction and the number of parts produced after decomposition. Time reduction experiments include experiments on the different parameters Ψ , λ and β . That is to compare the amount of time reduction to every parameter while keeping the other parameters constant. The same was done for the number of parts. After that the time reduction results will be compared to other methods, as well as the number of parts produced. Finally the simple shape output results will be discussed.

Lastly, chapter 6 will be a conclusion on this dissertation, where a conclusion to the investigative question will be made, as well as some suggestions on future improvements.

Chapter 2

Literature study

In this chapter background research as well as a literature study will be done. The background research will aim to help develop a better understanding and provide some essential background information of the problem mentioned in chapter 1. Therefore research will be done on shape descriptors in order to see the different types of methods used to describe a shape. This is also done to see where the shape decomposition method falls into place, and how the decision to make use of shape decomposition in order to solve our scenario from chapter 1 is made.

This will be followed by a literature study on some of the different shape decomposition methods and the selection of the final method, the MNCD method. The selection of this method is based on two criteria: how meaningful the decomposition is and the speed of the decomposition.

2.1 Background research

To start the background off with, some terms will be defined. This will give a better understanding of some of the terms that are used throughout this dissertation. An overview of different shape descriptors will be given in order to contextualize shape decomposition. This will then be followed by research on different types of shape decomposition algorithms and concluded with a discussion of this subsection topics.

2.1.1 Terminology

Before we start the background research, a few terms will first be defined.

The first and most used term is convexity. In this dissertation, *convex* will be used to describe an outline curved like the exterior of a circle. If a part or shape is referred to as being *convex*, then it is a polygon with all of its interior angles less than 180° , that is all of its angles are pointing outwards or away from the centre of the polygon [13]. Formally, for an object S to be convex any two points, p_1 and p_2 , and the line segment connecting these points must be contained within the object S [15]. This is illustrated in figure 2.1 (a). The pink line connecting points p_1 and p_2 lies inside the shape and thus indicates that the shape S is convex.

Naturally, *concave* will then be defined as the opposite of *convex*. That is, a curve that is described as the interior of a circle or a polygon with at least one interior angle that is more than 180° [13]. This is illustrated in figure 2.1 (b). The pink line connecting points p_1 and p_2 lies outside the shape S and thus indicates that the shape is concave.

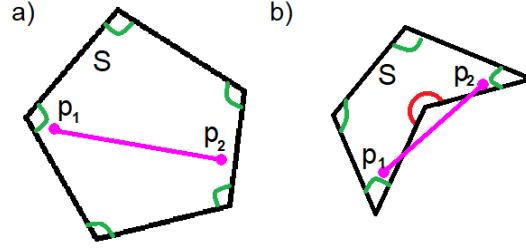


Figure 2.1: Figure illustrating the difference between convex (a) and concave (b), with the curves at the top, and the polygons at the bottom. The green lines indicate interior angles less than 180° , where the red line indicates interior angle greater than 180° .

Visual parts are then defined as the parts of a shape that the human cognitive system is the most likely to separate from the other parts of the same shape [30].

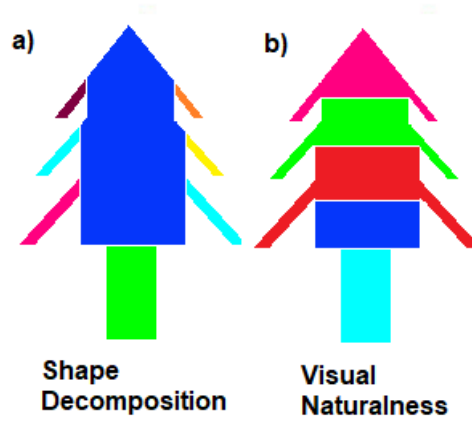


Figure 2.2: Figure illustrating the difference between computational decomposition (a) and human perceptual decomposition (b). This image is used to demonstrate visually naturalness.

Vertex or vertices are defined as the angular points that make up a polygon, polyhedron or other figures. Thus, a point where two lines meet to form an angle [32]. This is illustrated in figure 2.3 (a)

Notch or reflex, these two terms mean the same thing and can be defined as a vertex with an interior angle greater than π or 180° . So instead of describing the shape or a curve as a whole, a point is described by its angle [32]. This is illustrated in figure 2.3 (b)

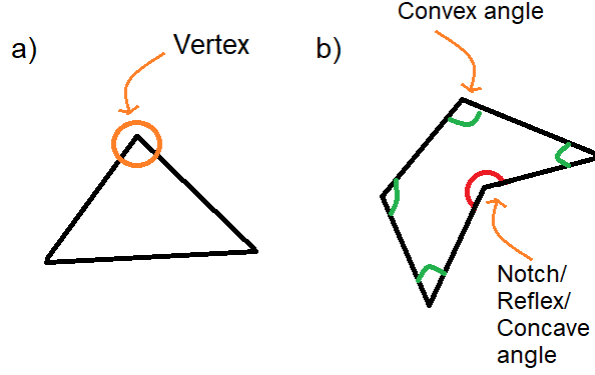


Figure 2.3: Figure illustrating a vertex (a) and a convex and concave angle (b). Concave angles are also referred to as notches or reflex.

Shape descriptors are computational tools used for analysing image shape information, and can be described as mathematical functions that produces numerical values when applied to an image. For example eccentricity is used to describe the ratio of the major axis to the minor axis of the bounding ellipse of a shape. In this way a simple ratio is used to describe the shape [36].

Concavity trees are data structures used for describing non-convex two dimensional shapes [37]. A concavity tree can be viewed as a rooted tree where the root corresponds to the base part of the object. The next level of the tree contains nodes that represent the concave parts of the object [37].

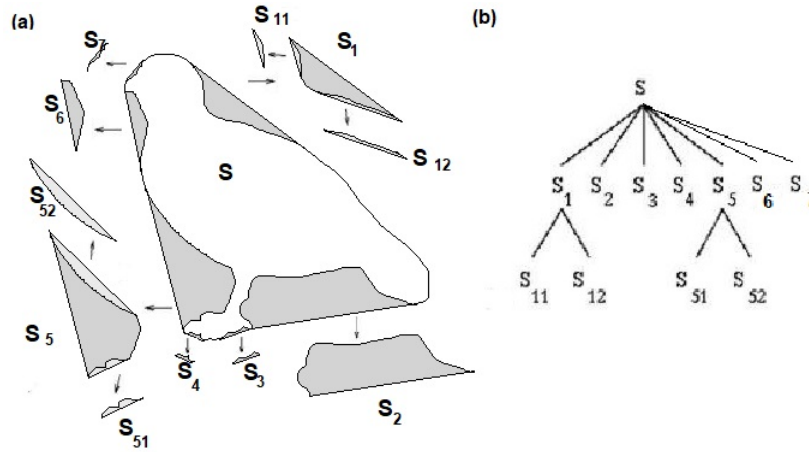


Figure 2.4: Picture showing how the different concave parts fit into a concavity tree of the shape [3].

Mutex pairs is short for mutually exclusive pairs. If the line joining any two vertices that lie on the contour of the shape, say p_1 and p_2 , and any points on this line are located outside the contour, or intersects with it, $(p_1; p_2)$ is known as a *mutex pair* [25].

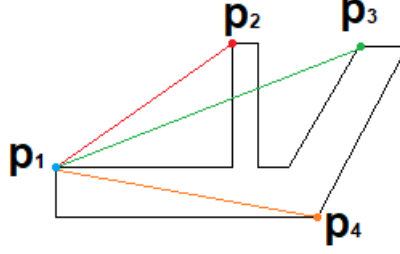


Figure 2.5: Figure illustrating the concept of mutex pairs, where $(p_1; p_2)$ locates completely outside of the contour and $(p_1; p_3)$ intercepts with the contour, and are thus mutex pairs. However $(p_1; p_4)$ are completely inside the contours and is thus not a mutex pair.

A *Morse function* is defined as the projection of a point in the given direction, for example if we have $f(p) = \langle d, p \rangle$ then $f()$ is the *Morse function*, d is the unit vector representing the direction and $\langle . \rangle$ is the dot product between the point p and the unit vector d [22].

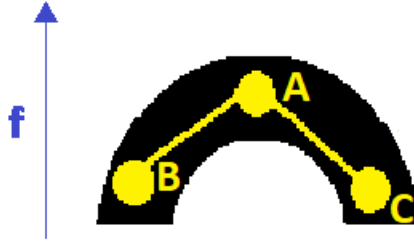


Figure 2.6: Figure illustrating the concept of a Morse function $f(p)$ where the green line is the Morse function of the half doughnut.

Natural decomposition will be defined throughout this dissertation as the decomposition that the human cognitive system uses to break down a shape into its minimal number of parts [38].

Near-convex parts can be defined as parts that are not strictly convex (all of the interior angles are not less than 180°). This concept was introduced because as decomposition of exact convex parts produces a large number of small redundant portions that are sensitive to small variations in shapes [2]. Figure 2.7 is used to demonstrate this concept.

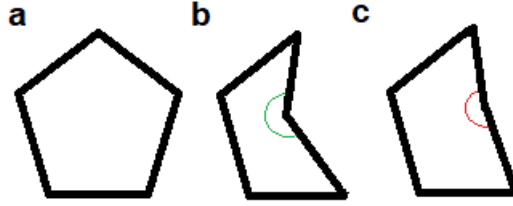


Figure 2.7: Figure illustrating the concept of near-convex. As can be seen in (a) the pentagon is strictly convex, in (b) there is a rather large angle (green) to indicate that it is concave. In (c) there is a smaller angle which if allowed can be classified as a near-convex shape

2.1.2 Shape descriptors

Shape descriptors can generally be described as a set of numbers used to describe a shape [39]. These sets of numbers try to convey shapes in such a way that agrees with human intuition [39]. In this section a discussion of the different shape descriptors that were considered for the scenario mentioned in chapter 1 will be done. The ultimate goal of this section is to understand the different types of descriptors and how the decision to make use of shape decomposition came about. When looking at the scenario in order to identify a bird for example, the shape descriptors that can be considered include: graph-based representation, shape signatures, convex hull, medial axis and shape decomposition

Graph based representation

In order to obtain a graph basic geometric properties are extracted from binary shapes. The first step is to convert the binary image to a polygon approximation vector image of the contours. After this is done, the primitive properties are represented as nodes while the relationship between nodes form the edges of the graph [4]. Advantages include that it is flexible and tolerant to scaling, rotation and translation [4]. Disadvantages of this method include that converting vectors into quadrilaterals can become quite complex, and that the recognition of mixed shapes still needs further work [4]. This process is shown in figure 2.8.

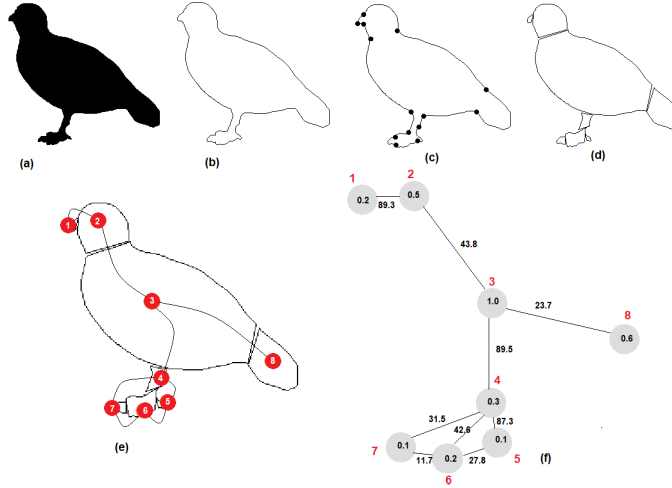


Figure 2.8: Picture to show the graph representation process: (a) original image, (b) contours of the original image, (c) convert contours into vectors, (d) obtain primitive quadrilaterals, (e) obtain number of nodes and (f) use vector length for node and angle for graph [4].

Shape signature

Shape signatures are one-dimensional mathematical functions obtained from the shape's contour and may be used as a shape descriptor [5]. Some of the most used shape signature methods used are the centroid distance function, the chord length distance, the angular function, the triangular centroid area, triangle area representation, the complex coordinates and the farthest point distance. Shape signatures are computationally simple, however they are sensitive to noise and slight changes in the boundary can cause large errors in matching [39].

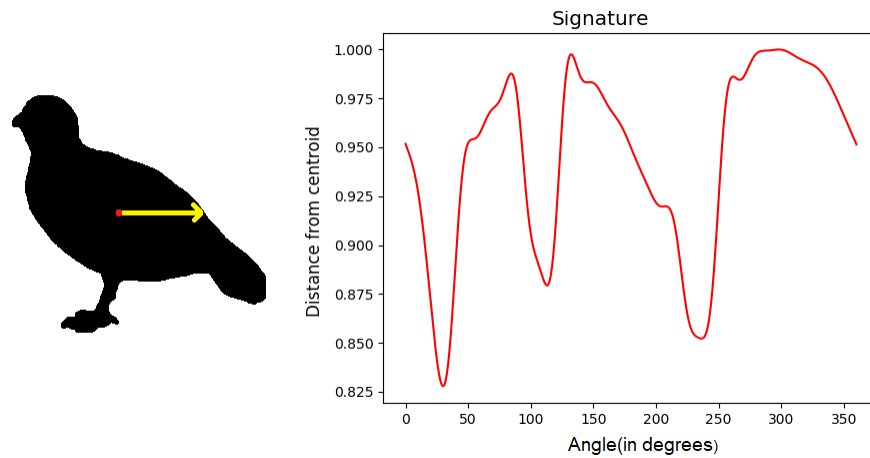


Figure 2.9: Figure showing the shape at the left, with a yellow arrow indicating the distance to the boundary. As this arrow moves along the shape boundary, the distance is mapped against the angle that this 'arrow' forms with the centroid of the shape (red dot). The results are shown in the graph next to the shape [5].

Convex hull

A *convex hull* is defined as the smallest convex polygon that completely contains an object [40]. In order to represent a shape using a convex hull, a recursive operation is used to obtain a concavity tree. After the recursive operation is done, each concavity can then be described using its area, chord length, maximum curvature and the distance from its maximum curvature point on the chord [3]. This process is shown in figure 2.10. Advantages include that it is rotation, scaling and translation invariant and it is robust against noisy shape boundaries. Disadvantages include that extracting the convex hulls proves to be a troublesome process [39].

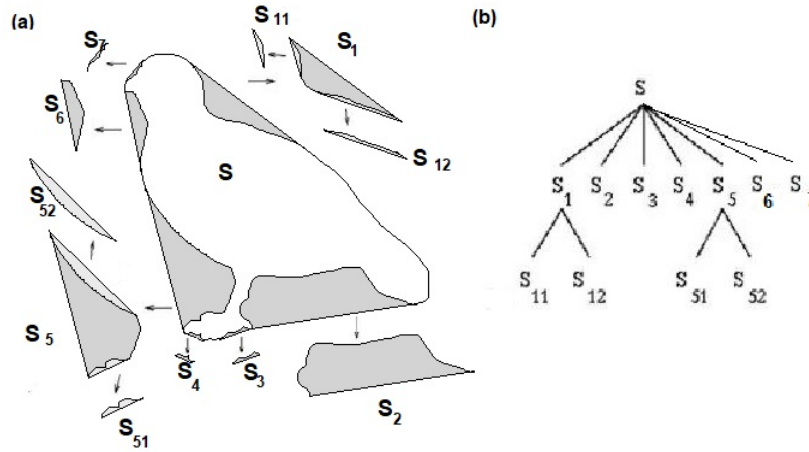


Figure 2.10: Picture showing (a) the convex hull of a shape and its concavities and (b) the concavity tree of the shape [3].

Medial axis

Another way of representing a shape is by its area skeleton. Skeletons are described as the associated set of central traces along the parts of a figure and is obtained by using basic lines and arc pattern structures [41]. The medial axis can be obtained by getting the locus of centres of the maximal circles that fit within the shape [40]. An example of the medial axis is shown in figure 2.11. Advantages of this method include that it is invariant to scale, occlusion and rotation. Disadvantages include that the computation of the medial axis is a challenging task, as it is sensitive to boundary noise [42].

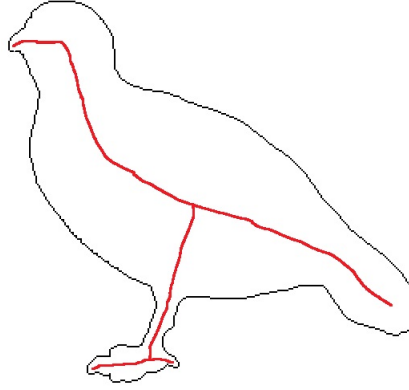


Figure 2.11: *Picture showing the medial axis (red) of the shape (black).*

Shape decomposition

Shape decomposition can generally be defined as the complete partition of a single, connected region(shape) into disjunct sets of connected regions (parts) [6]. Shape decomposition is a thoroughly studied field, and include many different methods to achieve the same goal [6]. In choosing the correct method of decomposition, one can obtain advantages like rotation, noise, scale and translation invariance. Disadvantages include that these methods are computational complex because decomposing a shape into parts that agree with human perception proves to be a difficult task [2]. Figure 2.12 demonstrates a simple example of shape decomposition.

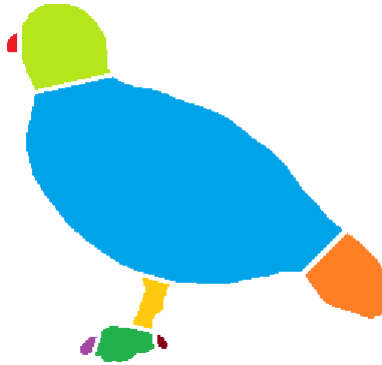


Figure 2.12: *Picture showing an example of shape decomposition [6].*

Discussion

Thus in order to decide which shape descriptor to make use of to solve the scenario mentioned earlier, the important properties mentioned in the above section will be summarized in table 2.1. Here it can be seen that the most important criteria that was look at, and which separates the different descriptors from each other includes the computation complexity, the invariance to noise, rotation and scale and whether not the background will have an influence on the task at hand.

Table 2.1: Table showing a summary of all the criteria that will be used to choose a shape descriptor to make use of.

Shape descriptor criteria					
	Graph based	Shape signature	Convex hull	Medial axis	Shape ecomposition
Time Complexity	No	Yes	No	No	No
Invariance	Noise	Yes	No	Yes	No
	Rotation	No	No	Yes	Yes
	Scale	No	Yes	Yes	No
	Background	Yes	Yes	No	Yes

Thus, it has been decided to make use of shape decomposition to extract features of an object as it is invariant to noise, rotation and scale, and the background does not have an influence on its results. In the next section, the shape descriptor will be discussed in more detail, as well as the different methods that make use of shape decomposition to decompose objects into meaningful parts.

2.1.3 Shape decomposition

In this section shape decomposition, will be discussed in more detail. First a formal definition will be given followed by a discussion of why shape decomposition is important in different tasks then a discussion on the different types of shape decomposition that can be found.

Part-based representation can be defined as the representation of a shape or an object in a number of its decomposed 'natural' parts [43]. Natural or meaningful parts will be discussed shortly. Decomposing a shape can lead to a better analysis, as well as an improved understanding of a shape by simplifying it into simpler parts [2, 30, 44].

Several studies have shown that when humans view objects we spontaneously divide the object into parts [45]. In human vision, decomposing a two-dimensional (2D) shape into visual meaningful and functional parts, is a fundamental process [38]. This visual concept obtains the most essential distinguishing features of the shape to deduct an initial recognition and then details are added to complete the task. Furthermore, it was found that surface characteristics of an object, such as colour and texture, play a secondary role in recognition and real-time recognition is mediated by the edge-based information [46].

Shape decomposition is used widely in image processes that include shape recognition and recovery [25, 2], skeletonization [34, 17, 47] and path planning [2, 48]. Possible examples of shape decomposition are decompositions into convex, spiral and monotone polygons [49].

Now that a better understanding of shape decomposition has been developed, we can have a closer look at the different types of decomposition.

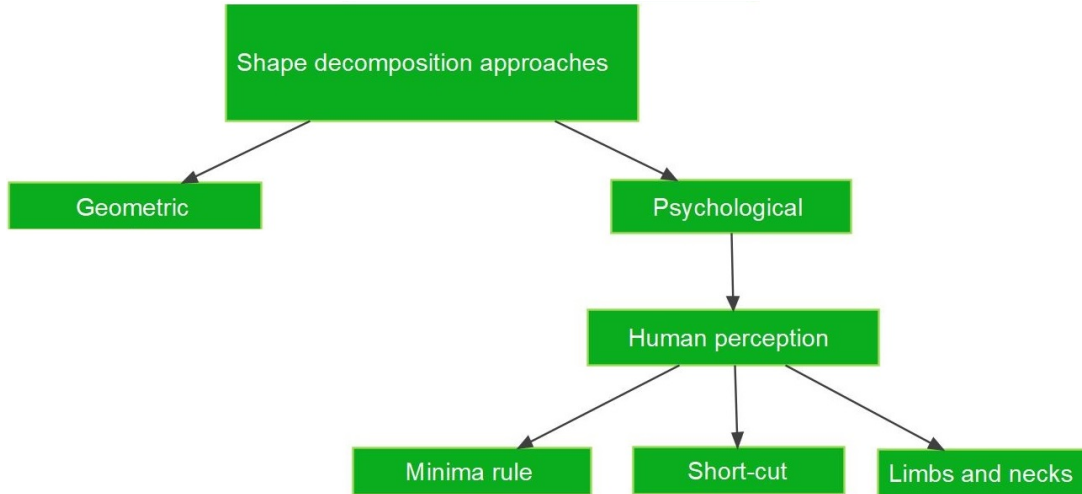


Figure 2.13: Graph explaining the different types of Shape decomposition [2].

2.1.4 Different types of shape decomposition

As can be seen in figure 2.13, most of the known shape decomposition methods can be classified into two classes. The first class is motivated by psychological studies [23], while the second is by geometric constraints [24]. These will be discussed shortly, with some examples of methods that has made use of each of these methods

Psychological

Driven by psychological studies, the first class is proposed to break down objects into natural parts [25, 26]. Natural parts can be defined as being dependent on the cognition system of humans and therefore has no verifiable explanation. There are, however, several fundamental perceptual rules that have been developed from cognitive science principles.

The first is called the *minima rule*. This rule points out that the human visual system is trained to observe boundaries of a shape at concave creases or at negative minima of the shapes' curvature [23]. In [50], Latecki and Lakämper proposed a method where they use discrete contour evolution in order to determine the minima on the curvatures, and so decompose a complex shape. In [7], De Winter states that Biederman [30] has found support in their experimental data that the minima rule can be used as a means of segmentation. It is very seldom that this rule is used on its own, and is mostly used in combination of some sort with the next rule.

Another well known measure is the *short-cut rule*. Here the rule points out that the human visual system favours the shortest viable cuts when we decompose complex shapes [27]. In [51], Siddiqi and Kimia only use the short-cut rule for calculating the cost of a cut.

The last rule to discuss is the *limbs-and-necks rule*. This rule builds on the minima rule, but here the following criteria defines when a cut is a limb or a neck [52]. The minima rule is depicted in figure 2.14 (a). *Limbs* are defined as being formed when at least one negative minima point can be connected in such a way that forms a continuous line with the contour, as can be seen in figure 2.14 (b) [7]. *Necks* are then defined as when a inner-circle with maximum radius is also the local

minimum of the outer-diameter, as can be seen in figure 2.14 (c). This method is used in [53] and in [28] to determine the size of object and to decompose shapes respectively.

De Winter and Wagemans [7] did a large scale study on the decomposition of object outlines into parts. In their study they asked a large number of people ($N=201$) to divide shapes into parts, and then they compared these results with models used for object partitioning. Their findings revealed that the minima rule has the greatest influence on segmentation of shapes, which is then followed by the short-cut rule and lastly the limbs and neck rules.

As can be seen all of these rules can be connected to the human cognitive system and are usually used in combination with each other to determine the more natural cuts, which in most cases tend to be the least number of cuts as well. Thus, when we are looking for a decomposition method, it is important that perception rules are present.

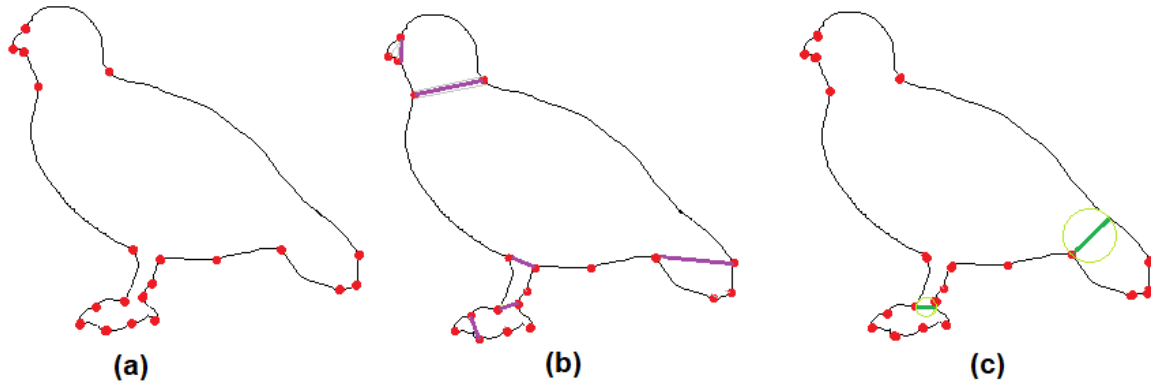


Figure 2.14: Picture to demonstrate the minima rule (red dots), the short-cut rule (purple lines) and the neck- and-limb rule (green circles and lines) [7].

Geometrical

The second class is driven by geometric descriptors and aims to decompose shapes into geometrically related parts [29]. The most popular geometric device that is used is convexity. This is due to the fact that convex parts mostly have decent geometrical and topological attributes that permits reliable mathematical operation and improves the effectiveness of algorithms. It is also a crucial constraint, as convexity plays a role in the human perception [30], as it has been found that humans by nature tend to decompose shapes into visual parts that are convex.

One of the approaches in this category is known as morphological skeleton transformations (MST). Here a union of maximal disks that are contained within the complex shape are used to represent it. Another method known as morphological shape decomposition (MSD), is similar to the first, but there exists no overlap between the disks [54]. In these methods, the shapes are decomposed primarily by using morphological operations.

Geometrical based shape decomposition can be summarized as decomposition methods that makes use of mathematical expressions to decompose complex shapes. Thus, this is a very important point to consider when one wants to improve time, as mathematical expressions tend to take less time than working on images. This will also be an important point for us to consider when we compare different decomposition methods with each other, as we aim to improve on existing

methods time complexity.

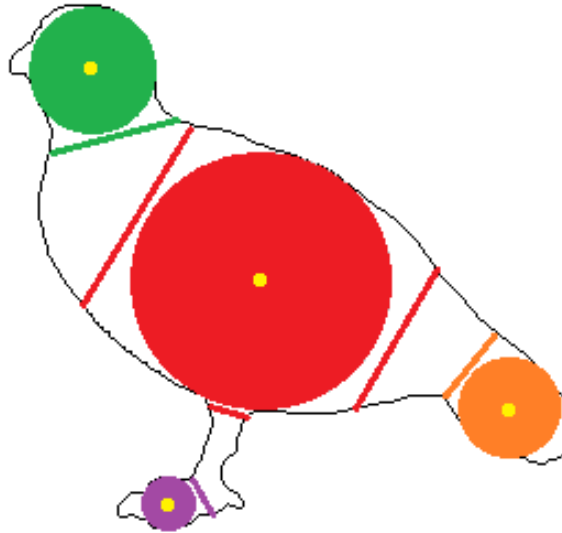


Figure 2.15: Picture to demonstrate the use of geometrical methods to decompose the object. In this case morphological operations were used to determine disk of most importance [8].

2.1.5 Discussion

In conclusion to this section, it is important to note that both the psychological and geometrical types of decompositions play important roles in the decomposition and recognition of objects. The psychological rules are simple and easy to implement, while the geometrical methods are quite complex. The advantage that the geometrical methods hold are that once determined, the geometrical properties of the decomposed shape is easy to use, while the psychological rules might need more computation before use. Thus, because the use of psychological rules are less complex and more solid in implementation, methods that make use of the psychological rules are favoured above geometrical methods.

2.2 Literature review

After doing some background research it has been found that shape decomposition is the desired method of feature extraction, and that the psychological methods are to be favoured as these rules are set and therefore it will be easier to implement. Now in order to determine the desired shape decomposition method, a literature review will be done. This section will start with a discussion of what is considered a good shape decomposition method. This will be followed by a discussion on different types of shape decomposition methods in terms of the different criteria that classifies a shape decomposition algorithm as a good algorithm.

2.2.1 Quality of the solution

In this section a specification of what exactly constitutes a good method will be given. This section is discussed first in order to determine which criteria will be looked at before the final choice of a shape decomposition method is made.

Human-perception

The first and most important criterion that will be looked at is human-perception. As mentioned earlier, this task proves to be difficult to define, as perception is different in each person [2]. In order to determine a general human-perception criterion, it was decided to create a short questionnaire.

This questionnaire is set-up in such a way to try and make the results as close to the South-African population as possible. A more detailed discussion on the set-up of the experiment is given in appendix B. The pictures that were selected in the questionnaire represent the most commonly used pictures found in most of the articles to be able to compare the results of the shape decomposition methods. These pictures are all obtained from the MPEG-7 and the Animal datasets, shown in appendix C. The pictures used for this experiment are shown in figure 2.16.

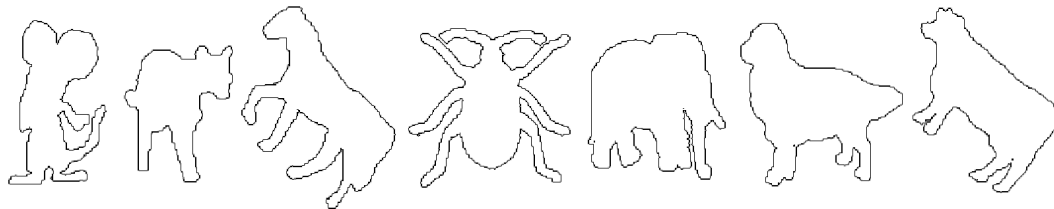


Figure 2.16: *The outlines of the most commonly used shapes found in different articles [2, 9, 10, 11, 12].*

A total of 100 people were asked to participate in the questionnaire. The results were captured and is shown in figure 2.17. Each questionnaire was looked at and lines drawn on the final result. The lines are drawn at a very low transparency, and will become darker the more people selected the same area as a cut. Then the final cuts was drawn by selecting the darkest areas.

It is also important to note that the number of cuts where selected looking at the average number of parts. That is, all of the questionnaires where recorded and the average of the number of parts where determined. Then the number of cuts are determined by subtracting one from the number of parts. That is, if we have n number of parts then there will be $n - 1$ number of cuts. The complete set-up, as well as the final transparent line drawings are discussed in appendix B.

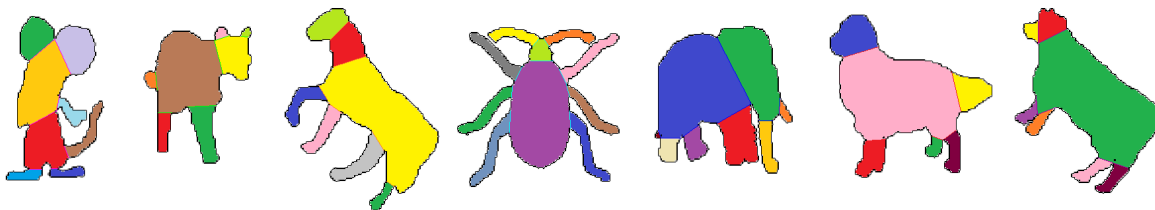


Figure 2.17: *Picture illustrating the results of shape decomposition done by humans through the use of questionnaires.*

Thus, to conclude this subsection, in order to determine the a percentage that a method deviates from human perception, the total amount of human perception cuts that correspond with the cuts of the method being investigated, is subtracted by the cuts of the method then divided by the total number of cuts of the human perception. This is then multiplied by 100 to obtain a percentage. That is, if c_{hp} represents the number of cuts that correspond to the human perception, and c_m represents the number of total cuts of that method, the percentage the method deviates from human perception, $\%_{deviation}$, can be calculated by:

$$\%_{deviation} = \frac{|c_m - c_{hp}|}{c_{hp}} \times 100 \quad (2.1)$$

This will be determined for each picture, and then the average of all the pictures will be used as the percentage a method deviation with human perception.

It was decided that a method should have a deviation of at least 35% of the human perception experiment in order for it to be classified as an average solution, 21-30% to be classified as a good solution, 11-20% to be very good and 0-10% to be described as an excellent solution. This does however have one flaw - what about the cuts that don't fall in the general cut area? This question will be solved in the next subsection by using the number of parts.

Number of parts

The second criterion that is going to be looked at is the number of parts. In order to make up for the flaw previously mentioned, the number of parts will be used. Here, instead of looking at the area where the cuts are produced, the average number of parts of the different methods will be looked at. Here, the average number of parts for each picture is determined. The same questionnaires as mentioned earlier are used to determine the number of parts humans will decompose the shapes into. To do this, the average number of parts, μ_{hp} , of each picture is determined, as well as the standard deviation, σ_{hp} , thereof.

In order to do this each questionnaire from the above mentioned experiment was used as the mean for evaluation. The standard deviation was calculated for each picture and this will then be used to determine if the method falls within one, two or three standard deviations of human-perception.

It is also important to note that the number of parts of a shape must be at least greater than one if there exist one minima-point in the picture [2]. This means that with one minimum point on the contour, the shape can be divided into two parts. Another way to determine if a shape can be divided into more parts is to determine whether the shape is convex or concave [13]. If the shape is concave, that is one vertex is greater than 180° , then the object can be decomposed into two parts. Thus, the number of parts will be greater than the number of convex vertices [13]. This is demonstrated in figure 2.18.

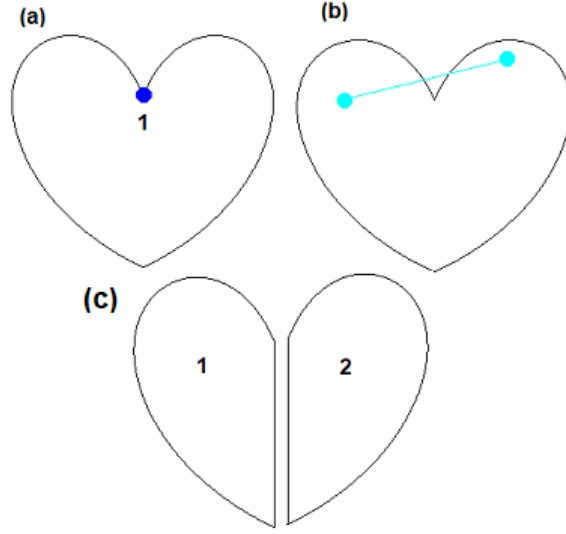


Figure 2.18: Picture to demonstrate how to determine the minimal number of parts. (a) shows that there exist one concave vertex, while (b) shows that the shape is indeed concave. (c) demonstrates that with one concave vertex, at least two parts can be formed [13].

These values are then compared to the number of parts, n_m of each method which is subtracted from the average number of parts, μ_{hp} , and the absolute value thereof is determined. This value is then compared to the standard deviation of the human perception, and given a number depending on the number of standard deviations the method is away from the average number of parts. Mathematically it can be determined by:

$$diff_m = |n_m - \mu_{hp}| \quad (2.2)$$

and then the value of $diff_m$ is compared to the amount of standard deviations, that is, for every deviation the average number of parts are above or below the average number of the human perception parts, a score of that deviation is given. The methods will then be ranked from the lowest to the highest average, en given a score accordingly - lowest will be given a "poor", and the highest an "excellent"

In conclusion to this section, it can be seen that the number of parts will also be used as a measure of how accurately the method compares to that of human-perception. This measure along with the assumption that a part is to be decomposed into at least $n + 1$ parts if n is the number of convex vertices. In the next subsection, the time complexity will be used as a criterion to determine how 'good' a decomposition method is.

Time complexity

The third criterion that will be looked at is time complexity. This will be discussed in this section. Time complexity can be defined as the time taken by an algorithm to run as a function of the length of the input [55]. Order of growth is how the time of execution depends on the function at hand, and there are three notations that can be used to describe the time complexity [55]. In this

dissertation we will only consider the O -notation. The O -notation is used to denote the asymptotic upper bound.

Simply put, this function is used to determine the average maximum time it will take to execute an array with length n . This is mostly given as a function itself - for example $O(f(n)) = n^2$ can be interpreted as an exponential complexity growth with an increase in array length, and that the maximum time it takes for a array with length n will be at most n^2 .

Thus, the less complex the time complexity of an algorithm, the faster the solution will be obtained. Therefore, in order to create a good measure of time complexity the length n of each picture that was used for the questionnaires mentioned earlier will be used, and the average of the time complexity will be used as the measure to decide if the time complexity criterion contributes to an average or excellent solution. Generally, a lower number will be better for use and thus will be more favourable above a higher time complexity.

In our specific case, re-inventing the wheel seems like an unnecessary task. Thus, there already exist methods that can decompose shapes, some of which can do in a quick time, while others do it in a long time. Therefore, in general the less time complex algorithms will be chosen as this will perform faster. It is important to note that this does not always mean that the algorithm will perform better. In this dissertation a few factors will be looked at simultaneously to evaluate the performance of an algorithm.

In conclusion, time-complexity will help to determine the speed of a possible solution, and how one would go about to improve on a solution. And in our case a method that makes use of a less complex algorithm will thus yield a better score. In the next section, a few different types of invariances will be discussed, and how this will effect the choice of method to be improved on.

Invariance

The fourth criterion that will be looked at, is invariance. This is a quite common criterion to look at in image processing as this can prevent a lot of trouble when chosen appropriately for the task at hand [56]. Thus four most common invariances will be shortly discussed in this section, and how each contribute to our specific scenario mentioned in chapter 1. The first is translation, followed by rotation, size and lastly noise invariance.

The first invariant that will be looked at is translation invariance. This refers to the translation, borrowed from geometry, when an object is moved an amount of pixels in any direction [57]. Thus for a method to be translation invariant the same object must be able to be translated and still give the same results. For example with the bird in the photo, the position of the bird will not always be the exact same distance from the origin. Therefore, the method needs to be translation invariant for our application to be able to work.

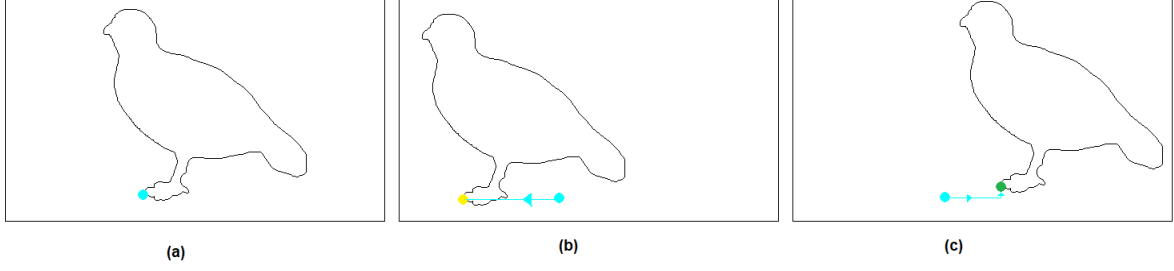


Figure 2.19: Picture to demonstrate translation invariance - (a) the original position with the blue dot as the origin, (b) shows a translation of 10px left, while (c) shows a translation of 7px right and 5px up.

The second invariant that will be considered is rotation invariance. This, as mentioned above is also borrowed from geometry, and refers to an object that has rotated a certain amount of degrees around the origin [57]. Thus, this is also important for if the picture is taken from a different angle to the original photo, it should still be recognised.

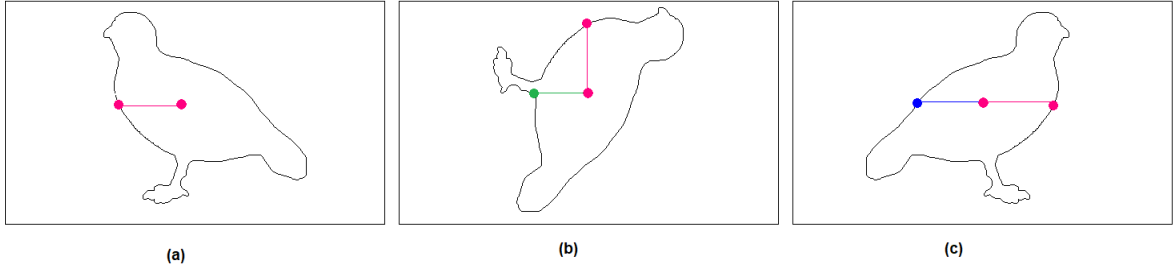


Figure 2.20: Picture to demonstrate rotation invariance - (a) the original position with the pink dot as the origin, (b) shows a rotation of 90° , while (c) shows a translation 180° in an anti-clockwise direction.

The third invariant to consider is size invariance. Here, borrowed from geometry, it refers to the fact that the object should still be recognised even though it might be a scale 1 : 2 smaller or larger than the original image [57]. Thus this is also important in our scenario as the distance that the bird is away from the camera might not always be the same as that of the original image. Therefore it is important for the method to be size invariant.

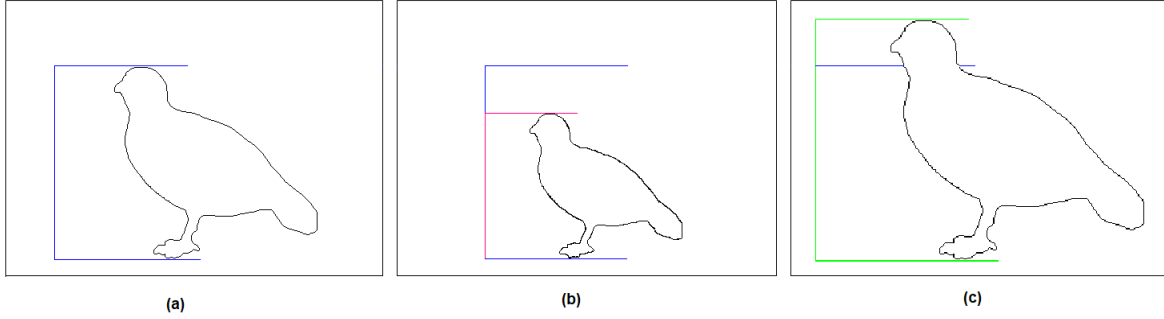


Figure 2.21: Picture to demonstrate size invariance - (a) the original size with the blue line to indicate size, (b) shows a resize of 75%, while (c) shows a translation of 125% of the original size.

The fourth and last invariant that will be considered is noise invariance. This usually goes hand-in-hand with distortion. Image noise is described as an aspect of electronic noise, and can be caused by several factors including the sensors or circuitry of the camera being used [58]. Distortion on the other hand can be described as a variation from rectilinear projection [59]. That is, when straight lines in a scene don't appear straight on an image taken of that same scene [59]. Examples of distortion and noise can be seen in figure 2.22.

As there are four invariances considered, a point for each invariance that the method has will be given, and so a maximum of four, and a minimum score of zero can be given to the method.

Looking back at our bird scenario, this invariance is important as an image taken with some noise, or distortion should still be recognised correctly. For this to take place, the feature extraction method should also suffice, and thus, it is important to look at methods that are distortion and noise invariant.

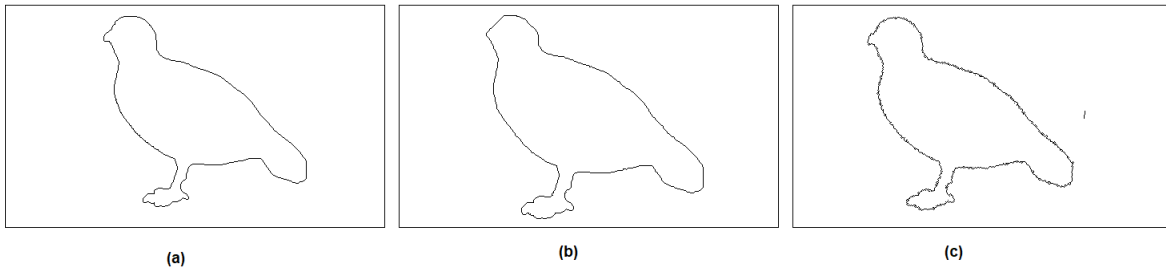


Figure 2.22: Picture to demonstrate noise invariance - (a) the original image, (b) shows slight distortion in the image (c) shows noise in the original image.

In conclusion to this subsection, it can be seen that the four invariances that were looked at contribute to solving the scenario at hand, and will be considered as criteria that contribute to a good method.

Now that all of the different invariance criteria has been discussed, the type of shape decomposition will be looked at as the last criteria for selection.

Type of Decomposition

The fifth and last criterion that will be considered is the type of decomposition method used. As mentioned earlier, there are mainly two types of shape decomposition methods used to classify a shape decomposition method according to the computation methods used. The first being psychological and the second being geometrical methods.

Of the two methods, the psychological method seems to be more of interest in our scenario, as there is a set of rules that can be used to decompose a shape. Compared to the geometrical methods, the psychological methods seems easier to implement, and will be less complex in terms of computation [25, 29].

In conclusion, a decomposition method that makes use of psychological decomposition will be considered as excellent, one that makes use of both geometrical and psychological will be very good, while methods that only make use of geometrical methods will be classified as good.

Now that the different criteria that is used to describe a method as a good method have been discussed, the different shape decomposition methods will be compared to each other. In order to compare the different shape decomposition methods with each other in a fair way, each method will be given a score based on all of the criteria mentioned above. Each method will be discussed shortly and then a table will be used to summarize that method. These methods will then be compared with each other in a final table which will make it a bit easier to compare the methods with one another.

To summarise the different criteria used to describe a method, the table 2.2 below is used.

Table 2.2: Table showing a summary of all the criteria that will be used to rank methods from highest to lowest in terms of improvement required.

Measure	Poor	Average	Good	Very Good	Excellent
Meaningfulness (%)	<50	51-60	61-70	71-75	>76
Number of Parts (STD)	>4.0	3.1-4.0	2.1-3.0	1.1-2.0	0-1.0
Time Complexity	Least		Middle		Most
Invariance	0	1	2	3	4
Shape Decomposition Type	G		P&G		P

To recap the different criteria, it was decided that meaningful cuts will be considered average when 51-60% of the cuts lie agree with those cuts of human perception, good will be 61-70%, very good is 71-75% and excellent will be 76% and above.

For number of parts, average will be between 3-4 standard deviations, good will be between 2-3, very good will be between 1-2 and excellent will be between 0-1 standard deviation of the average number of parts produced in the experiment. These values are chosen as the average deviation of each method will be calculated and will yield values with decimals.

For time complexity, the length of the pictures will be inserted into the respective time complexity functions in order to determine which time complexities are greater than the other. The greatest time complexity receiving excellent score, while the least complex receives a poor. This is because the improvement want to be made in the complexity of the methods that already exist.

For invariance, for each invariance of the four discussed earlier a score is given. Therefore, the more invariant the method, the greater the score.

Lastly, the type of decomposition is decided that if the method makes use of geometrical shape decomposition, it will score a poor, if a combination of geometrical and psychological is used, a score of good is given, and if only perceptual rules are used, an excellent is given.

The excellent to poor scale of measure is then converted to a score of between zero to four in order to have a fair measure of 'meaningfulness'. That is, for every score that a method receives, a number between zero and four will be given and summed to determine which method shows the most potential for improvement.

Table 2.3: Table showing the conversion of scores determined by the different criteria.

Score Convert	
Poor	0
Average	1
Good	2
Very Good	3
Excelent	4

To conclude this subsection, the different criteria and how the scores will be given for each has been discussed. The next section will include a literature study in several different methods, and each of the above mentioned criteria will be looked at as a base to determine which shape decomposition methods shows the most promise for improvement, or will be classified a 'good' method to improve on.

2.2.2 Shape decomposition methods

In this section the different shape decomposition methods will be discussed. Each method will be explained shortly and a figure will accompany the explanation where applicable. Then each algorithm will be discussed in terms of the different criteria mentioned above and at the end of this section all of the methods will be compared to each other in order to make a final decision as to the best shape decomposition method to make an improvement on.

Perceptually friendly shape decomposition

The first method to be discussed is Wang *et al.* [14]. In their paper they proposed a shape decomposition strategy that is focused on the analysis of the relationship between the part cuts and the segment points. This method is known as Perceptually Friendly Shape Decomposition (PFSD) [14]. In this method, the aim is to find perceptually friendly parts where human perception is taken into account when computing possible part cuts and when defining their costs.

In order to obtain results, they extract the Discrete Contour Evolution (DCE) and relevance measures as visual features reflecting human perception. This is shown in figure 2.23 (a). After that, they obtain the morphological skeleton of the object shown in figure 2.23 (b). This in combination with the DCE vertices are combined to identify areas where there will most likely be a cut - for example where the skeleton splits, and there is a local minimum on the contour close to this area.

This is shown in figure 2.23 (c). Based on this results, a minimization problem is solved to obtain the final cuts. The results of this method is shown in figure 2.23 (e)

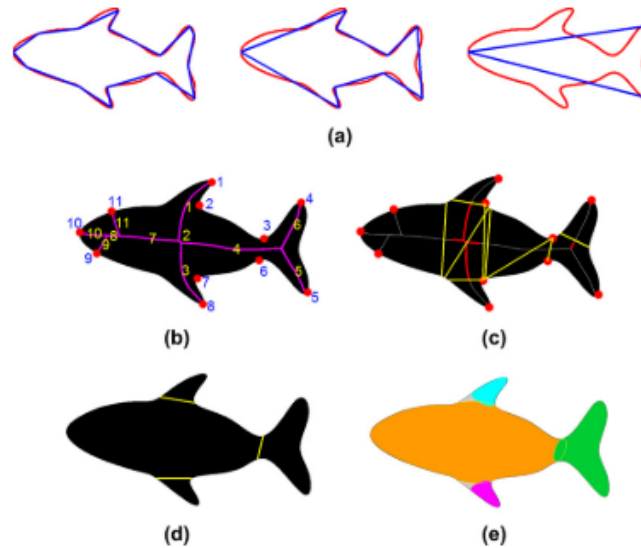


Figure 2.23: Some of the decomposition results of Wang *et al.* (PFSD) [14].

Summarizing the method on hand of the above criteria, the following has been found about the PFSD algorithm. It was found that an average of 73% of the cuts on these pictures are in similar areas than that of human perception. Thus, this method is a very good method compared to that of human perception. The PFSD method produced an average standard deviation of 1.0. That is thus an excellent score for this criterion. Upon investigation it was found that this method is translation, rotation, size, noise and distortion invariant [14]. Thus meeting all of the criteria looked at, and will thus also be an excellent solution to the scenario at hand. For their algorithm, the time complexity is $O(n \log n)$, where n is the number of boundary points [14]. This algorithm's time complexity is not dependent on another variable, but only on the input variable. Lastly, the type of method used to decompose shapes into parts is a mixture of geometrical and perceptual types [14]. This will score this method an average for type of method used, as we are looking for more perceptual types than geometrical types.

Thus to conclude, this method seems to offer a lot of promise, given the downfall of the type of method used, the rest seems to work well. The next method that will be discussed is the Minimum Near Convex Shape Decomposition (MNCD) method.

Minimum near convex shape decomposition

Ren, Yaun and Liu proposed a Minimum Near-Convex Decomposition (MNCD) method in [2]. In their method complex shapes are broken down into the minimal number of "near-convex" parts.

In this algorithm, a set of possible cuts are extracted by looking at the all of the vertices on the contour of an object. The cut set is then shrunk by rejecting cuts where both endpoints are convex. This is shown in figure 2.24 (b). By creating a non-overlapping matrix and a convexity matrix in order to solve two constraints, these are used together in a binary integer linear programming

problem to solve for the optimal number of cuts. Figure 2.24(c) demonstrates the non-overlapping constraint, while (d) demonstrates the convexity constraint. Parameters are used to ensure naturalness, convexity and a relation between two perception rules are also used to contribute to the optimal results. These are demonstrated in 2.24 (e) - (g).

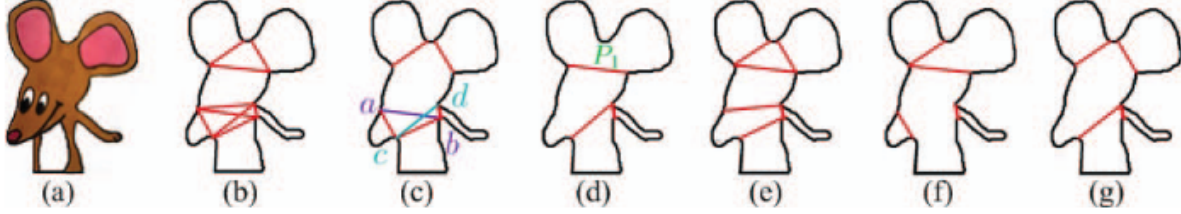


Figure 2.24: the shape decomposition steps of the MNCD algorithm [2].

Summarizing this algorithm it was found that an average of 77% of the cuts produced by this method agrees to that of the experiment. This can then be classified as an excellent solution. This method produces an average standard deviation of 1.1. This method however will then be given a very good for evaluation when it comes to the number of parts, as it is slightly more the one standard deviation from the human perception average. It was found that this method is also translation, rotation, size, noise and deviation invariant [2, 25]. Thus, it can be classified as an excellent solution for the scenario at hand. The time complexity of this algorithm was found to be $O(n^2)$. Lastly, this method makes use of perceptual rules to decomposed the shape and will thus scores an excellent.

To conclude this subsection, it can be seen that this method shows a little more promise then the previous mentioned method, and has room for improvement. It also agrees to the type of method that we are looking for, and produces accurate results. In the next subsection, the Convex Shape Decomposition method will be discussed.

Convex shape decomposition

In Liu *et al.* [15] they propose convex shape decomposition (CSD). In this method, the approximate optimal solution is found by minimizing the total cost of decomposition under certain concavity constraints. Their method is based on the Morse theory and multiple Morse functions. Furthermore, they use integer linear programming to optimize the solution.

In this method, the first step is to compute multiple Morse functions. This is demonstrated in figure 2.25 (b). For each Morse function that has been calculated, the Reeb graph, shown in figure 2.25 (c), is constructed, as well as mutex pairs. This is then followed by obtaining a possible cut set, shown in cyan in figure 2.25(d). Out of the cut set the cuts which satisfy the mutex pairs are identified - in this case the pink points $-p_1$ to p_3 in figure 2.25(d). After that, a linear programming problem is solved to obtain the optimal points to cut the object at. The final cut is then made with a distance e from the optimal points to ensure that there is more than one part after decomposition. This is demonstrated in figure 2.25(e).

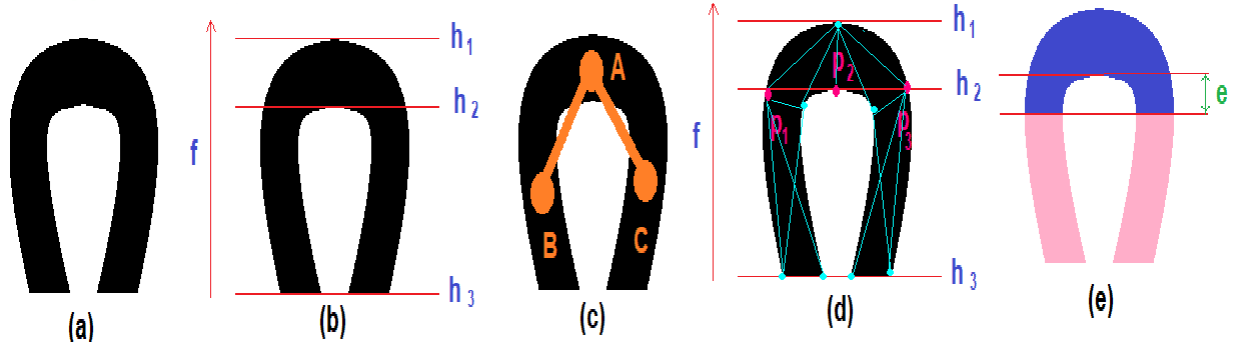


Figure 2.25: Some of the decomposition steps followed by the CSD method [15].

This method agrees with 45% of the cuts defined by the human perception results. This is a poor result and indicates that this method does not very accurately represent the human perception system. Furthermore, the number of parts of this method shows an average standard deviation of 1.6. This shows that on average there is at least one, if not two deviations from the average number of parts of human perception. This method will also fall under the very good category. In terms of invariance, it was found that this method is only translation and scale invariant [12]. This will result in the method falling under the good category. The time complexity of this method is $O(tn \log(n))$ [12], where n represents the basic number of elements in an object, and t is the amount of Morse functions, which in their case was chosen to be 16. Lastly, this method makes use of psychological type of shape decomposition [12]. Thus this method will score an excellent for this criterion.

To conclude this subsection, it is found that this method in terms of accuracy does not outperform the above mentioned methods, and produces a greater standard deviation than the above mentioned methods- indicating that this method will not be considered a good solution in this dissertation.

Approximate convex shape decomposition

In [16], Lien and Amato proposed an approximate convex decomposition method (ACD). Their algorithm is designed to be more efficient since strictly convex decomposition produces a lot of unnecessary parts, and takes a longer time to decompose. Furthermore, their algorithm makes use of iteratively removing the most significant non-convex feature. This then produces a hierarchical representation that provides a series of increasingly convex decompositions.

In their method, they make use of a concavity threshold value to determine which cuts to keep and which not to. Unlike the other shape decomposition methods, this method looks at the convex hull of the complex part, and iteratively decompose the shape according to the convexity threshold. This concept is depicted in figure 2.26 (a). As an example, figure 2.26 (b) is shown as the stating complex shape. The first step is to check the convex hull - shown with the orange lines. The next step is do determine where the best place is to cut the shape - which generally picks the minima points, combine with the shortest cuts - shown in cyan. This part will then be decomposed to produce P_1 and P_2 . Both these parts are then send through the same process, in which case P_1 meets the threshold value and is left as is, while part P_2 is still convex and the process continues

until all parts meet the convexity threshold. This is demonstrated in figure 2.26 (c) and (d).

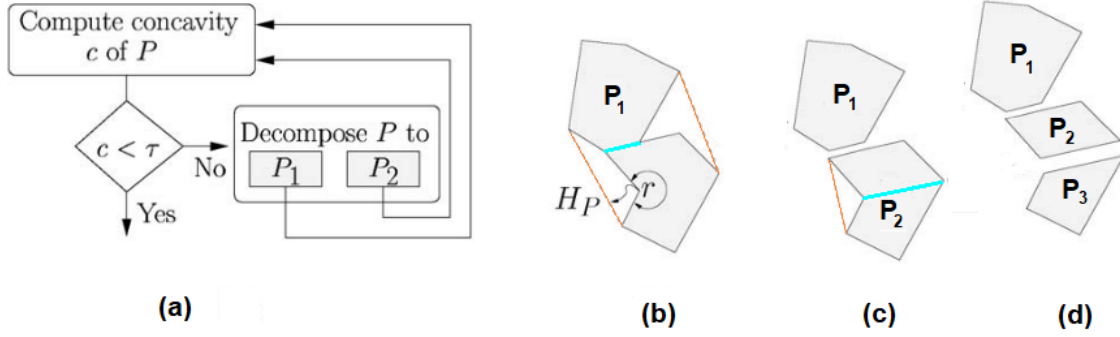


Figure 2.26: Some of the decomposition results of Lien *et al.* (ACD) [16].

For this method, about 50% of the cuts produced agree with that of human perception. This gives this method an average score. When looking at the number of parts, it was found that this method deviates an average of 2.9. After looking through some of the results, it has been found that this method is only translation and scale invariant [16] thus scoring this method a 'good' for invariance. The time complexity of this method is described as $O(nr)$ [16]. Here n represents the number of vertices, and r the number of notches. Lastly, this method also makes use of perceptual rules [16] to solve the problem and thus scores this method an excellent.

In conclusion, it can be seen that this method produces fast results, but not necessarily accurate results, as it produces more parts, as well as only 50% accuracy. Comparing this results to the other method, this method is not performing as well as the others. The final decision will be made in the discussion at the end of this section.

α shape decomposition

The last method considered is the α -decomposition (AD) method. Lu *et al.* proposed an α -decomposition (AD) of polygons [11] in their paper. Here they use an α parameter which is the diameter of a circle that is convolved with the input polygon.

In their method, as mentioned before, α is the diameter of a circle that is convolved with the input polygon. As can be seen in figure 2.27 (b), the larger the size of α , the larger the surrounding boundary - the yellow area around the object. The first step of their algorithm is to identify the pocket minima, shown as the large red points in figure 2.27 (b). This is obtained by convolution as mentioned earlier. The next step will be to identify possible cuts - this is done by using Delaunay Triangulation(DT). The resultant cuts are then placed into a vector and the optimal solution is found by making use of linear programming this is shown in figure 2.27 (c).

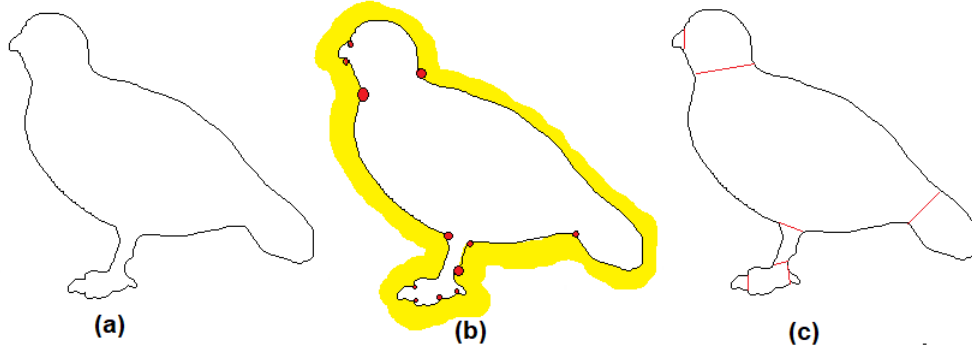


Figure 2.27: *Steps summarizing the AD algorithm [1].*

In their method, 62% of the cuts agree with that of the human perception cuts. This method will then receive a good score. In terms of the number of cuts, this method also produces a standard deviation of 1.1. Thus, this method is quite accurate in terms of number of parts, and will thus score a very good for a criterion. It was found that this method is invariant to translation, rotation, size, noise and deviation [1]. This thus scores this method an excellent. The time complexity tends to be $O(1)$ [1], which means that the algorithm is not dependent on the size of the input. This does not however mean that the time it takes is constant for all of the shapes. Lastly, the type of shape decomposition method that this method falls under is the geometry type of decomposition [1]. Thus scoring this method a poor as this type of decomposition is not what we are looking for.

In conclusion to this subsection, it can be seen that this method shows promise, with the largest negative criteria being that the time complexity is rather small and the shape decomposition type is not the favourable one. In the next section, the final comparison between the different methods will be summarised, as well as the time complexities will be given scores based on the other methods used.

2.2.3 Choosing the method to improve on

In this section the different methods will be discussed and compared to each other. This will be done with help from figure 2.28. The final results will then be summarised in table form, and according to the criteria mentioned in tables 2.2 and 2.3.

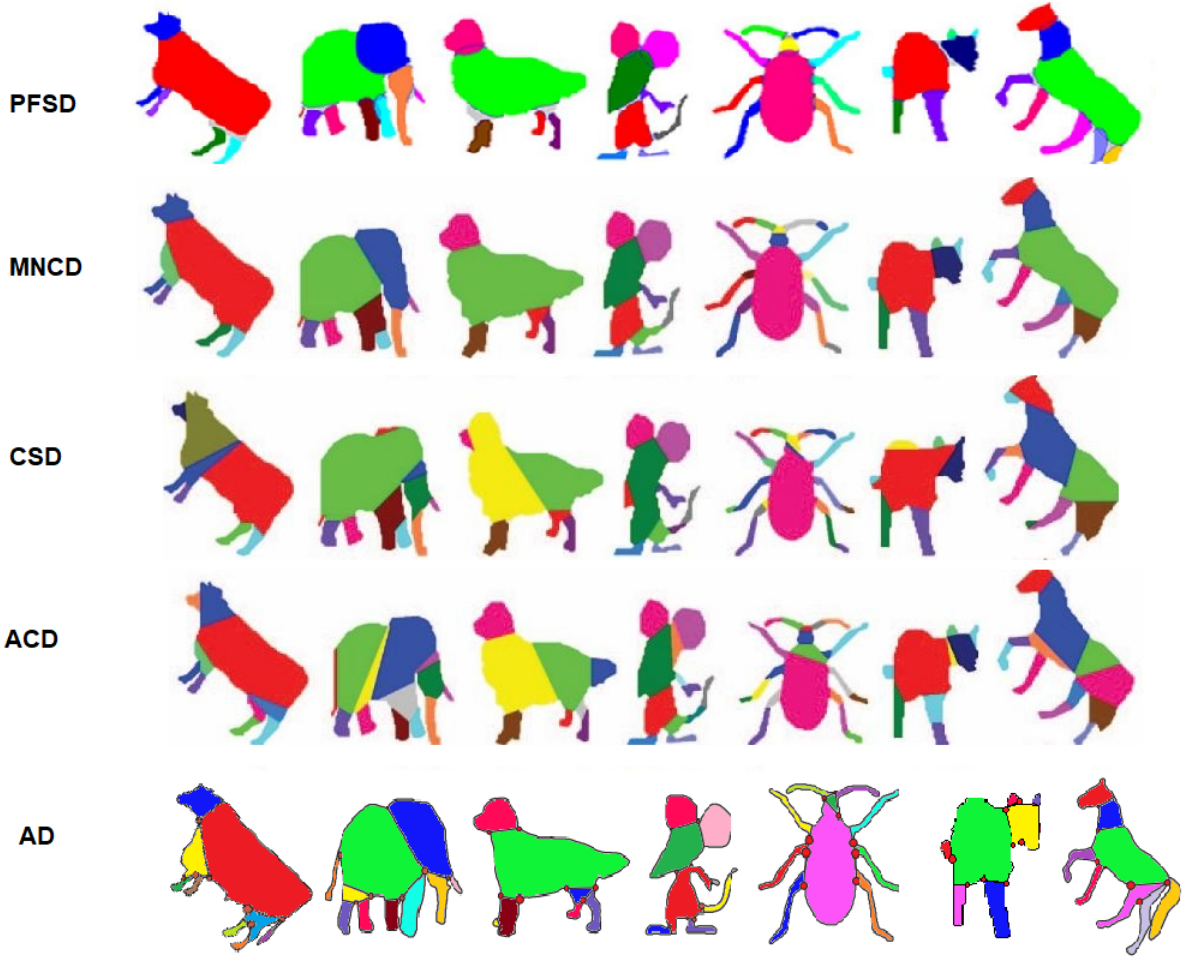


Figure 2.28: The decomposition results of the above mentioned algorithms [1].

When looking at the decomposition results, it can be seen that the algorithm that best relates to human perception is MNCD, followed by PFSD, then ACD, AD and lastly CSD. When it comes to the number of parts, one can see that the PFSD deviates the least from human perception, which is then followed by MNCD and AD, then CSD, and lastly ACD - which can also be seen by the amount of parts produced.

A summary can be given in a form of a table. This table 2.4 is shown below. All of the data and the calculations are can be found in Appendix B.

Table 2.4: Table showing a summary of all the decomposition methods.

Measure	Poor	Average	Good	Very Good	Excellent
Meaningfulness (%)	<50	51-60	61-70	71-75	>76
Number of Parts (STD)	>4.0	3.1-4.0	2.1-3.0	1.1-2.0	0-1.0
Invariance	0	1	2	3	4
Shape Decomposition Type	G		P&G		P

Replacing the scores with the values of table 2.3 and adding them up the result is shown in table 2.5 below.

Table 2.5: Table showing a summary of all the decomposition methods.

Method Values					
	Wang (PFSD)	Ren (MNCD)	Liu (CSD)	Lien (ACD)	Lu (AD)
Meaningful	3	4	0	3	2
Number of Parts	4	3	2	1	3
Invariance	4	4	2	2	4
Type	2	4	4	4	0
Score	13	15	8	10	9

Thus, as the scores indicate the method with the most possible improvement in terms of the different criteria is the MNCD method. This is then followed by PFSD, ACD, AD and CSD. This does however not indicate the CSD for example is the worst solution, it is simply a rank of the solution where the most improvement can be made on a method that is ranked the highest. It is however important to know that looking at the time complexities of this method, it is the highest of the considered method, and thus show room from improvement.

In conclusion following the criteria described above and the given scores as indicated in this and the previous sections, the final decision is to improve on the MNCD method.

2.3 Conclusion

To conclude this chapter, a summary will be given on what was done in this chapter, as well as what is to come in the next chapters.

In this chapter background research was done, as well as a literature study. The background study was aimed in aiding with the understanding of some of the terminology, as well as some of

the concepts used to recognise objects. This was based on the problem provided in chapter 1.

It was found that using shape decomposition will be a great solution to the problem. This was then discussed and the different type of shape decompositions were identified. Here psychological shape decomposition was a better solution, as it has a solid foundation and is easy to implement.

This was then followed by an literature review, where the very important question came up: *What is considered a good solution?*. In order to answer this question, we based our study on human perception. In order to compare method to human perception, a questionnaire was set-up and the results recorded. This then lead to the discussion of the quality of a solution, and which criteria will lead to a good solution.

After doing a literature reviews on several different methods, which include the PFSD, MNCD, CSD, ACD and AD, the criteria mentioned previously was applied. The method found with the most room for improvement, in terms of time complexity, and with high accuracy, in terms of human perception, was the MNCD method.

In chapter 3, this method will be discussed in more detail. This will include an in-depth study of the method in order to identify the areas where improvements can be made, and areas that don't need improvement. It is also important to keep the problem in mind throughout the in-depth study and how the improvements will effect the result. This will then be followed by chapter 4 where the detailed implementation will be discussed in more detail. Chapter 5 will show the results of the new implementation, whilst chapter 6 will conclude this dissertation.

Chapter 3

The MNCD method

3.1 Introduction

There exists a wide variety of ways to decompose a shape. Each method has its advantages as well as its disadvantages, and can be applied in many different fields. This can be seen by the literature study done in chapter 2. In the previous chapter it was found that the method showing the best results in terms of human perception and minimal number of parts is the Minimum-Near Convex Decomposition Method (MNCD) done by Ren, Yaun and Liu [2]. In their work the shape is decomposed into a minimum number of near-convex parts. The reason for choosing this method is that the cuts produced are the most meaningful in terms of human perception and at the same time it produces a minimal number of parts. Thus improving the time this method takes to decompose a shape and keeping the accuracy in terms of human perception and number of parts, this method will produce fast and accurate results.

In this chapter, the chosen method will be discussed in more detail. Before that is done, a background study will be done first in order to grasp a better understanding of some of the concepts used to produce a solution in the MNCD method. This chapter will then be concluded by identifying the areas where possible improvements can be brought to light and the changes will be discussed in the next chapter.

3.2 Background

Some of the concepts that will be discussed include mutex pairs, Morse functions, Reeb graphs and Binary Integer Linear Programming (BILP). All of the aforementioned concepts are used in conjunction to produce the results of the MNCD method.

3.2.1 *Mutex Pairs*

The first concept that will be discussed is Mutex Pairs. This term is used to describe mutually exclusive pairs of some sort. That is if any two points, nodes or events cannot be true at the same time, they are called *mutually exclusive* or *disjoint* [60]. The term *mutex pair* can then be described as an abbreviation and can found in several fields of study including Artificial Intelligence [61, 62], Statistics [60] and Computer Vision [63].

In this case, *mutex pairs* refers to coordinate pairs on an image. As mentioned in the previous chapter, these mutex pairs are used to describe the relationship between coordinate pairs on the contour of a shape [25].

Assume there is a line that joins any two coordinate pairs $p(p_x; p_y)$ and $q(q_x; q_y)$ and these coordinate pairs lie inside or on top of the contour of a shape. If any point on the line joining these two coordinate pairs lies outside the contour of the shape, or intersects with it, the coordinate pair p and q are known as a *mutex pair* written as $(p; q)$.

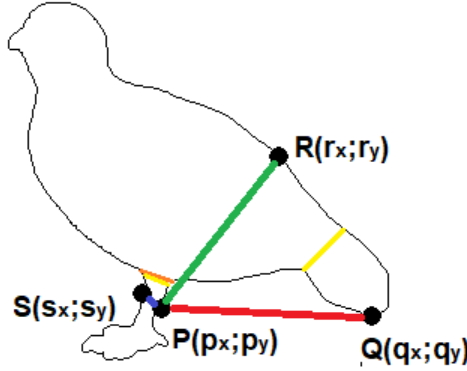


Figure 3.1: Picture demonstrating the concepts of mutex pairs.

When looking at figure 3.1, it can be seen that the coordinate pairs $P(p_x; p_y)$ and $Q(q_x; q_y)$ are mutually exclusive as the line joining them is completely out of the contour of the shape. Coordinate pairs $P(p_x; p_y)$ and $R(r_x; r_y)$ are also mutually exclusive as the line intersects with the contour. The coordinate pairs $P(p_x; p_y)$ and $S(s_x; s_y)$ however is not mutually exclusive as the line connecting them is completely inside the contour of the shape. Thus, the mutex pairs in the figure will be $(P; Q)$ and $(P; R)$.

When mutually exclusive pairs exist it can be concluded that the shape at hand is non-convex and it can possibly be divided into more parts. Thus, in determining the mutex pairs, these pairs can be used to determine whether or not the shape can be divided into more parts, and will also give an indication as to which parts can be divided. That is, looking at the picture shown in figure 3.1, the bird can most likely be cut along the lines of PQ or PR than PS , where the yellow lines indicate possible cuts that can separate PQ and the orange line a cut that can separate PR .

Thus, mutex pairs can be used as a great tool in shape decomposition, as it is easy to determine, and a lot of information can be taken out of the mutex pairs. Now that the mutex pair concept is explained and the reason for the use thereof is given, it can be seen why this tool is used and where it will be applicable. The next topic to be discussed is Morse Functions and Reeb graphs.

3.2.2 Morse Theory

The second concept that will be explained is *Morse Theory*. Before diving deeper, the concept of a manifold needs to be discussed first in order to better understand the Morse theory later. A manifold can be defined as a topological space that locally corresponds to the Euclidean space near a point [64]. In one dimension, a manifold can include lines and circles while in two dimensions a

manifold, also called a surface, can include a plane, a torus or a sphere. In this dissertation, only the one dimensional manifolds will be considered as our work is only limited to one dimensional manifolds. Now that a manifold is defined, let us look at the Morse Theory.

The *Morse Theory* has two basic operations for a manifold. The first operation includes the construction of a Morse function, while the second operation includes the reduction of the connected components of the level sets f^{-1} to points - better known as creating the Reeb graph.

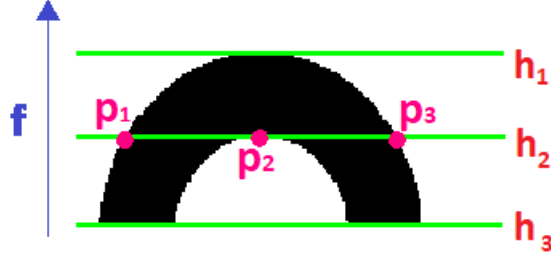


Figure 3.2: Picture demonstrating the Morse function of a half circle shown as the height function f in blue.

A *Morse function* can be defined as a projection from a higher dimension to a one dimensional manifold. Formally, the Morse function can be defined as follows [65]:

Definition 3.1. Let M be a manifold with or without boundary. We are interested in smooth maps $f : M \rightarrow \mathbb{R}$, and for simplicity such maps are known as *Morse functions*, which are defined by the following conditions:

1. all critical points of f are non-degenerate and lie in the interior of M ;
2. all critical points of f restricted to the boundary of M are non-degenerate; and
3. $f(x) \neq f(y)$ for all critical points $x \neq y$ of f and its restriction of the boundary.

In order to construct a Morse function, f , for each point p in an object $f(p)$ is the height of the point p . Thus simply put, the *height function* of a point p , is considered to be the Morse function of that object. This concept is illustrated in figure 3.2, where the Morse function f is shown next to the object, with the height points indicated at different levels.

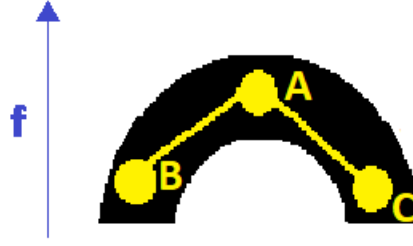


Figure 3.3: Picture demonstrating the Reeb graph of a half circle shown as the changes in the number of connected components in the function f^{-1} in blue.

A formal definition of the *Reeb graph* is given below [66]:

Definition 3.2. Let $f : X \rightarrow R$ be a continuous function and call a component of a level set, a contour. Two points x and $y \in X$ are equivalent if they belong to the same component of $f^{-1}(p)$ with $p = f(x) = f(y)$. The *Reeb graph* of f , denoted as $R(f) = X$, is the quotient space defined by this equivalent relation.

The *Reeb graph* can be determined by observing the changes in the number of connected parts in the function f^{-1} . As can be seen in figure 3.3, the Reeb graph is shown as the three yellow nodes. The number of critical points in the function f - the Morse function - correspond to the number of nodes on a Reeb graph. Furthermore, the Morse function used together with the Reeb graph can be used to determine the number of parts to decompose a shape into [17, 67, 68].

To conclude this section, the Morse function as well as the Reeb graph was defined and explained. It can be seen that this functions can be advantageous to shape decomposition application when the Reeb graph and the Morse functions are used in conjunction.

The next topic to be discussed is Binary Integer Linear Programming (BILP). This will be discussed as BILP will be used to solve the problem at hand and to produce the optimal solution, where the minimal number of cuts will be selected by solving a linear programming problem.

3.2.3 Binary Integer Linear Programming (BILP)

The next topic to discuss is *Binary Linear Integer Programming (BILP)*. *Linear programming* can formally be defined as a technique used to optimize a linear objective function that is subject to a set of linear equality and inequality constraints [69].

This technique can be used to determine the smallest number of optimal cuts to decompose a shape. The advantages of using this technique is the solid foundation thereof, and the fact that a large number of 'possible' cuts can be given as an input and the output will either be a one which will indicate a cut that needs to be kept, or a zero, which means the cut can be discarded.

The use of linear programming is important in many applications and was first used in 1937 by Leonid Kantorovich [69]. He developed a method to plan the way that the expenditures and returns are used in order to reduce the cost of running an army and increase the losses incurred by the enemy during the World War II. This lead to the discovery of BILP in 1984.

In general, an *integer linear programming* problem is a mathematical optimization problem where some or all of the variables are classified to be integers [70]. Thus, as the name suggests,

Binary Integer Linear Programming is a mathematical problem, where some or all of the variables are classified to be zeros or ones - thus implying that the variable either exists or not [69]. One of the constraints that this linear programming method holds is that the solutions cannot be continuous in nature, and hence additional measures are to be taken to determine the optimal solution [69].

This problem can generally be written as follows in matrix notation:

$$\begin{aligned}
& \textbf{maximize } C^T \mathbf{x} \\
& \text{subject to : } A\mathbf{x} \leq B \\
& \mathbf{x} \leq 0 \\
& \mathbf{x}_i \in \{0, 1\} \text{ for all } i = 1, 2, \dots, n.
\end{aligned} \tag{3.1}$$

Thus, here the cost vector, C , can be seen as a $n \times 1$ vector, the coefficient matrix, A is a $m \times n$ matrix, the requirements vector, B is a $m \times 1$ vector and the vector of unknowns, $\mathbf{x}$, is a $n \times 1$ vector [69].

An example of where BILP was used in shape decomposition to obtain an optimal solution can be found in [2]. Here the cost vector was represented by the a cost function, $\mathbf{w}$, which stores the cost of each cut, the coefficient matrix was represented by a convexity constraint, while the requirements vector was represented by a non-overlapping constraint. Solving this linear problem will produce a vector with the optimal cuts as the ones and the rest zeros.

These linear problems can sometimes become very difficult to solve due to a large number of optimization variables and constraints [71]. Due to this reason a few standard discrete optimization techniques have been developed, which include CPLEX, Lingo or integer relaxation techniques to name a few [2]. In their work, Ren *et al.* [2] made use of CPLEX to solve their linear integer programming problem.

To conclude the background study, a study was done on *Mutex pairs*, *Morse functions*, *Reeb graphs* and *Binary Integer Linear Programming*. This was done on some of the unfamiliar concepts to be able to discuss the Minimum Near-Convex Decomposition (MNCD) more depth. The next section the MNCD method will be discussed.

3.3 The Minimum Near-Convex Decomposition method

In this section the decomposition method used by the MNCD algorithm will be discussed. This is done in order to better understand their method and in order to identity areas where possible improvements can be made on it. In the next chapter the possible improvements will be discussed as to form the new algorithm to solve the problem at hand.

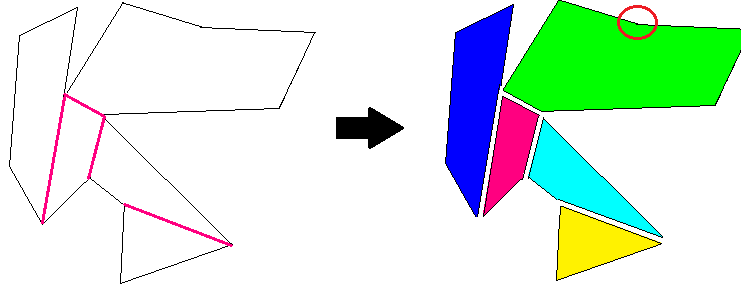


Figure 3.4: Picture demonstrating that near-convex decomposition does not decompose a complex shape into strict convex parts. As can be seen in the decomposed picture on the right, the red circle indicates a concave vertex, thus the green part is not convex, but near-convex.

Each decomposed part in a near-convex decomposition is not always found to be convex. This concept is illustrated in figure 3.4. A parameter, ψ , can be specified that is a convexity threshold of the broken down parts. *Convexity threshold* can be defined as the maximum convexity allowed on a part, before it has to be decomposed. It is a simple rule to allow near-convex shapes to be the result of decomposition, as strictly convex parts results in many unnecessary decomposed parts [2].

A formal definition to decompose a shape S into a ψ -near-convex decomposition can be given as [2]: for some decomposition, $D_\psi(S)$, containing only ψ -near-convex parts and contains no overlapping parts, can be written as:

$$D_\psi(S) = \{P_i | \cup_i P_i = S, \forall_{i \neq j} P_i \cap P_j = \emptyset, \text{concave}(P_i) \leq \psi\} \quad (3.2)$$

where P_i denotes the decomposed part. Thus, to explain this equation better, figure 3.5 is used. The shape at hand is called S , which in this case is the bird. The decomposed shape is a function of S called $D(S)$ shown to right. The decomposed shape exists out of a number of parts, denoted by P_i , where i is any number from 1 up. In figure 3.5, i will be 1 – 5, denoted as $i = 1, 2, \dots, 5$. Next we look at the rules set out in equation 3.2. The first rule is $P_i | \cup_i P_i = S$, which reads as every part of the decomposition belongs to the original shape, thus after decomposition the shape can be put back together again without any additional parts added. Thus, when looking at figure 3.5, when all the parts of the decomposed bird is put back, it will still produce the original bird.

The next rule that will be looked at is $\forall_{i \neq j} P_i \cap P_j = \emptyset$ which states that for all of the parts, the overlapping collection must be empty - that is, there is no overlapping parts. When looking closely to the right of figure 3.5, the parts are separated by cuts, and none of the end parts overlap with each other.

The last rule is $\text{concave}(P_i) \leq \psi$. With this rule, it states that the concavity of each part can be less or equal to a certain threshold value, ψ . That is, each part does not have to be strictly convex, but that if the concavity falls under a threshold, usually specified by the user, the parts can be accepted. This concept is demonstrated on the right in figure 3.5 by the red circles showing the vertices that are not convex, but concave. The smaller the threshold value the more parts will be created.

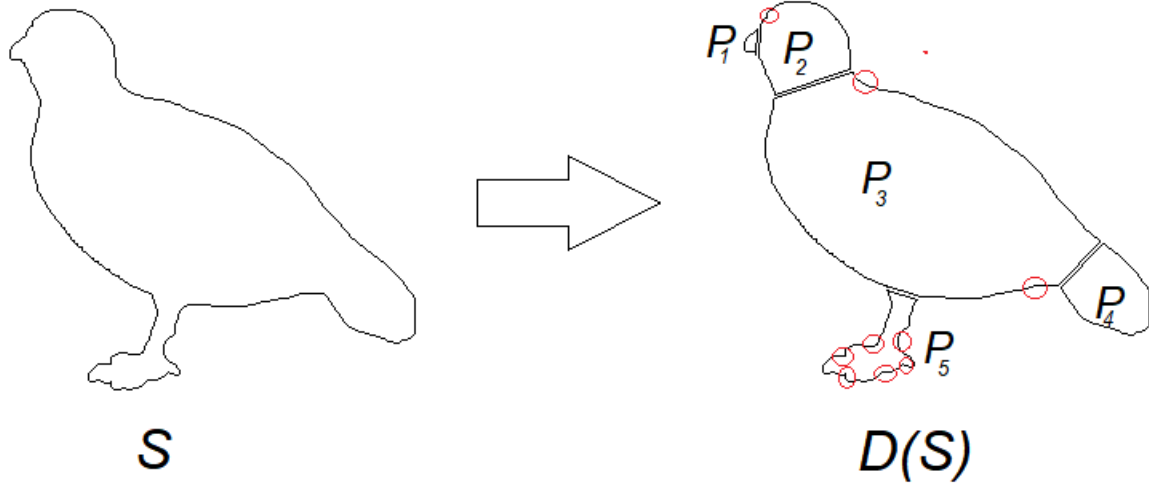


Figure 3.5: Picture demonstrating the definition of a decomposed shape.

As can be seen in (3.2), there exists two constraints. The first is the *non-overlapping constraint* which is denoted by $\forall_{i \neq j} P_i \cap P_j = \emptyset$. The second is the *convexity constraint* which is denoted by $\text{concave}(P_i) \leq \psi$. The concavity of P_i can be described by $\text{concave}(P_i)$.

A group of cuts forms a part, P_i . A cut can be defined as follows: if for any two points that lie on the border, say q and p , and the line joining these two points are located inside the border, then the line qp is called a cut [25]. This is demonstrated in figure 3.6. The complete candidate cut set in shape S is denoted by $C(S)$.

As mentioned earlier in this chapter, the mutex pairs can be used in different applications. In this case, the mutex pairs are used in order to determine the concavity of a part. Formally defining this concept:

Definition 3.3. For any two points, v_1 and v_2 , on the contour of a shape, if the line connecting points v_1 and v_2 locates outside the contour or it intersects with the contour, then (v_1, v_2) is known as a **mutex pair** (mutually exclusive pair).

In order to measure $\text{concave}(P_i)$, the shape feature *mutex pairs* proposed in [25] will be used.

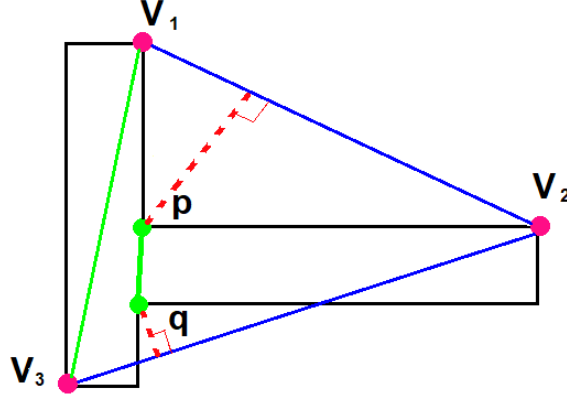


Figure 3.6: A figure demonstrating that any two vertices pq or v_1v_3 with a line connecting them that lies completely within the contour, is considered a cut (shown in green). The line v_1v_2 is found outside the concave part of the contour and line v_2, v_3 intersect with the contour, which forms mutex pairs (shown in blue).

As can be seen in figure 3.6, (v_1, v_2) and (v_2, v_3) are two mutex pairs. The maximal concavity of each mutex pair can be used to determine the concavity of each partition P_i :

$$\text{concave}(P_i) = \max_{(v_1, v_2) \in P_i} \{\text{concave}_m(v_1, v_2)\} \quad (3.3)$$

where (v_1, v_2) denotes the mutex pair in part P_i and $\text{concave}_m(v_1, v_2)$ is the concavity of the same mutex pair. Hence, we can obtain the concavity of a part, $\text{concave}(P_i)$, by measuring all of the mutex pairs' concavity, $\text{concave}_m(v_1, v_2)$, in P_i . Thus, $\text{concave}_m(v_1, v_2)$ is defined as:

Definition 3.4. The maximum orthogonal distance between a line (v_1v_2) , connecting a mutex pair, and the closest minima point on the contour is defined as the *concavity* of that mutex pair.

As can be seen in figure 3.6, the red dotted line indicates how the concavity can be measured and shows the concavity measures of $\text{concave}_m(v_1, v_2)$ and $\text{concave}_m(v_2, v_3)$. It can be seen that $\text{concave}_m(v_1, v_2) > \text{concave}_m(v_2, v_3)$, and thus the concavity of this part can be measure by $\text{concave}_m(v_1, v_2)$.

To ensure that the convexity constraint $\forall P_i, \text{concave}(P_i) \leq \psi$ is met, every single concavity of each mutex pairs in all of the partitions P_i , must not be greater than ψ . Thus to implement a convex decomposition that meets the ψ threshold of a complex shape it is necessary to classify all of the mutex pairs with concavity larger than ψ , into distinct parts so that the constraint $\text{concave}(P_i) \leq \psi$ is met. In figure 3.6, this concept is illustrated where the 'T'-shape is split into two parts by cut pq splits, and this also splits mutex pairs (v_1, v_2) and (v_2, v_3) . Therefore, the concavities of these two sections will not be affected by $\text{concave}_m(v_1, v_2)$ and $\text{concave}_m(v_2, v_3)$.

3.3.1 Overview

In this subsection, a short review of the MNCD method will be given. Here the algorithm used by Ren *et al.* [2] will be used as a guideline to explain the method in a more informal way. Each step will also be explained using pictures. It is important to note that references will be made to

equations that are only discussed after this section. These equation can be found in sections 3.3.2 and 3.3.7 and in the detailed discussion that is to follow.

Algorithm 1 MNCD (S, ψ)

Input: A shape, (S), and a concave threshold (ψ)

Output: ψ -MNCD of $S, \{P_i\}$,

- 1: compute the candidate cut set, $C(S)$;
 - 2: shrink the candidate cut set, $C(S)$;
 - 3: compute ψ -mutex set of $S \rightarrow M^\psi(S)$;
 - 4: **for each** mp_i in $M^\psi(S)$ **do**
 - 5: **for each** cut_j in $C(S)$ **do**
 - 6: check whether cut_j separates $mp_i \rightarrow a_{ij}$;
 - 7: **end for**
 - 8: **end for**
 - 9: **for each** cut_j in $C(S)$ **do**
 - 10: compute its cost $\rightarrow w_i$
 - 11: **for each** cut_j in $C(S)$ **do**
 - 12: check whether cut_i intersects with $cut_j \rightarrow b_{gz}$;
 - 13: **end for**
 - 14: **end for**
 - 15: obtain the optimized solution by solving in Eq. 3.10 $\rightarrow \{P_i\}$
-

In the first step of this algorithm, the candidate cut set, $C(S)$, is determined. It is important to note that this includes all of the possible cuts that can be made between any two points on the vertices. Figure 3.7 is used to demonstrate this. It is important to note that not all of the possible cuts have been shown, as this will completely colour the inside of the contour and the concept won't be effectively carried over.

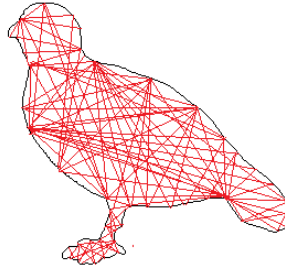


Figure 3.7: A figure demonstrating all of the possible cut in the cut set $C(S)$.

The next step will be to shrink the candidate cut set. This is done by eliminating all of the cuts where both of the vertices are convex, as these cuts cannot separate mutex pairs [2]. Thus, one goes through all of the cuts and shrink the cut set to cuts where at least one of the vertices are concave. This is demonstrated in figure 3.8, where it can be seen that there exists less cuts.

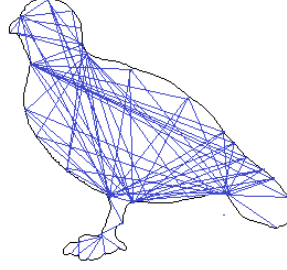


Figure 3.8: A figure demonstrating the cut set $C(S)$ that has now been shrunk to only contain cuts with at least one concave vertex.

The next step is to compute the ψ -mutex sets of the shape. This is done by using the Morse function as mentioned earlier to generate areas where possible mutex pairs can exist. As an example of what the result might look like, figure 3.9 is used. As can be seen, there are about six different areas that have been identified by this method and can be used to produce decomposition results.

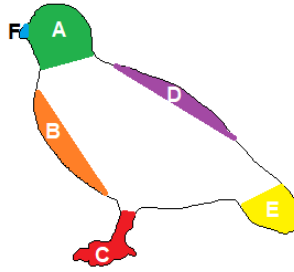


Figure 3.9: A figure demonstrating mutex areas identified by using multiple Morse functions.

The next step is to determine if a cut separates a mutex pair. By doing so a matrix, $\mathbf{A}$ is generated. This is done by checking each cut, $1, 2, \dots, j$ and if it separates a mutex pair mp_i a one is placed in the matrix, if not a zero. This then results in a matrix of size $i \times j$ filled with ones and zeros. Figure 3.10 demonstrates this concept, where $\mathbf{A}$ is the output of this step to be used later. It can also be seen that the cuts that remain are the possible cuts that separates mutex pairs and can be considered as containing the final cut set, which will be determined later.

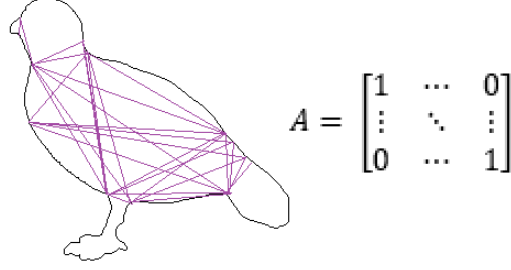


Figure 3.10: A figure demonstrating the cuts that separate the mutex pairs and the $\mathbf{A}$ matrix that shows which cuts separates which mutex pairs.

The next step is to go through the cut set and determine the cost of each cut. This is done by using equation 3.6, where the distance between the two endpoints of each cut, and the curvature of each endpoint is used to determine the cost of each cut. This is done in order to implement the minima and the short-cut perception rules. Thus, it can be seen that cuts that are short and/or cuts containing two concave endpoints are favoured. It is also important to note that the β value is used as a ratio to determine the relationship between the short-cut rule and the minima rule, and is set to one, meaning that each rule is considered equally important. Furthermore, the length of the vector w will be the same as the number of cuts, that is j .

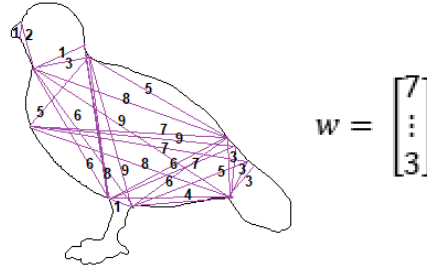


Figure 3.11: A figure showing the cost of each cut and the cost vector obtained w .

The next step is to determine whether any of the cuts intersect with each other. In order to do this, every cut from 1 to j must be checked against all of the other remaining cuts 1 to i , and if the cut j intersects with cut i a one is inserted into the intersection matrix $\mathbf{B}$, else a zero is inserted. This matrix, like matrix $\mathbf{A}$ mentioned above will also be used later. Figure 3.12 is used to demonstrate this concept - for each dot one cut intersects another cut, and a one is placed in the $\mathbf{B}$ matrix. It is important to note that the number of rows of matrix $\mathbf{B}$ will be equal to the number of intersections t . Thus if any row only contains zeros, it is to be removed. The number of columns will be the same size as the number of cuts j . Thus matrix $\mathbf{B}$ is of size $t \times j$.

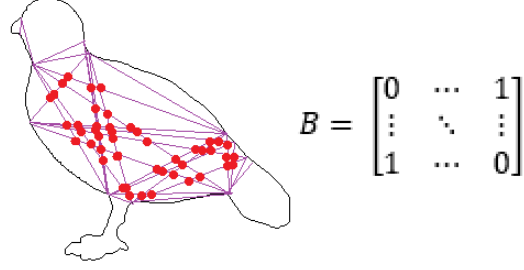


Figure 3.12: A figure showing the intersection of each cut and the intersection matrix $\mathbf{B}$ obtained.

The last step is then to solve for equation 3.10. This is done by using BILP. In order for the problem to be more simplified, matrix $\mathbf{D}$ is created, which is basically matrix $\mathbf{A}$ placed on top of matrix $\mathbf{B}$. Since both of these matrices has the same number of columns, j , the number of rows will become $i + t$. This is then set as one of the conditions in the BILP problem where matrix $\mathbf{D}$ should be less or equal to $\mathbf{u}$. Here, $\mathbf{u}$ is a vector containing -1 for i amount of rows and 1 for t amount of rows. This BILP problem is then solved by using a library called CPLEX produced by IBM. The result of this problem is then the optimized cuts, $\mathbf{x}$, which will be a vector containing ones, which indicate a cut that is to be kept, while a zeros indicates an invalid cut. Thus the final result is shown in figure 3.13.

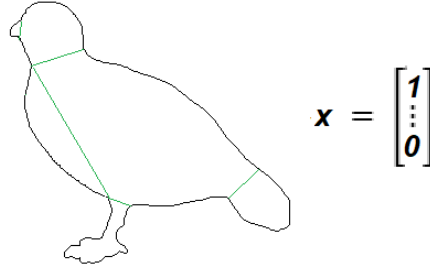


Figure 3.13: A figure showing the final selected cuts and the selection vector $\mathbf{x}$.

Now that a general overview of the MNCD algorithm is given, a more detailed discussion can be done.

3.3.2 Decomposing a shape

To obtain the minimal number of parts, while the visually naturalness is kept at a high degree, it is necessary to optimise the selection of cuts [25]. In a shape S , for a total of n potential cuts, that is $C(S) = \{cut_1, ..., cut_n\}$, a subset that consist out these cuts, written as $C'(S) \subseteq C(S)$, is known as the final decomposition. For each cut_i in $C(S)$, a binary variable x_i is assigned where:

$$x_i = \begin{cases} 0, & \text{if } cut_i \notin C'(S) \\ 1, & \text{if } cut_i \in C'(S) \end{cases} \quad (3.4)$$

Thus the cut selection or rejection from $C(S)$ is represented by a binary vector $x_{n \times 1} = (x_1, x_2, \dots, x_n)^T$. By using perception rules to minimize the total number of cuts, and with the two limitations in equation 3.2, the ψ -MNCD can be expressed as a BILP problem:

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_0 + \lambda \mathbf{w}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A} \mathbf{x} \geq \mathbf{1} \\ & \mathbf{B} \mathbf{x} \leq \mathbf{1} \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned} \tag{3.5}$$

here the number of chosen cuts in $C'(S)$ is represented by the zero-norm $\|\mathbf{x}\|_0$ of the vector $\mathbf{x}$. Furthermore in order to control the assortment of cuts the visually naturalness statement, $\lambda \geq 0$, is introduced to $\mathbf{w}^T \mathbf{x}$, so that the cuts that have higher visually naturalness are favoured. The variable λ will be discussed in a later subsection. Different from [25], a new explanation of a matrix is proposed for $\mathbf{B}$, as well as a Binary Integer Linear Programming (BILP) formulation in order to obtain an optimized number of cuts [2].

3.3.3 Visually naturalness condition ($\mathbf{w}^T \mathbf{x}$)

The *short-cut rule* [30] and the *minima rule* [51] are both made use of so that a degree of visually naturalness is enforced. In order to measure visually naturalness, a cost to each cut is given and a larger cost indicates a lower degree of visually naturalness:

$$w_{v_1 v_3} = \frac{dist|v_1 v_3|}{1 + \beta |min\{cur(v_1), 0\} + min\{cur(v_3), 0\}|} \tag{3.6}$$

where each term can be defined as follows:

- The vertices are represented by v_1 and v_3 .
- The normalized distance between v_1 and v_3 is $dist|v_1 v_3|$
- The normalized curvature of v_1 and v_3 are shown as $cur(v_1)$ and $cur(v_3)$
- $min\{cur(v_3), 0\}$ simply means that if the curvature is not less than zero, it must be made zero - thus choose the minimal value between the curvature or zero.
- The variable $\beta = 1$ represents the balance of two perception rules.

The normalized distance represents the *short-cut rule*: the longer the cut the higher the cost while the normalized curvature represents the *minima rule*: a cut that separates a shape at points with smaller negative curvature, has a higher cost. In order to balance these two perception rules, the variable β is introduced. This is set to $\beta = 1$ as both perception rules are of equal importance for natural decomposition.

The cost of n candidate cuts is denoted by $\mathbf{w}_{n \times 1} = (w_1, w_2, \dots, w_n)^T$. From Eq. 3.6 it can be seen that the cuts with larger lengths and vertices with smaller curvatures have greater costs. Therefore the cuts with lower visually naturalness are more probably neglected by minimizing $\mathbf{w}^T \mathbf{x}$.

3.3.4 Convexity constraint ($\mathbf{Ax} \geq \mathbf{1}$)

As mentioned earlier, to see to the convexity constraint $\forall(P_i), \text{concave}(P_i) \leq \psi$, the mutex pairs with concavities larger than ψ need to be divided into different parts. In order to achieve this the ψ -mutex set of S , $M^\psi(S)$ is first created. Here if the concavity of the mutex pair is greater than ψ , it forms part of this set. Assorted mutex pairs can be split¹ by a candidate cut, for example shown in figure 3.6 the cut pq can split a mutex pair. The mutex pairs that can be separated by a possible cut in $C(S)$, cut_i , forms a subset of $M^\psi(S)$, written as M'_i . Thus, $\{M'_i, i = 1, 2, \dots, n\}$ is obtained.

Definition 3.5. Let us assume that in the ψ -mutex set, $M^\psi(S) = \{mp_1, \dots, mp_m\}$, there are m mutex pairs. Then for each mutex pair in $M^\psi(S)$, mp_i , and for every cut that can separate it, the set $C'(S)$ must contain at least one cut. Thus, for every mp_i , this gives a restraint:

$$\sum_{j=1}^n a_{ij}x_j \geq 1, \text{ where } a_{ij} = \begin{cases} 1, & \text{if } mp_i \in M'_j \\ 0, & \text{if } mp_i \notin M'_j \end{cases} \quad (3.7)$$

Where $\mathbf{A}_{m \times n}$ denotes $(a_{ij} | i = 1, \dots, m; j = 1, \dots, n)$ and $\mathbf{1}_{m \times 1}$ denotes $(1, \dots, 1)^T$. Considering m , all of the mutex pairs in $M^\psi(S)$, one of the convexity constraints, $\mathbf{Ax} \geq \mathbf{1}$ in (3.5) is obtained.

3.3.5 Non overlapping constraint ($\mathbf{Bx} \leq \mathbf{1}$)

In the function $C(S)$, for any of the possible cuts it might happen that two cuts can intersect. In order to represent the intersection relationship, an intersection matrix $\mathbf{B}_{t \times n}$ will be defined.

Definition 3.6. Suppose cut_i and cut_j meet at the k^{th} intersection, then in matrix $\mathbf{B}$ the k^{th} row can be defined as:

$$b_{kw} = \begin{cases} 1, & \text{if } w = i \text{ or } j \\ 0, & \text{if otherwise} \end{cases} \quad (3.8)$$

To guarantee that the non-overlapping constraint, $\forall_{i \neq j} P_i \cap P_j = \emptyset$, in (3.2) is met, intersection of the selected cuts in $C'(S)$ may not take place. That is: $\forall_{g=1, \dots, t} \sum_{j=1}^n b_{gw}x_w \leq 1$. Thus the constraint of intersection can be written as: $\mathbf{Bx} \leq \mathbf{1}$, where $\mathbf{1}_{t \times 1} = (1, 1, \dots, 1)^T$.

3.3.6 Selection of variables

An important parameter to consider for visual naturalness in shape decomposition is λ . If the visual naturalness of the decomposition is not considered, that is $\lambda = 0$, (3.5) can be rewritten as:

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_0 \\ \text{s.t.} \quad & \mathbf{Ax} \geq \mathbf{1} \\ & \mathbf{Bx} \leq \mathbf{1} \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned} \quad (3.9)$$

Here the answer of $\mathbf{x}$ guarantees a minimum number of parts but it is not unique. Although with different end functions it can be proved that the formulation of Eq. 3.5 can obtain the same results as Eq. 3.9 if λ is selected suitably. This proof is done in [2]. Here the results show that Eq. 3.5 can separate a shape into the minimal number of parts when $0 \leq \lambda \leq \frac{1}{\sum_{i=1}^n w_i}$.

¹A mutex pair is *split* when a cut separates the parts of an object in such a way that the two vertices of the mutex pair are no longer on the same part [2].

3.3.7 Binary Integer Linear Programming (BILP)

In binary problems, each variable can only take on the value of 0 or 1. This may represent the selection or rejection of an option, the turning on or off of switches, a yes/no answer or many other situations [72].

An optimization of the expression created in (3.5) can efficiently be implemented using BILP. The binary nature of vector $\mathbf{x}$ can be used to rewrite (3.5) in a linear form: $\|\mathbf{x}\|_0 + \lambda \mathbf{w}^T \mathbf{x} = (\mathbf{1}^T + \lambda \mathbf{w}^T) \mathbf{x}$ where $\mathbf{1}_{n \times 1} = (1, \dots, 1)^T$.

A matrix can be defined as $\mathbf{D}_{(m+t) \times n}$ such that $\mathbf{D} = \begin{bmatrix} -\mathbf{A} \\ \mathbf{B} \end{bmatrix}$, and a vector $\mathbf{u}_{(m+t) \times 1}$ such that $\mathbf{u} = \begin{bmatrix} -\mathbf{1} \\ \mathbf{1} \end{bmatrix}$, where $-\mathbf{1}_{m \times 1} = (-1, -1, \dots, -1)^T$ and $\mathbf{1}_{t \times 1} = (1, 1, \dots, 1)^T$. Thus representing (3.5) as a BILP problem:

$$\begin{aligned} \min & (\mathbf{1}^T + \lambda \mathbf{w}^T) \mathbf{x} \\ \text{s.t. : } & \mathbf{D} \mathbf{x} \leq \mathbf{u} \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned} \tag{3.10}$$

By making use of discrete optimization techniques, for example integer relaxation, GLPK, CPLEX or Lingo, the above BILP problem can be solved. In their work Ren *et al.* [2] made use of CPLEX.

CPLEX is a library created by IBM that is used to solve linear programming problems. CPLEX implements optimizers based on the simplex algorithms as well as primal-dual logarithmic barrier and shifting algorithms to obtain the optimized solution to a problem [73]. Lingo is also a optimization library that has been produced by Lindo Systems and is described as a tool for utilizing, solving and analysing large linear and non-linear optimization problems [74]. There exist other libraries as well and each has its own advantages and disadvantages. The discussion thereof however, falls out of the scope of this dissertation.

3.4 Conclusion

In conclusion to this chapter a background study was done on the different topics that are used in the MNCD algorithm. This included discussions of mutex pairs, Morse functions, Reeb graphs and BILP. This was followed by a discussion of the Minimum Near-Convex Decomposition (MNCD) method, which started with an introduction to the MNCD method, followed by an overview and lastly the different topics of the MNCD method was discussed. In the next chapter the improvements will be discussed where the new approach will be explained in more detail.

Chapter 4

Improved implementation

4.1 Introduction

Up to this point, a scenario was mentioned where a bird needs to be quickly and accurately recognised. Accuracy here refers to how close the algorithm can recognize a shape to human perception. It was found that from a collection of shape descriptors, that shape decomposition is a good way of recognising an object. Furthermore different shape decomposition methods were looked at and it was found that the MNCD method most accurately relates to human perception. This method was discussed in depth in the previous chapter, and the different improvements that can possibly be implemented will be discussed in this chapter.

This chapter will start with an overview of the new improved algorithm. After that the possible improvements that can be made to the MNCD method will be discussed. Before the improvements are to be implemented, a short background study will be done on the different unfamiliar topics will be discussed. This chapter will start with a background study on Harris Corner Detection (HCD), Discrete Contour Evolution(DCE) and Delaunay Triangulation(DT). These concepts are implemented in the improved method, and a bit of a background study can help better understand the topics mentioned.

4.2 Algorithm overview

Before all the changes can be discussed, a overview will be given on the improved method. This will be done in the same way as 3.3.1, in order to show how the improved method differs from the MNCD method, and where they are similar. Here the improved algorithm based on the MNCD method is given, and a short summary of the improved method will be given as well as a figure explaining each process.

Algorithm 2 Algorithm for complex shape decomposition

Input: A complex shape (S), and concave threshold (ψ)

Output: Decomposed shape P_i into simpler convex shapes

- 1: Compute corners using Harris-Corner detection, $H(S)$
 - 2: Obtain DCE and optimal number of vertices $H(S) \rightarrow J(S)$
 - 3: Compute cut set using Delaunay triangulation, $C(S)$
 - 4: Compute ψ -mutex set of $H(S) \rightarrow M^\psi(S)$
 - 5: **for each** mp_i in $M(S)$ **do**
 - 6: **for each** cut_j in $C(S)$ **do**
 - 7: check whether cut_j separates $mp_i \rightarrow a_{ij}$;
 - 8: **end for**
 - 9: **end for**
 - 10: **for each** cut_j in $C(S)$ **do**
 - 11: obtain distance and curvature to determine cost $\rightarrow w_i$
 - 12: **end for**
 - 13: Solve the optimization problem in Eq. 3.9 $\rightarrow P_i$
 - 14: **for each** P_i in $P(S)$ **do**
 - 15: Fit simplest convex shape $\rightarrow CP_i$
 - 16: **end for**
-

For this algorithm, just like the MNCD method, a complex shape as well as a convexity threshold, ψ , must be given as input. To compare the improved method with that of the MNCD method, the same bird figure will be used. The first step of the algorithm will then be to determine a number of corners or vertices using the Harris Corner Detection algorithm. This gives the results shown in figure 4.1. As can be seen a lot of green vertices are produced, but this is seemingly less than using each vertex on the contour.

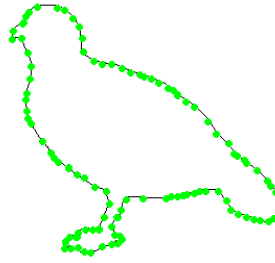


Figure 4.1: *First step of the algorithm: Obtain corners using HCD.*

The second step is to obtain the DCE and the optimal number of vertices. This is done as to be explained later in this chapter. This and the previous step is not in the MNCD method, and is introduced as a means of making the MNCD faster, but producing less vertices which will be used to obtain cuts, and will be discussed in the next step. Figure 4.2 shows the results of the DCE and the pink dots indicate the vertices that will optimally describe the shape without leaving out too much detail.

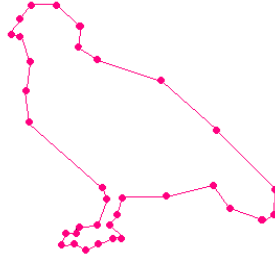


Figure 4.2: *Second step of the algorithm: Obtain vertices using DCE.*

The next step in this algorithm is to determine the cut set, $C(S)$. This step is the first step of the MNCD algorithm, and as can be seen some pre-processing has been done before we get to this step. Here the cut set is also determined by applying the Delaunay Triangulation algorithm to the vertices obtained in the previous step, other than that of the MNCD algorithm where the cuts were determined by drawing lines from every single point of the contour to every other point thereof. Figure 4.3 shows the result when the DT algorithm has been applied to the vertices. As can be seen there is a lot less cuts compared to that of figure 3.7, thus creating a reduced cut set and in turn reducing the amount of time it takes to produce results.

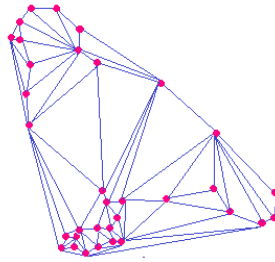


Figure 4.3: *Third step of the algorithm: Obtain cut set using DT.*

The fourth step in this algorithm is to compute the ψ -mutex set, or that is, the mutex pairs/areas. In contrast to the improved algorithm, the MNCD algorithm made use of multiple Morse functions, and then areas that are mutually exclusive to each other was determined. In this algorithm however, the vertices are used as mutex pairs, and no areas are defined. In order to determine which vertices are mutually exclusive, the line connecting these points must lie outside the contour, or cut through the contour. In order to determine the concavity of the line, ψ , a perpendicular line was drawn from the line connecting the two vertices to the closest contour line to it. The furthest perpendicular distance is then used as the concavity measure, and any lines that are less or equal to the threshold value of ψ will then be regarded as a mutex pair. This process is explained in more detail in the sections 3.2.1 and 4.4.2. Figure 4.4 shows all of the mutex pairs by making the lines connecting these vertices green.

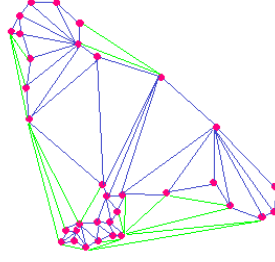


Figure 4.4: Fourth step of the algorithm: Determine ψ -mutex set.

The fifth step is to determine the concavity matrix, $\mathbf{A}$. As in the MNCD algorithm, this is determined by checking which cuts separates which mutex pairs. The method of obtaining this matrix is discussed later in this chapter. Figure 4.5 is used to demonstrate this step. The orange lines are to show the contour, and to indicate that no cuts will go through the contour, but would rather lie on top of the contour. These lines are removed out of the cut set as well as the mutex set as they don't fall into any of those categories.

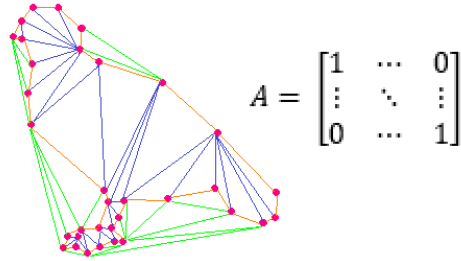


Figure 4.5: Fifth step of the algorithm: Determine convexity matrix $\mathbf{A}$.

The next step is to determine the cost function of all the cuts in the cut set. This step is also included in the MNCD algorithm. In order to determine the cost vector, $\mathbf{w}$, the length of each line as well as the curvature of each vertex has been calculated as mentioned later, and the cost function shown as equation 3.6 is used. Figure 4.6 is used to demonstrate this concept.

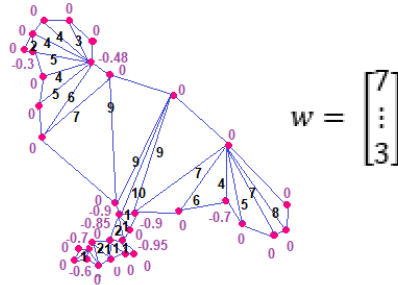


Figure 4.6: Sixth step of the algorithm: Determine cost vector $\mathbf{w}$.

The seventh step is to solve the BILP in equation 3.10. This equation, however changes slightly as there is no non-overlapping matrix, $\mathbf{B}$. Thus, $\mathbf{D}$ becomes $-\mathbf{A}$, and vector $\mathbf{u}$ becomes $-\mathbf{1}$. Now equation 3.10 can be solved using the above constraints to obtain the final cut set $\mathbf{x}$. Figure 4.7 shows the results of the BILP in vector form, as the final cuts selected in cyan on the picture of the bird.

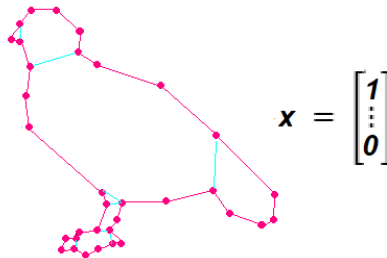


Figure 4.7: *Seventh step of the algorithm: Solve the BILP and obtain selected cuts vector $\mathbf{x}$.*

The last step is to simplify the cut parts into simpler shapes, as describe later in this chapter. Thus, the final figure 4.8 shows how the bird can be described by three triangles, one quadrilateral, one pentagon and four hexagons. Although the results might look a bit unnatural to the eye, this step is done to store the information in an easy and compact way. This is also done in order to identify shapes easier, as it is easier to compare a new object with example five pentagons and three hexagons to this object in which case will thus have no probability of being the same object.

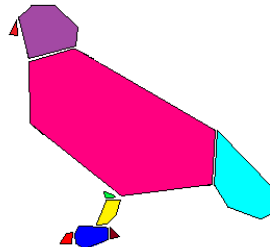


Figure 4.8: *Eighth step of the algorithm: Obtain primitive shapes.*

Thus, to conclude the overview of this algorithm, one can see that there are a few improvements that will be discussed in section 4.4. These improvements will be implemented at different stages of the MNCD algorithm and one can see where the improved algorithm is similar to the MNCD algorithm and where not.

4.3 Background

In this section a background study will be done on some of the concepts used to improve on the MNCD method. The areas that has been identified as problem areas include the following: the use of the all of the vertices when determining the possible cuts, the use of the Morse function

and no attempt was made to simplify the decomposed parts into primitive shapes. Of the above mentioned problems, the first two are improvements that can be made, while the last is a new implementation. Therefore, in this section the concepts that are going to be discussed include Harris Corner Detection(HCD) and Discrete Contour Evolution(DCE) - that will be used to decrease the number of vertices - and Delaunay Triangulation(DT) - that will be used to decrease the number of cuts as well as eliminating the non-overlapping matrix.

4.3.1 Harris Corner Detection

The aim of this section is to explain how corner detection is used to identify corners on the contour of a shape. Corners are known to be one of many interesting feature points in an image [19]. These features are in abundance in images where man-made objects are present and can quickly be identified in an image.

A classical approach to find corners in an image is to use the Harris feature detector. Harris looks at the average directional intensity change in a small window around a point of interest in order to define the detection of corners in an image. The average intensity change, when considering a displacement vector $(\Delta x, \Delta y)$, is given by [75]:

$$f(x, y) = \sum_{(x_k, y_k) \in W} (I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y))^2 \quad (4.1)$$

The summation is done over an outlined section around the regarded pixels. The average intensity change in all directions can then be calculated which leads to the definition of a corner as a point at which the average directional change is high in more than one direction. The Harris test can then be performed as follows: first the direction of maximal average intensity change is determined. Next the intensity in the perpendicular direction is determined. If this is also high then there is a corner.

This statement can be tested mathematically by using an approximation of (4.1) using the Taylor expansion. Without loss of generality, we will assume a grayscale 2-dimensional image is used. Let the image be represented by I . Consider taking an image sample over the area (x, y) and is shifted by $(\Delta x, \Delta y)$. The *sum of squared differences* (SSD) between these two samples, f , is given by Eq.4.1 where $I(x + \Delta x, y + \Delta y)$ can be approximated by a Taylor expansion.

Let I_x and I_y be partial derivatives of I , such that:

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y \quad (4.2)$$

This produces the approximation:

$$f(x, y) \approx \sum_{(x_k, y_k) \in W} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2 \quad (4.3)$$

Which is then rewritten in matrix form:

$$f(x, y) \approx [\Delta x \quad \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad (4.4)$$

where M is the structure tensor:

$$M = \sum_{(x_k, y_k) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x_k, y_k) \in W} I_x^2 & \sum_{(x_k, y_k) \in W} I_x I_y \\ \sum_{(x_k, y_k) \in W} I_x I_y & \sum_{(x_k, y_k) \in W} I_y^2 \end{bmatrix} \quad (4.5)$$

The Harris Corner Detection algorithm can be summed up in these five steps:

1. Convert image to a grayscale image
2. Calculate the spatial derivative
3. Constructing the variable matrix M
4. Calculate the Harris response
5. Convert the Harris response image back to grayscale

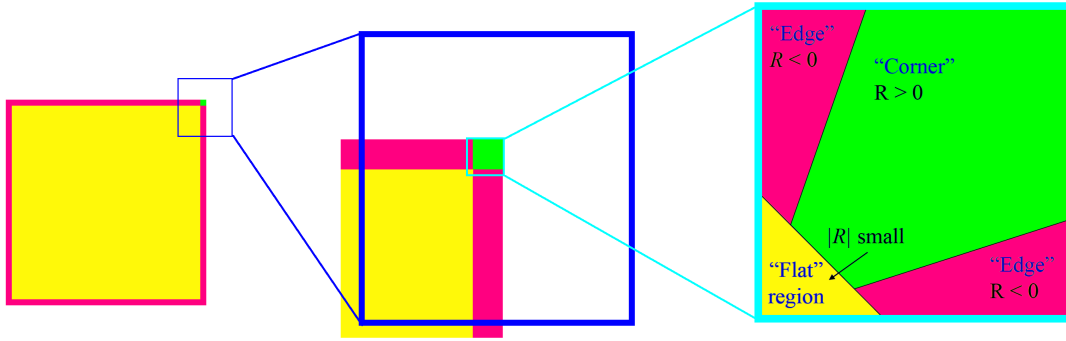


Figure 4.9: *Harris corner detection: a simple image to show the criteria required to be classified as a corner. When assuming that the rectangle is the derivative of the original image, and that it is in grayscale. The colours are inserted for it to makes sense. The green area shown high intensity, the pink shows lower intensities and yellow region with very low, or zero intensity [18].*

The first step is to convert the coloured image to a grayscale image. This is done to improve the processing speed of the corner detection. In the second step the spatial derivative is calculated using $I_x(x, y)$ and $I_y(x, y)$. Thereafter the structure tensor, M , can be set-up using the spatial derivatives. The fourth step includes using an approximation equation to calculate the smallest possible eigenvalue of the structure variable by using the following:

$$\lambda_{min} \approx \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} = \frac{\det(M)}{\text{trace}(M)} \quad (4.6)$$

where $\text{trace}(M) = m_{11} + m_{22}$ and λ_1 and λ_2 are the eigenvalues of M . Another generally used Harris response calculation is:

$$R = \det(M) - k(\text{trace}(M))^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (4.7)$$

where k is determined by trial and error and should be in the range: $k \in [0.04, 0.06]$.

Looking at this equation, it can be seen that the eigenvalues decide whether a region is a corner, an edge or flat. When $|R|$ is small, which happens when both λ_1 and λ_2 are small, the region is flat. When $R < 0$, which happens when $\lambda_1 \gg \lambda_2$ or vice versa, the region is an edge. Lastly when R is large, which happens when both λ_1 and λ_2 are large and $\lambda_1 \sim \lambda_2$, the region is a corner. This is depicted in figure 4.9.

The corner detection algorithm is used to obtain corners that will produce fewer points, and thus make the decomposition faster as a few vertices are better to use than every single vertex on the contour. One problem that is faced with this implementation, is that there exists a lot of different input variables that produce different results. This is difficult to implement as for each picture the optimal number of corners will be different, and the same variable values will produce excellent results for one image, but taken from just a slight different angle can produce completely different results.

Thus, implementing HCD will be advantageous up to a certain extent, then it will also be disadvantageous. In order to solve the problem of the parameter that have to be tuned manually, Discrete Contour Evolution was looked at. Implementing HCD and DCE together might produce faster and more accurate results. Next, some background information on the DCE method will be discussed.

4.3.2 Discrete Contour Evolution (DCE)

The aim of this section is to explain the concept of DCE. As mentioned in the previous section, HCD has many variables which makes it hard to obtain an optimal solution for many different scenarios. Therefore DCE can be used to obtain the most important vertices on the contour of an object. This algorithm is used as it is accurate and only contains one variable - the stopping criteria, which will be discussed next - instead of other corner detection algorithms which make use of more than one variable. DCE is also implemented to decrease the total number of cuts, and so the time complexity of the MNCD method.

Discrete Contour Evolution describes an iterative approach to reduce the contour of an object [76]. It can be viewed as an algorithm that removes redundant vertices and only keeps those vertices with the greatest influence on that contour. Therefore, this algorithm is a good alternative to corner detection algorithms when the variables for those algorithm are unknown. DCE is also of good use when the contours are often misrepresented due to segmentation errors and digitization noise [77]. Hence the use of DCE seems more applicable to our scenario than using different corner detection algorithms. The process of DCE will be explained using figure 4.10.

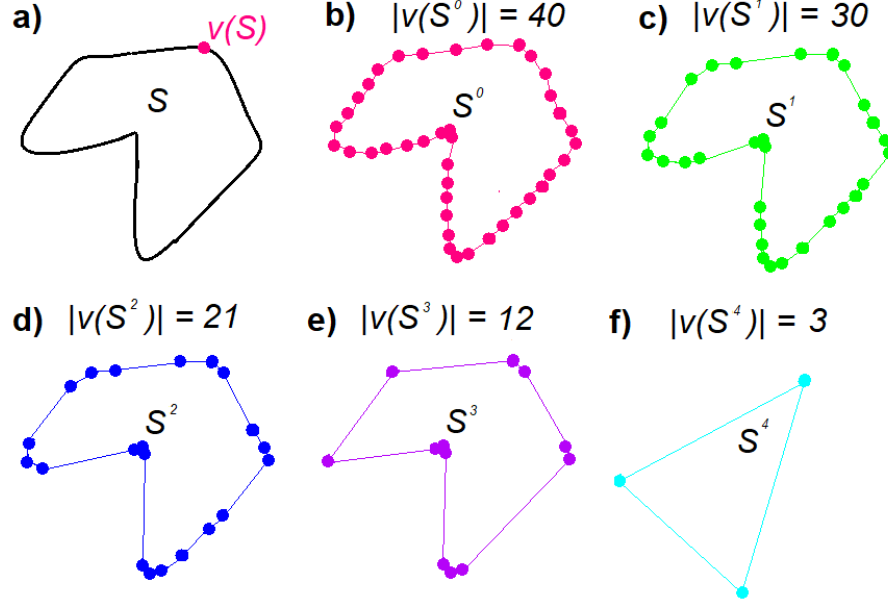


Figure 4.10: Process showing how the DCE process works.

As can be seen in figure 4.10, (a) is a visual representation of the above explained concept. In (b) the number of vertices are $|v(S^0)| = 40$. This then decreases as the relevance of each vertex is determined, and in (c) $|v(S^0)| = 30$. This process then iteratively continues until $|v(S^0)| = 3$, shown in (f), which is the general stopping criteria for a DCE algorithm.

As can be seen, this method reduces the number of vertices and only makes use of the vertices that have the greatest impact on the shape at hand to completely describe the shape. This technique was looked at in order to reduce the number of vertices, and thus the cut set, $C(S)$ of the MNCD method mentioned in the previous chapter. In doing this the time complexity is reduced as less vertices are used, and so less cuts to process and data to go through before getting to the final problem at hand.

For the rest of this section, the majority of the work will be referenced out of [77]. Consider S to be a closed complex polygon (that is, a polygon that is concave and has more than three vertices). A vertex of S will be denoted as $v(S)$. A sequence of polygons, $S = (S^0, \dots, S^m)$ are produced until $|v(S^m)| \leq 3$, with the *discrete contour evolution (DCE)* technique. Here $|\cdot|$ indicates the number of vertices. This is demonstrated in figure 4.10.

A relevance measure $K(v, S^i) \in \mathbb{R}_{\geq 0}$, is assigned to each vertex v in S^i . In general an evolution can be described by using the following simple idea: a polygon S^{i+1} is created for every step $i = 0, \dots, m - 1$ in the evolution after all of the minimal relevance vertices has been removed from S^i .

Before DCE can be defined, the following definition is given:

Definition 4.1. If we let the smallest measure of the relevance values be given by $K_{min}(S^i)$ for the vertices of S^i , then:

$$K_{min}(S^i) = \min\{K(u, S^i) : u \in v(S^i)\} \quad (4.8)$$

and then let the set $V_{min}(S^i)$ comprise the vertices whose relevance measures are negligible in S^i :

$$V_{min}(S^i) = \{u \in v(S^i) : K(u, S^i) = K_{min}(S^i)\} \quad (4.9)$$

for $i = 0, \dots, m - 1$.

Definition 4.2. For a relevance measure of K and any given polygon S , a *discrete contour evolution* can be defined as a process that produces a sequence of polygons $S = S^0, \dots, S^m$ such that [77]:

$$v(S^{i+1}) = \frac{v(S^i)}{v_{min}(S^i)} \quad (4.10)$$

where $|v(S^m)| \leq 3$.

After every step of evolution, the number of intersection points decrement by one at least, which will cause the evolution process to terminate when the number of vertices is three. This is because two vertices produces a line and one produces a point. Theee vertices is thus the minimum number of vertices required to make a polygon. This leads us to the following proposition:

Proposition 4.1. A convergence to a convex polygon will be reached by the DCE as there exist $0 \leq i \leq m$ in such a way that S^i is convex, and if $0 \leq i < m$, all of the polygons $S^{i+1}, \dots, S^m$ will be convex.

The relation between the geometric properties of the evolved polygons and the mathematical simplicity of this evolution approach is demonstrated by this proposition. The order in which the substitution takes place is the most important property. This is done using the K relevance measure given by:

$$K(\beta, l_1, l_2) = \frac{\beta l_1 l_2}{l_1 + l_2} \quad (4.11)$$

The primary concept that can be derived is: the greater the value of $K(\beta, l_1, l_2)$, the larger the contributing factor of the arc $\overline{vu} \cup \overline{vw}$ is to the polygon S^i . As demonstrated in figure 4.11, β is the turn angle at the common vertex v in S^i . Furthermore l_1 and l_2 are the lengths of $\overline{vu}$ and $\overline{vw}$. Both of these lengths are to be normalized to the total length of S . The contribution of each vertex v in S^i is represented by K .

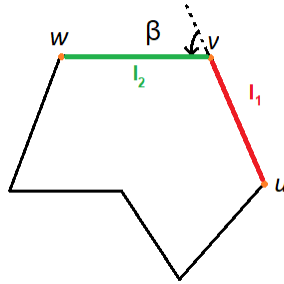


Figure 4.11: DCE: an image to demonstrate how the relevance value is determined.

It is important to note that this relevance measure is computed locally to S^i for every vertex, and that it is not a local property to the polygon S . Initially there are no control parameters required for this method, but it might be necessary for a stop parameter to be implemented. The reason for this is that we do not want every object reduced to a triangle as the iterations will continue until there is only 3 point left. A method of determining a stopping criterion will be discussed in section 4.4.1. The above concept is illustrated in figure 4.10.

The problem that is faced with this technique is that a stopping criteria must be specified by the user that might not always yield the optimal results. Another problem faced with this technique is that going through all of the vertices, removing some and then looping through all of the points again is a time consuming process. These problems will be addressed in section 4.4, when the implementation of the improvements are discussed. In the next subsection, Delaunay Triangulation(DT) will be discussed.

4.3.3 Delaunay Triangulation (DT)

The aim of this section is to explain the concept of DT. This algorithm was thought good to implement as it will avoid the chances of cuts overlapping, thus completely removing the need of the overlapping matrix and the non-overlapping constraint mentioned in the MNCD method. This will also decrease the complexity of the MNCD solution as there will be less cuts to consider, thus speeding up the process of decomposing an object. As an added advantage it also ensures that the cuts will be convex in nature. Thus, a short background study on this algorithm will be given, and the implementation thereof discussed in the next section.

Invented in 1934, Delaunay Triangulation (DT) is a technique that connects input points into groups of triangles in such a way that the minimum angle of all the angles in the triangulation is maximumly optimized [19]. When triangulating points, the DT tries to avoid long skinny triangles. Figure 4.12 shows that triangulation is done in such a way that any circle that is fit to three points at the vertices of any given triangle contains no other vertex. This is called the *circum-circle property*.

For effective computation, a far-away outer bounding triangle from which the algorithm starts is invented by the Delaunay algorithm. figure 4.12 (b) illustrates the outer invented triangle by dotted lines going out to its intersection points. 4.12 (c) shows some examples of circum-circle property including one of the circles linking two outer points of the real data to one of the vertices of the inverted external triangle.

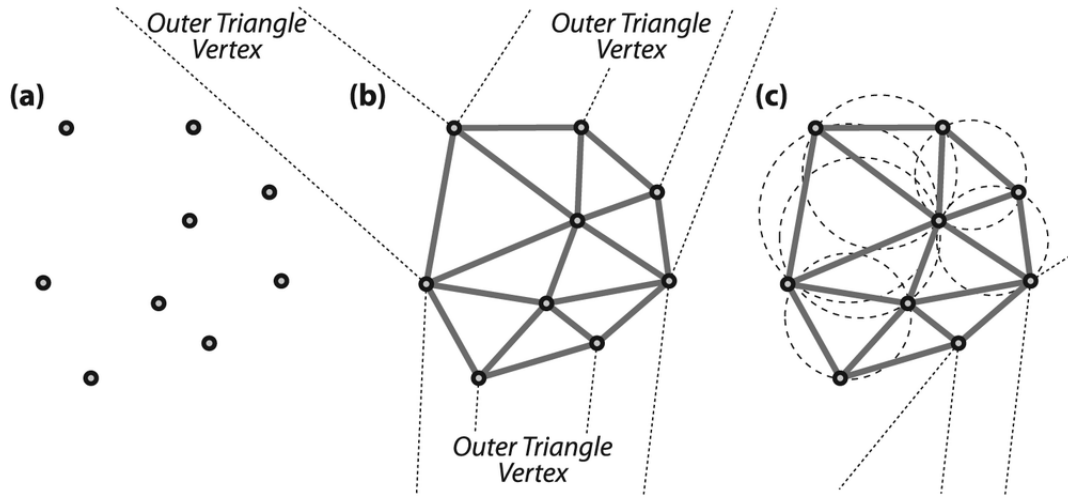


Figure 4.12: Image illustrating Delaunay triangulation where (a) is the input points drawn, (b) is the Delaunay triangulation of the input points and (c) showing an example of the circum-circle properties [19].

Many algorithms currently exists that can compute the DT of complex shapes. Several algorithms have difficult internal details but are very effective. The following summary shows the steps of the less complex algorithms:

1. Create the outer triangle and begin at one of its intersection points - this ensures a explicit external beginning point.
2. Add an inner point then remove those triangulations whose circum-circles contains that point.
3. Re-draw the triangles, ensuring to include the new point in the circum-circles of the just erased triangles.
4. Repeat steps two and three until there are no more points left to add.

The complexity order of this algorithm is $O(n^2)$ in the number of data points. The best algorithms are (on average) as low as as $O(n \log n)$ [78]. In computational geometry and mathematics $DT(P)$ can be described as the DT of a set of discrete points P in a plane.

Since this method has been used many times before, there exist some standard DT algorithms that can be used to determine the DT of a set of points, without having to do it from first principles. Thus, now that an understanding of some of the background concepts behind DT is given, the algorithm will be easier implemented and used.

4.4 Improved MNCD-method

In this section, the improvements made to the MNCD method will be discussed. This will be done in detail so that the changes are presented an understandable way. In this section each improvement that have been made to the MNCD method will be discussed as to be able to associate where the changes are made. This section will then conclude with a review of the new algorithm and the results thereof will be discussed in the next chapter.

4.4.1 Reduce the cut set $C(S)$

The first change that is implemented, is to decrease the size of the cut set, $C(S)$. It was decided that the size of the cut set must be made smaller in order to decrease the time it takes for the algorithm to execute. In their work, Ren *et al.* [2] made use of all of the vertices, and then made the cut set smaller by only considering vertices which contains at least one convex endpoint. This still contains a large amount of possible cuts as there exists a lot of vertices on an image. Thus, a few steps have been taken to decrease the cut set which includes making use of Harris corner detection, DCE and DT. Each one of these changes will be discussed in the next sub sections.

Harris Corner Detections

In order to obtain fewer contour vertices, HCD was used. When applying HCD the results produce a large number of potential corners depending on the variable values specified. The parameters for the HCD were chosen in such a way to produce a lot of points. This was decided in order to give more or less the same results for different objects. If the variables are too large the results get more accurate, but for the next object it might produce no results at all. Thus, the HCD was implemented to narrow down the number of vertices, and so make the number of vertices to consider as possible cuts less.

In order to solve the above variable problem, it was decided that making use of DCE will be a solution to this problem. Thus, applying the HCD will also decrease the number of vertices for the DCE algorithm to run through. In order to apply this algorithm, an OpenCV library was used. This algorithm works on the same principle as mentioned in the previous section.

After obtaining the different corner points by using HCD, the DCE algorithm will be looked at. This is discussed in the next sub section.

Discrete Contour Evolution

In order to obtain even less vertices, the DCE algorithm is used. It is important to note that this algorithm requires a stopping criteria, and if not specified will stop automatically at three - which means it will evolve the contour until there only remains three points - and thus a triangle. Using a triangle will be very unsatisfactory to describe a shape, as a lot of important information will be lost when this happens. It was therefore decided to try and automatically obtain a stopping criteria based on the contour of the object at hand.

As can be recalled from section 4.3.2 the equation that was used was:

$$K(\beta, l_1, l_2) = \frac{\beta l_1 l_2}{l_1 + l_2} \quad (4.12)$$

In order to solve the stopping criteria problem, an algorithm was created to solve this problem.

In order to better explain the algorithm, figure 4.13 has been added and an explanation on how to determine the stopping criteria is to follow. As can be seen in figure 4.13 (a) we have our original shape. In figure 4.13 (b) the contour of the original shape is shown, and figure 4.13 (c) shows the original area $A_{original}$ (in grey), and original length $L_{original}$ (pink vertices). In this case, the values are $A_{original} = 500$ and $L_{original} = 39$. The next step is then to determine the vertices with the least amount of contribution, this is shown as the cyan dots in (d). These vertices are then removed, and lines are drawn to the new vertices, shown in (e). Also shown in (e), although very small, is the error $error(I)$ in area (shown in dark blue). Now the new area in (f) can be

calculated as $newArea(I) = 490$ and the new length is $newlen(I) = 23$. Next, the area ratio $E(I) = \frac{eSum(I)}{Area(I)} = \frac{10}{500} = 0.02$ and the length ratio $S(I) = \frac{newlen(I)}{len(I)} = \frac{23}{39} = 0.589$ are calculated. Thus now $J(I) = E(I) + S(I) = 0.61$, which is only one entry in the vector $J(I)$. This process is then repeated until the length is equal to three, as three vertices are needed in order to form a polygon of any sort. Repeating these steps we found from (g) to (i) the new area and length. As can be seen in (h) the error becomes larger with a decrease in number of vertices.

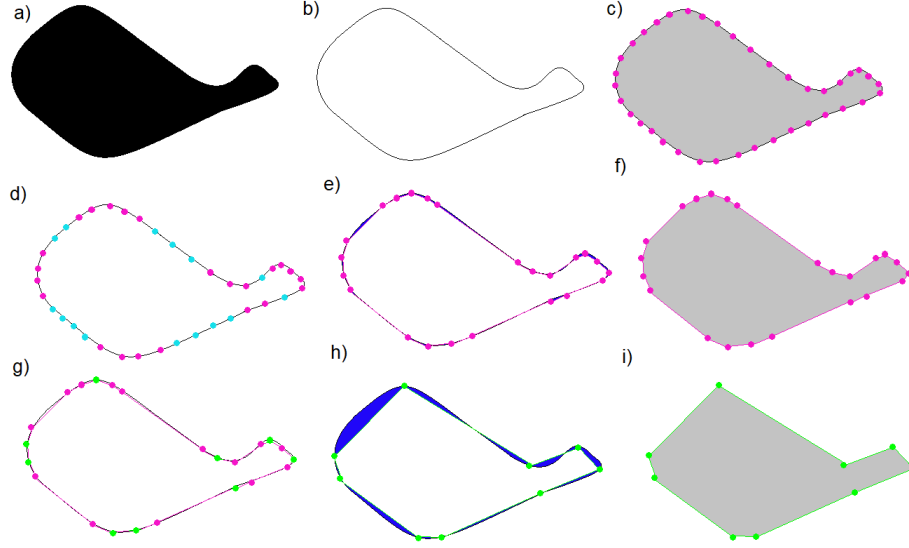


Figure 4.13: Algorithm to obtain optimal DCE explained step-by-step.

After each process has been calculated, the (J) is then plotted against the number of points that have been removed from the original length. In order to determine the stopping criterion, the minimum point will be considered, as that is where the shape will still keep all of the information of the original image, with the least number of vertices. An example plot of the above image and process is given in figure 4.14. A more formal description follows hereafter in order to mathematically express the process that was explained.

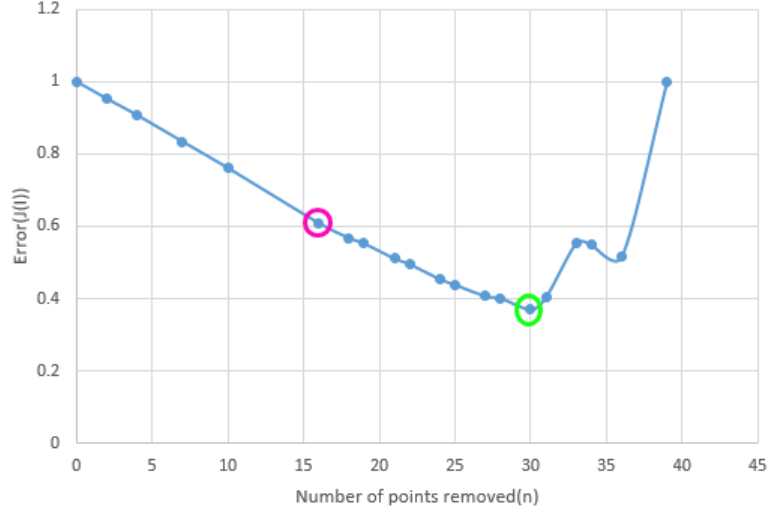


Figure 4.14: Plot of $J(I)$ vs. Number of parts removed. The pink circle and green circle indicates the DCE process of figure 4.13 between (d-e) and (g-i) respectively. The green circle also indicates the stopping criteria, which is 9 vertices.

The first step here is to determine the original shape contour, $Contour(S)$. After that has been determined the original length of the number of vertices is determined, $len_{original}$, as well as the original area of shape, $Area_{original}$. All of these functions are determined by using OpenCV functions. The next step is to create a clear image, $N(I)$. This image is a matrix filled with zeros that is of the same size as the original image I . After this is done, the contribution of each vertex is determined using equation 4.12. These values are then placed in a vector, $C(S)$ with all of the contributing factors of each vertex. In order to determine which vertex to delete, the minimum of vector $C(S)$ is determined. A new vector, **newC(S)**, is then created by storing the coordinates of $Contour(S)$ in the case where the contribution is not the minimum value of $C(S)$. It is known that the DCE will delete one vertex at a time, which will result in a length of $n - 1$ for each time the loop is executed. Thus to define it mathematically:

$$newC_i = \begin{cases} contour_i, & \text{if } c_i \neq \min\{C(S)\} \\ \emptyset, & \text{if } c_i = \min\{C(S)\} \end{cases} \quad (4.13)$$

Where $\mathbf{newC}_{(n-1) \times 1} = (newc_0, newc_1, \dots, newc_{n-1})^T$ is a vector. Once **newC** is determined, the new shape is drawn on the clear image, $N(I)$, which will exclude the deleted vertex. Now it is required to measure the new length, $m = length(\mathbf{newC})$, and the new area, $A_{new} = Area(\mathbf{newC})$.

In order to determine A_{diff} the first step would be to determine the error between the original image, $I(S)$, and the new image, $N(S)$. This is achieved by an exclusive disjunction (XOR) of the images with each other. The XOR operation will output true only when the two inputs differ - that is if one is true and the other is false. Thus the error, $E(S)$ can be determined by:

$$E(S) = I(S) \oplus N(S) \quad (4.14)$$

This operation will produce the error matrix $\mathbf{E}(S)$ of the same size as the original image, $I(S)$. In order to determine the difference in area A_{diff} , all of the values in $E(S)$ must be summed:

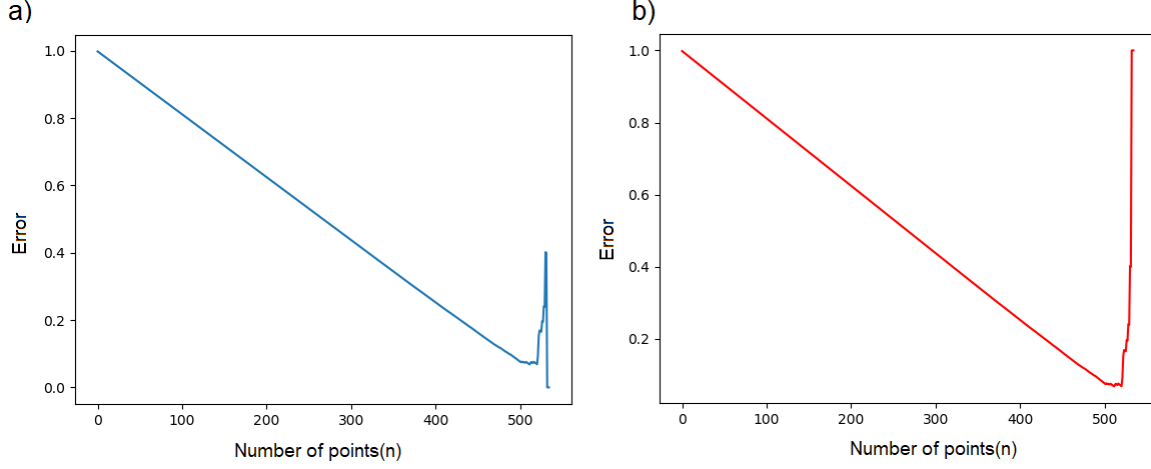


Figure 4.15: These two graphs show the effect of filling $\mathbf{J}$ with zeros (a) and ones(b). As can be seen the minimum values for (a) is zero and the index is equal to the original length where the index in (b) is not zero and thus the index is not equal the original length.

$$A_{diff} = \sum_{i=0}^n \sum_{j=0}^m E_{ij} \quad (4.15)$$

Where m and n represents the height and width of the original image. The next step is to determine the ratios of the areas and the lengths. For the area ratio, A_{diff} and $A_{original}$ will be used:

$$A_{ratio} = \frac{A_{diff}}{A_{original}} \quad (4.16)$$

As can be seen, this ratio will yield a small difference in the beginning of every evolution step and will increase as the number of vertices decrease. For the length ratios, $len_{original}$ and len_{new} will be used:

$$L_{ratio} = \frac{L_{new}}{L_{original}} \quad (4.17)$$

Here it can be seen that the length ratio will be decreasing with every evolution step. Next the $\mathbf{J}$ vector is determined:

$$\mathbf{J}(S) = A_{ratio} + L_{ratio} \quad (4.18)$$

Lastly, the minimum value of $\mathbf{J}$ is determined and the index of this value subtracted from the original length, $L_{original}$, to determine the stopping criteria, sc , for the evolution:

Let i be such that $J_i = \min\{\mathbf{J}\}$

$$sc = L_{original} - i \quad (4.19)$$

The vector $\mathbf{J}$ is the same length as the original contour $L_{original}$ and is filled with ones. This is done for the following reasons:

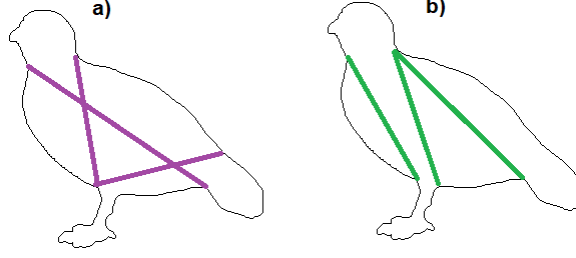


Figure 4.16: Figure demonstrating the concept of (a) overlapping and (b) non-overlapping cuts.

1. The evolution will continue to decrease until there is no vertices left. This will create a very small ratio of L_{ratio} as well as a small A_{ratio} and will result in zero.
2. As we are trying to determine the minimum number of vertices, while trying to keep the new shape as close to the original shape, the ratios will add up eventually add up to zero.
3. The minimum number of vertices will then be equal to the same number of the original length as $\min\{\mathbf{J}\}$ will be zero as $i = L_{original}$ yielding $sc = 0$.

Thus if $\mathbf{J}$ is filled with ones, the minimum value can be determined without the worry of having the shape staying the same as the original shape. This explanation can be seen in figure 4.15.

Thus, now that the DCE of the shape is determined and the number of vertices has been decreased to only a small number, the number of cut will also be significantly less. The next problem that is faced is that there exist a lot of cuts where overlapping takes place. As can be seen in figure 4.16, in (a) the cuts are overlapping each other, where in (b) the cuts are not overlapping each other. Even though there is for example ten remaining vertices, that means that for each vertex, there is at most nine options of cuts to consider, and so would end up with 55 possible cuts to consider. Thus, in order to reduce the number of cuts even more, an alternative way of cutting a shape must be looked at. Therefore applying Delaunay Triangulation to the remaining vertices will produce a triangulation of possible cuts that can be used to solve the problem. This will be discussed in the next subsection.

Delaunay Triangulation

Since one of the constraints to the equation 3.5 require that the cuts are not to overlap, DT is used to create less cuts and to avoid cuts overlapping. In doing this, the need of the overlapping matrix $\mathbf{B}$, as can be seen in equation 3.9, is unnecessary as none of the cuts will overlap. The DT algorithm will also ensure that that shortest possible cuts between the different vertices will be used, and thus implementing the *short-cut rule*. This algorithm is implemented using OpenCV libraries with the points as obtained by the DCE as input.

In order to apply DT in the project, OpenCV was used. Steps on how this was implemented are discussed next. The OpenCV DT function requires one to give a set of points that the DT can be applied to. These points are determined in the previous step by using DCE. The remaining vertices after the stop criteria was calculated and the evolved shape has been created, are the set of points given to the function as input.

Some of the steps followed to obtain the DT areas include: [19]:

1. Collect all the points in a vector.
2. Define the space you want to partition using a rectangle (**rect**)
3. Create an instance of the DT function called **Subdiv2D** with the rectangle in the previous step.
4. Insert the points into **subdiv** using **subdiv.insert (point)** function.
5. Use the function **subdiv.getTriangleList** to obtain a list of DT triangles.

The DT of an arrow is given in figure 4.17. As can be seen, there are no lines overlapping each other. The only down side to using DT is that some possible cut lines lie on the contour.

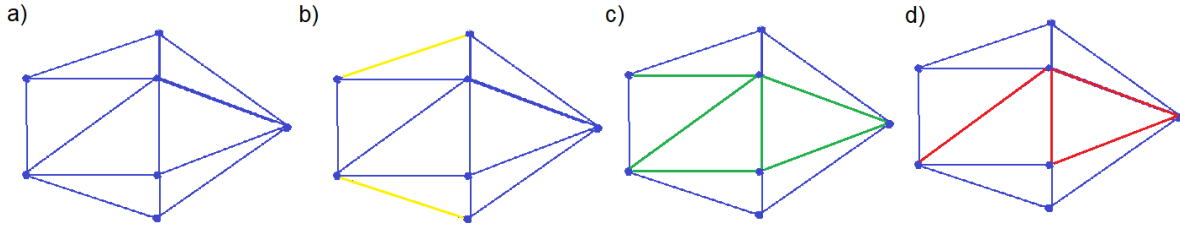


Figure 4.17: (a) A figure that shows the DT of an arrow when a set of points (blue) is given. Yellow lines (b) are to indicate that these lines aren't part of the contour, green lines (c) show duplicate lines, red lines (d) show desired triangulation.

In order to remove these cuts that lie on the contour, the lines connecting the two points was converted to *line* vectors as mentioned in section 4.4.3. This was done to see how far the points on the line lie from the contour. The distance of each point on the cut is taken and an average is calculated to determine the average closeness of the points to the contour. If the number of points of a cut is given as n , then this process can be described as follows:

$$d_{average} = \frac{\sum_{i=0}^n dist_i}{n} \quad (4.20)$$

Giving a mathematical expression to obtain the average distance, $d_{average}$, of a cut, with number of points n . This is calculated for each cut in the cut set $C(S)$. If the absolute distance $d_{average}$ is greater than a threshold, in this case the pixels, then the cut is kept else it is discarded. Thus, this will remove any cuts that lie on the contour, and will yield the final cut set, which should be a lot less than the original amount of cuts obtained.

Thus to conclude this subsection it was decided that in order to reduce the time complexity, the number of cuts can be reduced. In order to do this, the Harris Corner detection algorithm was applied, producing less vertices than using all of the points on the contour, which will then produce less cuts than considering all of the contour points. Next, the Discrete Contour Evolution algorithm is applied to reduce the number of vertices even more. Lastly, the Delaunay Triangulation algorithm was applied on the remaining vertices to produce less cut and also non-overlapping cuts, thus removing the need for the non-overlapping matrix in the final linear problem. This was the first major implementation attempt to decrease the time it takes for the MNCD algorithm to produce

results. The next improvement that needs to be implemented was to remove the use of Morse functions. This will be discussed in the next subsection.

4.4.2 Morse function removal

As the name suggests, the second change that is implemented is the removal of the Morse functions. This decision was made based on the fact that the Morse function is high in complexity and results in a time consuming process. In order to remove this and so use an alternative to this, it was decided that instead of using multiple Morse functions to determine a mutex areas, to use each vertex that remained after DCE as 'mutex areas'.

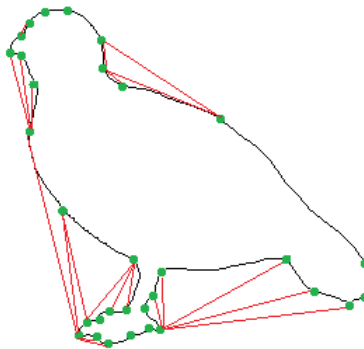


Figure 4.18: (a) shows all of the mutex pairs and how the Morse function is no longer used.

That is instead of creating areas where a shape can be split, the vertices where a line connecting them lies outside the shape, or cuts through the contour is considered a mutex pair. Making use of DT, there should not be any lines that cut through the contour, but rather lie on top of the contour as mentioned earlier. As can be seen in figure 4.18, the green points indicate the vertices after DCE and the red lines connecting these points outside the contour. All of these points will now be considered as mutex pairs and no areas will be identified as mutex areas. This will cause the convexity matrix, $\mathbf{A}$, to increase in size as there will be more mutex pairs than areas. Although this might cause some time delay at later stages, the use of the Morse function has been removed, and therefore will save a lot more time than the convexity matrix will cause delay by.

To conclude this subsection, it can be seen that removing the Morse function can decrease the time complexity, although it might increase the size of the convexity matrix. Next to be discussed is the smaller changes that will be implemented. This includes how the different vertices and lines are processed and how the mutex pairs as well as the convexity matrix is determined. This will be discussed in the next sub section.

4.4.3 Preprocessing

In this section the different preprocessing algorithms will be discussed. This includes the preprocessing of the vertices as well as the lines. Some of the preprocessing needed for the vertices to be used is to determine the curvature of a vertex as well as the concavity thereof. For the lines, there are a lot of different forms that a line can be represented, so as to select the correct one for this application, the process thereof will be discussed.

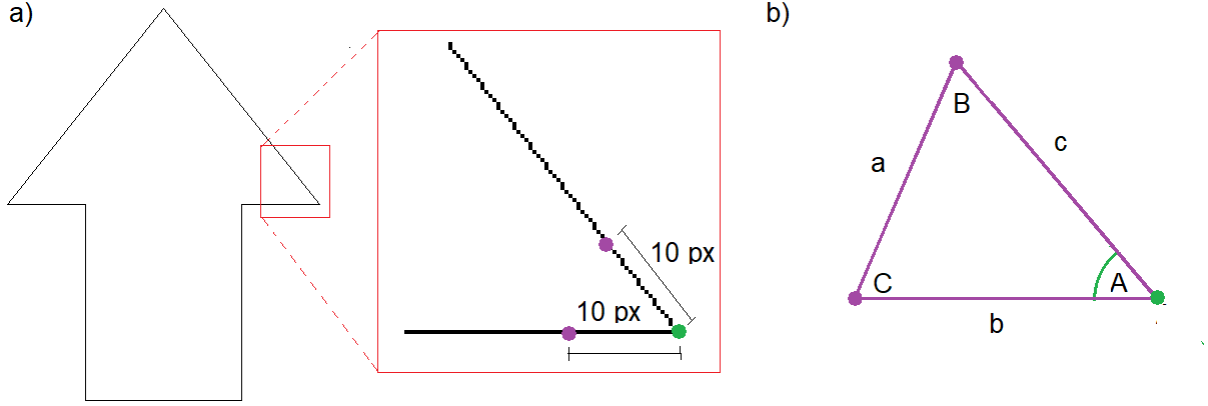


Figure 4.19: (a) shows the corner that has been detected and a zoomed section showing that two points approximately ten pixels away are also used to determine the angle of the corner(A) by using the cosine rule.

Vertices

Concavity

In order to determine the concavity of a point, the following was done. First, after a point has been identified, two points are to be drawn on the contour that is an equivalent distance away from the point. This required the points to be on the contour. In order to ensure that they are in the contour, the distance from each corner point and the contour is determined. If the distance is not equal to zero, the point is moved to the contour until the distance is zero. Once this is done it is needed to determine the index that is ten intervals away from the point in the contour vector. Thus, following the description of figure 4.19 (b), mathematically:

Let i be such that $contour_i = point(A)$ then:

$$point(B) = contour_{(i+10)} \text{ and } point(C) = contour_{(i-10)} \quad (4.21)$$

Now that the points B and C are determined, the cosine rule can be used to determine the angle of A . That is:

$$\cos(A) = \frac{b^2 + c^2 - a^2}{2bc} \quad (4.22)$$

From equation 4.22 it can be seen that we know the values of $b = 10$ and $c = 10$, but we do not know the value of a . This value can easily be determined by using the distance formula:

$$a = \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2} \quad (4.23)$$

The next step is to determine if the angle is convex or concave. As the triangle will always result in a convex result, it was seen that the triangles with one side, a outside the contour shape is the triangle with angle, A , concave instead of convex.

To determine the position of a relative to the contour an OpenCV function called the **point-PolygonTest** is used. This function takes as input the contour, a coordinate pair to be tested and a true or false. When true is given as input, the actual signed distance will be returned, else it will return a +1,-1 or 0 for points lying inside, outside or on the contour respectively.

Thus in order to use this function, a coordinate pair has to be given. It was decided that the point between B and C will be suitable as it will always lie inside or outside the shape. Thus using the midpoint formula:

$$M = \left(\frac{x_b + x_c}{2}, \frac{y_b + y_c}{2} \right) \quad (4.24)$$

Thus using the OpenCV function and setting the third criteria to false:

$$dist = pointPolygonTest(M, Contour, false) \quad (4.25)$$

This function will return a +1 when M is inside or a -1 when M is outside the contour. Referring to figure 4.19 we find that when the distance, $dist = 1$, the inner angle, $\angle A_{inner}$, can be used as the angle of point A , $\angle A$. However, when the distance, $dist = -1$, the angle gets calculated by subtracting the inner angle from 2π (360°). That is: $\angle A = 2\pi - \angle A_{inner}$.

These values are stored and will be used at a later stage again. Next the method to determine the curvature will be discussed.

Curvature

Recall from equation 3.6 that the curvature of a point is to be used as part of the minima rule [25]. In order to determine the curvature of a point, the same process will be used as above. Points B and C are used again as mentioned above. In order to determine the curvature of a curve, the reciprocal of the radius can be used [79]. Thus if we have three points, $A(a_1, a_2)$, $B(b_1, b_2)$ and $C(c_1, c_2)$, all of which lie on the circumference of the circle, the equation of the circle can be determined by solving the following expression:

$$\begin{vmatrix} (x^2 + y^2) & x & y & 1 \\ (a_1^2 + a_2^2) & a_1 & a_2 & 1 \\ (b_1^2 + b_2^2) & b_1 & b_2 & 1 \\ (c_1^2 + c_2^2) & c_1 & c_2 & 1 \end{vmatrix} = 0 \quad (4.26)$$

Using Laplace expansion, this matrix can be written as:

$$(x^2 + y^2) \begin{vmatrix} a_1 & a_2 & 1 \\ b_1 & b_2 & 1 \\ c_1 & c_2 & 1 \end{vmatrix} - x \begin{vmatrix} (a_1^2 + a_2^2) & a_2 & 1 \\ (b_1^2 + b_2^2) & b_2 & 1 \\ (c_1^2 + c_2^2) & c_2 & 1 \end{vmatrix} + y \begin{vmatrix} (a_1^2 + a_2^2) & a_1 & 1 \\ (b_1^2 + b_2^2) & b_1 & 1 \\ (c_1^2 + c_2^2) & c_1 & 1 \end{vmatrix} - 1 \begin{vmatrix} (a_1^2 + a_2^2) & a_1 & a_2 \\ (b_1^2 + b_2^2) & b_1 & b_2 \\ (c_1^2 + c_2^2) & c_1 & c_2 \end{vmatrix} = 0 \quad (4.27)$$

In order to simplify the equation c_i will be used to represent the coefficients of each of the terms above (see equation 4.32). Using Laplace expansion on each c_i term, we find:

$$c_1 = a_1 \begin{vmatrix} b_2 & 1 \\ c_2 & 1 \end{vmatrix} - a_2 \begin{vmatrix} b_1 & 1 \\ c_1 & 1 \end{vmatrix} + 1 \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix} \quad (4.28)$$

similarly:

$$c_2 = (a_1^2 + a_2^2) \begin{vmatrix} b_2 & 1 \\ c_2 & 1 \end{vmatrix} - a_2 \begin{vmatrix} (b_1^2 + b_2^2) & 1 \\ (c_1^2 + c_2^2) & 1 \end{vmatrix} + 1 \begin{vmatrix} (b_1^2 + b_2^2) & b_2 \\ (c_1^2 + c_2^2) & c_2 \end{vmatrix} \quad (4.29)$$

$$c_3 = (a_1^2 + a_2^2) \begin{vmatrix} b_1 & 1 \\ c_1 & 1 \end{vmatrix} - a_1 \begin{vmatrix} (b_1^2 + b_2^2) & 1 \\ (c_1^2 + c_2^2) & 1 \end{vmatrix} + 1 \begin{vmatrix} (b_1^2 + b_2^2) & b_1 \\ (c_1^2 + c_2^2) & c_1 \end{vmatrix} \quad (4.30)$$

and

$$c_4 = (a_1^2 + a_2^2) \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix} - a_1 \begin{vmatrix} (b_1^2 + b_2^2) & b_2 \\ (c_1^2 + c_2^2) & c_2 \end{vmatrix} + a_2 \begin{vmatrix} (b_1^2 + b_2^2) & b_1 \\ (c_1^2 + c_2^2) & c_1 \end{vmatrix} \quad (4.31)$$

Thus simplifying equation 4.27 to:

$$c_1 (x^2 + y^2) - c_2 x + c_3 y - c_4 = 0 \quad (4.32)$$

Now by completing the squares, equation 4.32 can be re-written in the standard circle equation form:

$$\left(x - \left(\frac{c_2}{2}\right)\right)^2 + \left(y - \left(\frac{c_3}{2}\right)\right)^2 = \frac{2c_4 + c_2c_4 + c_3c_4}{2c_1} \quad (4.33)$$

From which the radius r and the centre point, C , can be determined as follows:

$$r = \sqrt{\frac{2c_4 + c_2c_4 + c_3c_4}{2c_1}} \quad (4.34)$$

and

$$C = \left(-\frac{c_2}{2}, \frac{c_3}{2}\right) \quad (4.35)$$

Finally, the curvature can be determined by using:

$$\kappa = \frac{1}{r} \quad (4.36)$$

Figure 4.20 demonstrates the concept of curvature, where in (a) two points on the contour is selected a distance of 10 pixels away from the corner whose curvature is calculated. In (b) a circle is drawn with the three points lying on the circumference thereof, and these points are used to determine the curvature of the corner. As with the convexity, the curvature is given a negative sign if the midpoint, M lies inside the contour, and a positive sign if it lies outside the contour. That is:

$$\kappa = \begin{cases} -\frac{1}{r}, & \text{if } dist = 1 \\ \frac{1}{r}, & \text{if } dist = -1 \end{cases} \quad (4.37)$$

Now that the preprocessing algorithms have been discussed for the vertices, it can be seen where the different values are obtained in the later stages of implementing the MNCD algorithm. It is also important to note that the methods chosen in this preprocessing step are chosen in such a way as to simplify the manner in which the convexity and the curvature of the vertices are determined. Now that the vertices' preprocessing has been discussed, the discrete line preprocessing will be discussed.

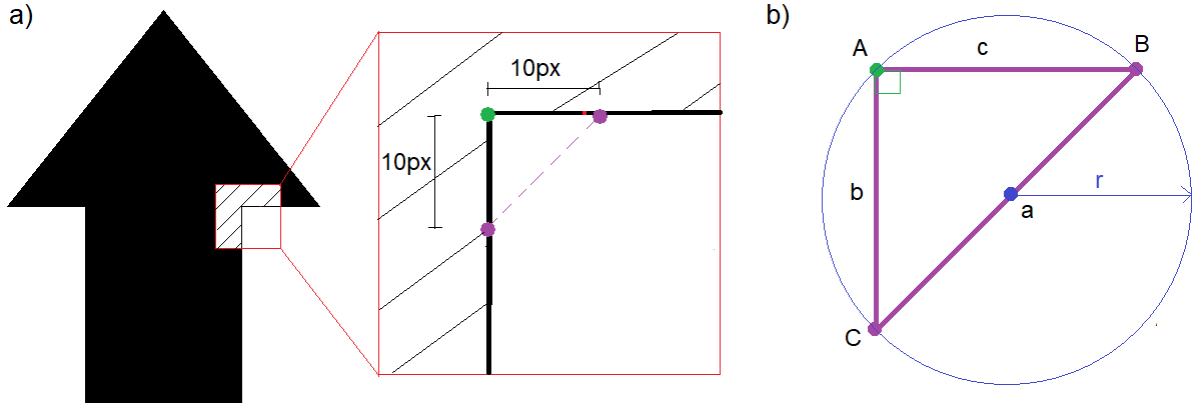


Figure 4.20: (a) shows the corner that has been detected and a zoomed section showing that two points approximately ten pixels away are also used to determine the curvature of the corner(A).

Discrete lines

In this subsection, creating discrete lines will be discussed. In any image, a line drawn will be a discrete line. In order to determine if a line lies inside, outside or on the contour of a shape, the discrete points of a line can be used. This is important to determine when two points are mutex pairs for example, or when cuts are drawn on the contour and cannot be used to cut a part. For any lines to be drawn, the input of the two end points are required and then a line is drawn between these two points.

OpenCV has a function to draw lines called *line*. Input to the function is the image you wish to draw on, the coordinate pairs pt_1 and pt_2 are the endpoints of the line, *RGB* is the colour of the line and *type* is the line type. The last two parameters are often discarded.

This function makes use of the *Bresenham's line algorithm* to determine which coordinates to store. The steps of this algorithm are described below. Take $A = (a_x, a_y)$ and $B = (b_x, b_y)$ as the two input points, then the first step will be to obtain the absolute value of the difference between these two points:

$$\Delta X_{abs} = |\Delta X| \text{ where } \Delta X = b_x - a_x \quad (4.38)$$

and

$$\Delta Y_{abs} = |\Delta Y| \text{ where } \Delta Y = b_y - a_y \quad (4.39)$$

The next step would be to determine if the line is vertical $b_x = a_x$, horizontal $b_y = a_y$, or is diagonal $b_x \neq a_x$ and/or $b_y \neq a_y$. This is followed by a test see which x coordinate and which y -coordinate is larger.

The Bresenham's algorithm then follows a simple set of rules to determine the next value of the vector. If the length of $\Delta X_{abs} = n$, then this matrix is of size $n \times 3$. The first two columns contain the x and y coordinates of the line, while the third column shows the intensity thereof, $I(x, y)$. Therefore, the matrix can be mathematically described when considering the following conditions as:

If $a_x = b_x$ and $a_y > b_y$ then:

$$\sum_{i=1}^3 \sum_{j=0}^n = line_{ij} \text{ where } \begin{cases} line_{1j} = a_x \\ line_{2j} = j \\ line_{3j} = I(a_x, j) \end{cases} \text{ if } \forall j \in Image \quad (4.40)$$

for the above case, if $b_y > a_y$ then equation 4.40 stays the same, but the order of $line_{2j}$ swaps around. This is in actual terms a vertical line.

If $a_y = b_y$ and $a_x > b_x$ then 4.40 becomes:

$$\sum_{i=1}^3 \sum_{j=0}^n = line_{ij} \text{ where } \begin{cases} line_{1j} = j \\ line_{2j} = a_y \\ line_{3j} = I(j, a_y) \end{cases} \text{ if } \forall j \in Image \quad (4.41)$$

for the above case, if $b_x > a_x$ then equation 4.41 stays as is, but $line_{1j}$ order will change this time. This line can be seen as a horizontal line.

Lastly, if $a_x \neq b_x$ and $a_y \neq b_y$, the first test would be to check the slope. If the slope is steep, that is $\Delta Y_{abs} > \Delta X_{abs}$ we can calculate the steep slope by:

$$m_{ss} = \frac{\Delta X}{\Delta Y} \quad (4.42)$$

else the slope is calculated as normal by:

$$m_s = \frac{\Delta Y}{\Delta X} \quad (4.43)$$

Knowing this, equation 4.40 becomes:

$$\sum_{i=1}^3 \sum_{j=0}^n = line_{ij} \text{ where } \begin{cases} line_{1j} = m_{ss}(j - a_y) + a_x \\ line_{2j} = j \\ line_{3j} = I(a_x, j) \end{cases} \text{ if } \forall j \in Image \quad (4.44)$$

and equation 4.41 becomes:

$$\sum_{i=1}^3 \sum_{j=0}^n = line_{ij} \text{ where } \begin{cases} line_{1j} = j \\ line_{2j} = m_s(j - a_x) + a_y \\ line_{3j} = I(j, a_y) \end{cases} \text{ if } \forall j \in Image \quad (4.45)$$

And the same conditions as mentioned above apply to both equations 4.44 and 4.45. It is important to note that the matrix *line* will be an integer matrix, as the pixels in an image are all integers. Now that the *line* is determined, we can use this information in conjunction with the function *pointPolygonTest* mentioned in (4.25) to determine if a line lies inside, outside or on the contour. In order to do this, it was decided to classify a line as being outside the contour if 3 or more points lie outside the contour. The same for if the line lies inside the contour, three or more points must lie inside the contour, else the line lies on the contour.

Thus, in order to be able to tell if a line lies on, outside or inside of a contour this function stores the values of each line to be recalled when needed. Concluding this subsection, one can see that in many cases it seems unnecessary to store each point of a discrete line, but in our case it is very helpful to be able to access every point of a line.

Ψ -concavity

After the mutex pairs have been identified, the concavity of each mutex pair must be determined. As described in section 3.3 the concavity is determined by taking the maximum perpendicular length of each line to the contour.

$$\text{concave}(P_i) = \max_{(v_1, v_2) \in P_i} \{\text{concave}_m(v_1, v_2)\} \quad (4.46)$$

where $\text{concave}_m(v_1, v_2)$ in this case is measured using the distance equation. In order to determine which point is perpendicular we make use of simple geometry. By using the line equation, getting the inverse of the slope of the line, and applying that to a known point, one can determine which point on a line will be the point where perpendicularity is achieved. Mathematically, we look at the line equation:

$$y = mx + c \quad (4.47)$$

assume point $A(a_1, a_2)$ and point $B(b_1, b_2)$ are the two end points of the mutex pair. Then the slope of the line connecting the two mutex points can be written as:

$$m_{AB} = \frac{b_2 - a_2}{b_1 - a_1} \quad (4.48)$$

we can then determine the equation for the line AB by using the "point-slope" equation:

$$y - y_1 = m_{AB}(x - x_1) \quad (4.49)$$

and replacing the values and choosing point $A(a_1, a_2)$ we find:

$$y = \frac{b_2 - a_2}{b_1 - a_1}(x - a_1) + a_2 \quad (4.50)$$

we know that for two lines to be perpendicular, the product of the slope of the two lines must be equal to -1 . That is:

$$m_{AB} \times m_{CD} = -1 \quad (4.51)$$

Thus knowing that the furthest point of a mutex pair will always be a concave point, we know the coordinates of $C(c_1, c_2)$, and can thus determine the coordinates of $D(d_1, d_2)$. Firstly, to determine the slope, we use equation 4.51:

$$m_{CD} = -\frac{1}{m_{AB}} = -\frac{1}{\frac{b_2 - a_2}{b_1 - a_1}} = -\frac{b_1 - a_1}{b_2 - a_2} \quad (4.52)$$

now to determine the equation of the line BD we use the equation 4.49:

$$y = -\frac{b_1 - a_1}{b_2 - a_2}(x - c_1) + c_2 \quad (4.53)$$

Solve for $D(d_1, d_2)$ by finding the point of intersection of AC and BD . This can be done by equating equations 4.50 and 4.53 respectively and solving for x or in this case d_1 :

$$\frac{b_2 - a_2}{b_1 - a_1}(x - a_1) + a_2 = -\frac{b_1 - a_1}{b_2 - a_2}(x - c_1) + c_2 \quad (4.54)$$

simplified the equation can be written as:

$$d_1 = -\frac{\frac{a_1 b_1 - b_1 c_1}{a_2 - c_2} + \frac{a_1 a_2 - a_1 c_2}{a_1 - c_1} + b_2 - a_2}{\frac{a_2 - c_2}{a_1 - c_1} - \frac{c_1 - a_1}{a_2 - c_2}} \quad (4.55)$$

and thus d_2 can be found using equation 4.53:

$$d_2 = -\frac{b_1 - a_1}{b_2 - a_2}(d_1 - c_1) + c_2 \quad (4.56)$$

once these points are found, the distance formula can be used to determine the concavity of each mutex pair by:

$$concavity_i = \sqrt{(c_{1i} - d_{1i})^2 + (c_{2i} - d_{2i})^2} \text{ for } \{i | i = 0, 1, \dots, m\} \quad (4.57)$$

where m is the number of mutex pairs.

Now that there is a process to determine the concavity of the mutex pairs, the preprocessing thereof is done discussing, and these methods can be implemented into the original MNCD method to apply the new changes.

4.4.4 Determining the convexity matrix (A-matrix)

In their work, Ren *et al.* [2] suggests that the convexity matrix is obtained by checking to see if a cut separates mutex pairs. The means of how this is determined is not discussed in their paper though and therefore a method to determine this is discussed in this subsection.

In order to determine if a cut separates a mutex pair, the following is done. The first step is to draw each cut separately, and split the part into two parts. The next step then is to only draw one of the two parts. This is then used with the corner points to determine which points are on the shape and which are not. These two steps are depicted in figure 4.21.

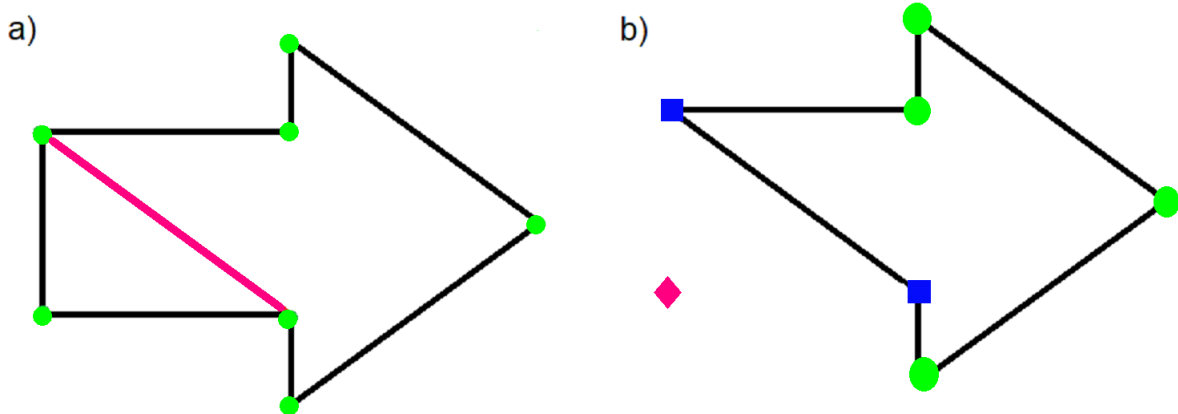


Figure 4.21: a) Part shown with one possible cut, in magenta. b) shows the one part that is cut and the green circle corner points that lie on the cut part, the blue square corner points that form part of the cut coordinates and the magenta diamond point that is outside the cut part.

In *a*) it can be seen that the first cut in the cut-set $\mathbf{C}(S)$ is considered, shown in the magenta line. In *b*) the part is cut into two parts and only one part is shown. The corner points on this part is then indicated as green circles, while the corners on the cut is blue squares and the corner out of the part is shown in magenta diamond.

For each corner point that lies on the contour of the cut part, a one is assigned, while the rest of the corner points are assigned zeros. These values are then stored in a "look up" matrix, $L(S)$. In order to determine if the point lies on the cut part, the contour, $\text{contour}_p(S)$, of the cut part must first be determined. Once this is determined, the **pointPolygonTest** of equation 4.25 is used to determine if the point lies on the contour or not. That is:

$$\text{dist} = \text{pointPolygonTest}(M, \text{Contour}, \text{false}) \quad (4.58)$$

this will return a 1 if the point lies inside the contour, -1 when it lies outside the contour or 0 when on the contour. Thus if the $\text{dist} = 0$, then the point is assigned a one else a zero.

In order to avoid unwanted ones in the $\mathbf{A}$ it is important to also classify corner points that lie on the cut line as 'outside' the part. These are normally the endpoints of the cut, for example if the end points are called $F(x_1, y_1)$ for the starting point and $G(x_2, y_2)$ for the ending point, the points F and G will lie on the cut parts' contour. Thus, looking at the figure 4.21 the blue square points will be classified as zeros. Therefore the look-up matrix can be created by:

$$l_i = \begin{cases} 1, & \text{if } \text{dist} = 0 \text{ and endpoints} \neq S \text{ or } E \\ 0, & \text{otherwise} \end{cases} \quad (4.59)$$

thus yielding a vector $\mathbf{L}(S)$, with size n where n is the number of corner points. After this vector has been created, the $\mathbf{A}$ matrix can be determined column by column.

This is done by checking the coordinate pairs of all of the mutex pairs. For each mutex pair, mp_i , there will be two sets of coordinates to indicate the mutex pair. Thus, for each mutex pair, the first coordinate pair, $A(x_a, y_a)$ is looked up in the look-up vector, giving us the first half, denoted as $A_{i,1a}$. If the value is one in the look-up vector, it will also be one in $A_{i,1a}$. The second coordinate pair, $B(x_b, y_b)$ is then looked up, and we obtain the second half of the column, denoted $A_{i,1b}$. These two vectors are then exclusive disjunctive or better known as 'XOR'-ed with each other to reveal the first column of $\mathbf{A}$, denoted by $\mathbf{A}_{i,1}$. Thus, mathematically it can be written as:

$$a_{i1a} = \begin{cases} 1, & \text{if } A(x_a, y_a) \in \mathbf{L}(S) \\ 0, & \text{otherwise} \end{cases} \quad (4.60)$$

and

$$a_{i1b} = \begin{cases} 1, & \text{if } A(x_b, y_b) \in \mathbf{L}(S) \\ 0, & \text{otherwise} \end{cases} \quad (4.61)$$

Now

$$a_{i1} = a_{i1a} \oplus a_{i1b} \quad (4.62)$$

These equations are specifically used to determine the first column of the $\mathbf{A}$ matrix. Thus if $i = 0, 1, 2, \dots, m$ where m is the number of mutex pairs, and if $j = 0, 1, 2, \dots, n$ where n is the number of cuts, then equation 4.62 becomes :

$$a_{ij} = a_{ija} \oplus a_{ijb} \quad (4.63)$$

Which will give an $\mathbf{A}$ matrix of size $m \times n$. This method was created as to determine which cuts separates which mutex pairs. This is not a improvement on the current method of MNCD, but rather an explanation on what was done to determine if a cut separates a mutex pair.

4.4.5 Identifying simpler shapes

The last step that will be added to the MNCD method is identifying simpler shapes. The idea behind adding this algorithm is to simplify any outlines into simple shapes. Lets look for example at the arrow shown in figure 4.22. As can be seen, this arrow can be broken down into a triangle and a square. If a new image is decomposed into a triangle and a square, we know that it will fall into this 'category' of classification. Thus, the discussion in this subsection is about obtaining simpler shapes.

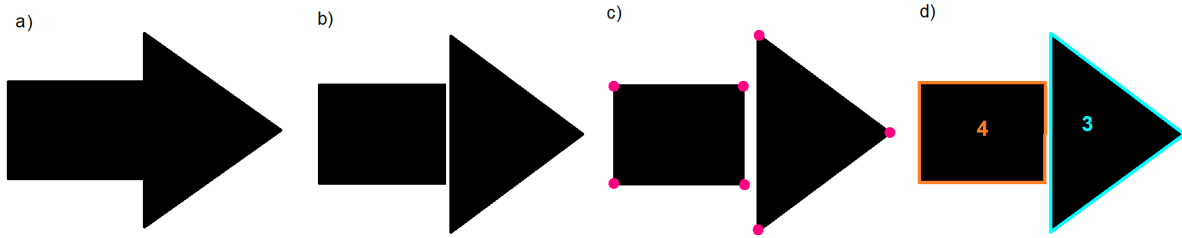


Figure 4.22: Figure showing (a) final selected cut and (b) two simpler shapes identified

In order to do this, the original image is converted to black and white, ensuring that the background is white. This is shown in figure 4.22 (a). Next the final cut coordinates are used to draw white lines on the black shape, and thus decomposing the shape. This can be seen in figure 4.22 (b).

The next step will then be to obtain the contours of the new shapes. In order to only store the corner points of the polygons after decomposition, an OpenCV function was used to determine the contours. The results of this applied function is shown in figure 4.22 (c).

The last step is to determine the amount of sides each decomposed part has. This is done with the OpenCV function called the polygon approximation function. This function requires the contours identified from the previous step as input to be able to approximate the polygon to be produced. This step is shown in figure 4.22 (d).

It is important to note that this method will only be used to identify the number of sides the shape has, and won't be used to identify specific shapes, for example, a right-angle triangle and an acute triangle will both be classified as a triangle. The same goes for a square, rectangle and parallelogram, as they will all be classified as quadrilaterals.

Each shape is then classified according to the number of points the approximate contour vector

has. Mathematically, it can be written as:

$$shape_i = \begin{cases} triangle, & \text{if } len(contour_i) = 3 \\ quadrilateral, & \text{if } len(contour_i) = 4 \\ pentagon, & \text{if } len(contour_i) = 5 \\ hexagon. & \text{if } len(contour_i) = 6 \\ check, & \text{otherwise} \end{cases} \quad (4.64)$$

As can be seen, for now the classification stops at hexagons, and only starts at triangles. This is because a shape with only two points can only be drawn as a line, and a shape with more than six points may be broken down into more simple points. In that case, the decomposition will be run again on that shape, until the shape can be classified as one of the above in equation 4.64.

4.5 Conclusion

In this chapter a discussion on the implemented improvements are done. The first topic of discussion was an algorithm overview to give the reader a better understanding of what is to follow. Next the improvements were discussed starting with the stopping criteria of discrete contour evolution (DCE) to make decomposition easier. Next corner detection is discussed and for each corner, concavity and curvature is calculated. The corner, if not on the contour of the shape, is then also moved onto the contour. Next discrete lines are discussed in order to determine if lines lie inside, on or outside the contour. After that a discussion on the MNCD method is done and it is found that using the Non-overlapping matrix is no longer needed as Delaunay Triangulation (DT) was used. Mutex pairs are then discussed and how to go about to determine if a pair of points are mutually exclusive to each other. The Ψ -concavity is then discussed as this is the parameter used to determine the concavity threshold. Next a discussion is done on how the $\mathbf{A}$ matrix as well as the cost function $\mathbf{w}$ are determined. This is followed by a discussion on the Binary Linear Integer Programming (BILP), and lastly how the simple shapes are identified.

It can be seen that there is a lot of different techniques that can be used to improve on the MNCD method. It is important to keep in mind that these improvements are implemented to try and improve the amount of time it takes to decompose a shape, while keeping the accuracy the same. In the next chapter, the experimental results will be discussed on the improvements that is implemented.

Chapter 5

Experimental results

In chapter 2 the MNCD algorithm was chosen on the basis of the close correspondence between its decomposition results and human shape decomposition. However the MNCD algorithm is quite slow. Improvements on the MNCD algorithm were suggested in chapter 4 in order to speed it up. In this chapter the improved MNCD algorithm is going to be applied to the same pictures from the MPEG-7 dataset, that were used in the MNCD experimental results to test the different parameters and how they change or react to the improvements.

The MPEG-7 dataset consists of 70 classes each having 20 different shapes , for a total of 1400 shapes [46]. The library contains a variety of natural and artificial objects. This dataset has been used in many shape decomposition papers before [2, 25, 12, 11, 10]. The same general objects are used in these papers in order to compare the different shape decomposition results with each other. This dataset was chosen for the same reason as mentioned above - to be able to compare the results of this algorithm to those tested before.

The results will also be compared to the ground truth experiment to see if the accuracy is kept the same. This will be referred to as the ground truth for the rest of this chapter. Based on these results the hypothesis will be revisited and possible improvement will be discussed in the next chapter. The 8 pictures that have been used for the experimental comparison between the MNCD algorithm and the improved algorithm are shown in figure 5.1.



Figure 5.1: *Figures that form part of the MPEG-7 shape dataset and that will be used in the experiments.*

This chapter will start off by discussing the research question and the hypothesis. This will be followed by the different experiments to test these hypothesis and will then be concluded by comparing the results to the expected outcomes. Possible improvements and future work will then be discussed in the next chapter as a conclusion to this dissertation.

5.1 Formulating hypothesis

Before the experiments start, a short revisit to the research question will be made and a hypothesis will be formed. Recall from chapter 1 that the investigative question is: Which decomposition method will decompose a complex shape into the least number of simple shapes, that relates to the ground truth, in the shortest amount of time? Thus, a hypothesis that can be formed based on this question will be: An improvement made on the Minimal Near-Convex Decomposition algorithm will decompose a complex shape into the least number of simple shapes, that relates to ground truth, in the shortest amount of time.

Having our main hypothesis formed, some sub questions can be looked at. Some of these sub-questions will be based on the experiment done in chapter 2. The first sub-question that can be formed to answer the main question would be how accurate will the improved method be compared to the MNCD method. Thus, the hypothesis that can be formed around this is that the improved method is expected to maintain the same accuracy as that of the MNCD method. Here accuracy refers to how closely a shape decomposition algorithm relates to human shape decomposition. This correspondence was measured by doing an experiment where by people were asked to decompose complex shapes into parts. More details of this is given in section 2.2.1.

This will then be followed by a next question being that will the time it takes to decompose a complex shape be less than that of the MNCD method? Thus, formulating a hypothesis from this: The improved algorithm will take less time than that of the MNCD algorithm.

The third question that can be asked is that does the improved algorithm produce that same amount of parts as the MNCD algorithm does? Thus, a hypothesis that can be formed is that the number of parts will be equal to or less than the MNCD algorithm.

The fourth question that can be asked is will the improved method have the same invariances than that of the MNCD method? Thus, hypothesising this question will give us: The improved method will have the same invariances than the MNCD method.

The final sub question that comes to mind will be: how does the improvements made to the MNCD method effect the parameters used in the method? Recall from chapter 3 that there are three main parameters that might have different outcomes on the results the first, and most important being Ψ . This is the concavity measure and is used to determine how convex a part is to be after decomposition. Next is the β parameter, that is used in the cost-function $\mathbf{w}$ to determine the relationship between the *short-cut rule* and the *minima rule*. Lastly we have the parameter λ which is used to determine the visually naturalness of the results. Thus, a hypothesis for this question will be that the parameters behaviour will not change due to changes in the MNCD method.

Thus, to conclude this subsection, experiments will be done on the following:

- Test to see if the results are as close to the ground truth as the MNCD methods is - thus testing accuracy.
- Test to see if the results are produced at a faster rate than the MNCD method.
- Test to see if the number of parts produced after decomposition is equal to or less than the MNCD algorithm.
- Test to see if the invariances are the same as that of the MNCD method.
- Test to see if the parameters are affected by the changes in the MNCD algorithm.

Each one of these experiments will be done in the rest of the chapter, and the results and outcomes will be discussed. All of these experiments are done in order to compare the accuracy of the improved solution to those that already exist and to see where possible further research or improvements can be done in the future.

5.2 Experiments

5.2.1 Accuracy evaluation

In this section the accuracy of the improved method will be tested. In order to do this several images from the MPEG-7 dataset will be given as input to the algorithm, and the output will be compared to that of the ground truth experiment done in chapter 2. The accuracy¹ is determined by looking at the results of the dataset obtained from the experiment - that can be seen in appendix B. Recall from section 2.2.1 that in the experiment a questionnaire was completed where humans were asked to decompose complex shapes. The results of the questionnaires were recorded and the most prominent cuts selected as the final decomposition cuts.

All of the cuts in the decomposed shaped that agree with the cuts of the ground truth, c_{dm} are then subtracted from the total number of the ground truth cuts, c_{hp} and this is then divided by the total number of the ground truth cuts, c_{hp} and multiplied by 100 in order to obtain a percentage deviation from the ground truth. That is:

$$\%deviation_{hp} = \frac{|c_{hp} - c_{dm}|}{c_{hp}} \times 100 \quad (5.1)$$

Thus, the lower the ground truth is, the better the method is. The results of the ground truth cuts (left) and the final cuts of the improved method (right) is shown in figure 5.2. The thick red lines on the figures on the right indicate algorithm cuts that are considered as "agreeing" with cuts from the ground truth cuts, while the thin red lines are algorithm cuts that do not agree with the ground truth. In order for the cut to agree, the endpoints must lie within 80% of the ground truth cut on the contour of the object.

¹ *Accuracy* here refers to how closely the shape decomposition of an algorithm relates to the shape decomposition of humans.

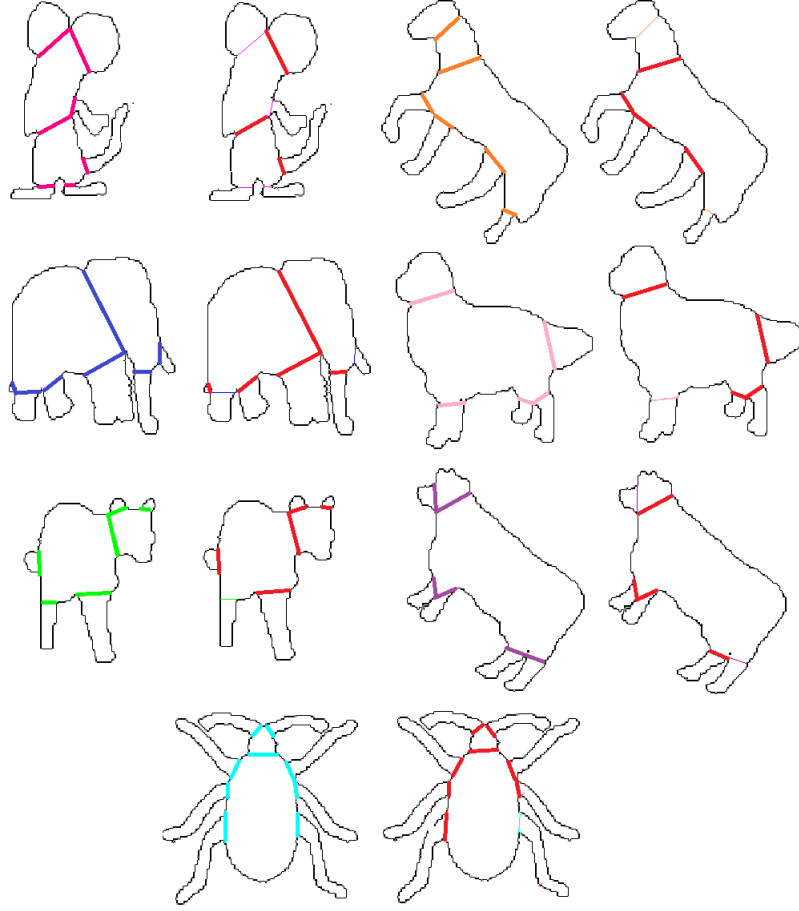


Figure 5.2: Figure showing several images used from the MPEG-7 dataset. Each picture on the left represents the ground truth. On the right the picture of the algorithm cuts is shown with thick red lines indicating cuts that agree, while the thin red lines shows cuts that do not agree.

Using the above seven images from the MPEG-7 dataset that have been used for the experiment in Chapter 2, and setting parameter values² as follows: $\psi = 0.5R$, $\lambda = \frac{1}{\sum_{i=0}^n w_i}$ and $\beta = 1$, the results are summarized in table 5.1 below:

²These parameter values are mentioned in chapter 3 and symbols R represents the radius of the enclosing circle of the object, n represents the number of cuts that are considered before decomposition and w is the cut cost function.

Table 5.1: Table showing the average % deviation for the MNCD and the improved method from the ground truth experiment for the pictures of the MPEG-7 dataset.

Number of cuts agreeing with human perception (% deviation)		
Picture	Ren (MNCD)	Improved Method
Cow	17	33
Elephant	0	29
Dog	40	20
Mouse	14	57
Bug	0	11
Cat	0	17
Horse	0	33
Average %	10	29

As can be seen, the Improved method has an average of 29% deviation, whereas the MNCD method has a deviation of 10%. Thus, it can also be seen that the accuracy is not kept at the same level as it should, and it can thus be concluded that the improvements implemented will cause the accuracy to go down with 19%. This is a rather large amount and the complete experiment should be looked at a whole to be able to determine if the speed improvement will be worth the accuracy cost it will take by implementing this method. This then leads us to our next experiment which will be on the time it takes to decompose these pictures.

5.2.2 Time evaluation

In order to evaluate the time it takes to decompose an object into simpler shapes, the algorithms were timed at the times are recorded in table 5.2 below. As this was done on my own personal computer, this can only be considered as a guideline, and all of the experiments were done under the same conditions. It is important to note that this is not a fixed amount of time, and that there might be other factors that can affect the time as well. Thus, for the outcome of this dissertation, the time of each algorithm was recorded under the exact same circumstances as a means to compare two methods with each other. One note to make throughout this section is that the term *reduction* will be represented by a downwards facing arrow ($\downarrow$).

Table 5.2: Table showing the average % deviation(%) for the MNCD and the improved method from the ground truth experiment for the pictures of the MPEG-7 dataset.

Total Time to decompose shape (s)			
Picture	Ren (MNCD)	Improved Method	% improvement
Cow	63	30	110.3
Elephant	143	50	184.4
Dog	4	3	53.6
Mouse	16	11	49.0
Bug	68	113	-40.0
Cat	23	12	85.4
Horse	29	41	-30.5
Average	49	37	59

As can be seen, there is an average improvement of 12 seconds on the decomposition of the different images. This is about an average of 33% improvement and can in some cases be seen as a great improvement (e.g. the cow's decomposition) and some not (e.g. the bug's decomposition). It was found that in cases where the number of Harris corners detected is high - thus a object with a high number of corners, the decomposition tends to take longer than that of the MNCD method. Thus, on the time front, the improved MNCD method shows a slight improvement in the time it takes to decompose a shape.

5.2.3 Number of parts evaluation

The next hypothesis to test in order to evaluate the quality of decomposition, the number of parts produced after decomposition will be looked at. In order to evaluate that, the number of parts produced is simply counted and tabulated. In order to compare the results, the standard deviation of the ground truth experiment was used as a measure. In order to be classified as a good method, the number of parts must not lie outside of three deviations. Thus, as can be seen in table 5.3, the average standard deviation of the MNCD method is 1.1 where the improved method is about 5 - which in this case is not a good result. This means that on average the number of parts that this method produces will lie within 5 standard deviations from the ground truth mean compared to the 1.1 number of parts that the MNCD method produces.

Table 5.3: Table showing the number of parts that the same shapes are decomposed into for the different methods. The last column showing the percentage improvement.

Number of Parts						
Picture	Human Perception (average)	Standard deviation	Ren (MNCD)	Number of Deviations out	New	Number of Deviations out
Cow	7	2	7	0	15	4
Elephant	8	1	9	1	15	7
Dog	6	1	5	1	12	12
Mouse	8	2	9	1	10	1
Bug	10	3	18	3	15	2
Cat	7	1	7	0	12	5
Horse	7	2	10	2	14	4
Average	7.6	1.7	9.3	1.1	13.3	5.0

5.2.4 Invariance evaluation

In this section, the different invariances will be tested in order to compare the improved results with that of the MNCD results. As all of the methods are translation and scale invariant, only distortion and rotation invariance will be tested.

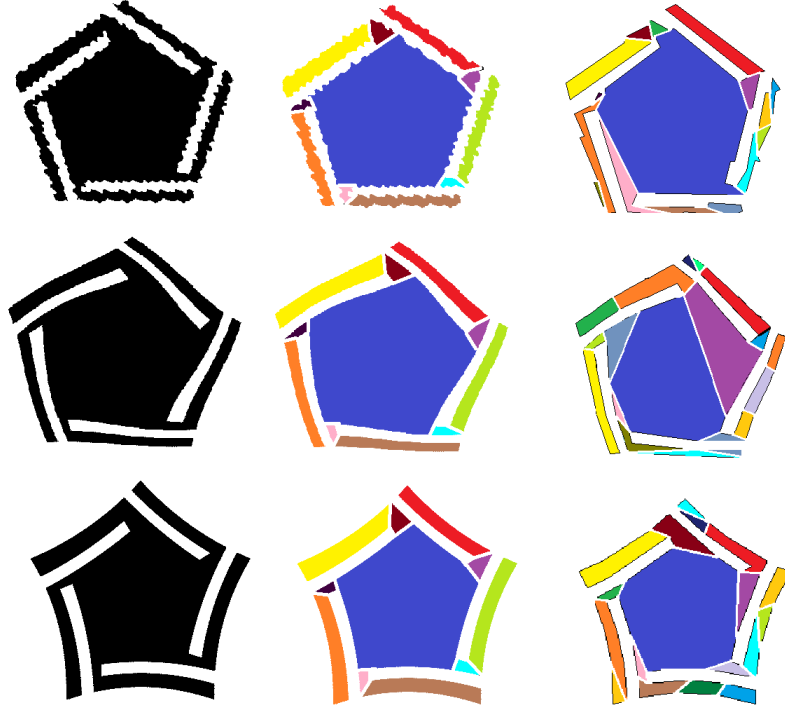


Figure 5.3: Figure showing the results of the MNCD algorithm (middle) and the improve algorithm (right) decomposition results after different distortion were applied to an object.

In figure 5.3, different distortions are applied to an object before processing and the results of the MNCD method is shown first while the improved method is shown second. As can be seen in the results, the different distortions produced the same results for the MNCD method, while for the improved method the results differ, thus indicating that this method is not distortion invariant.

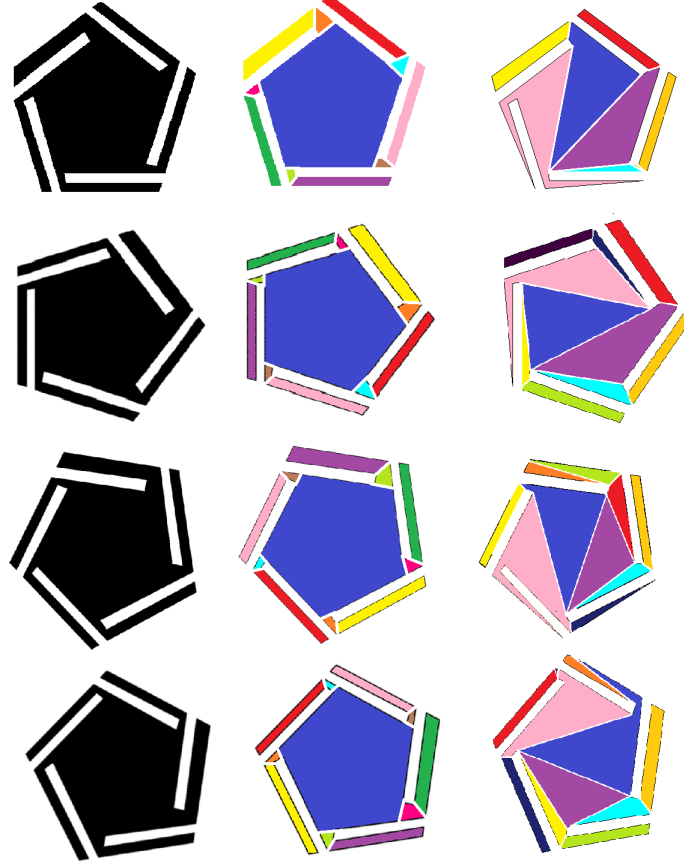


Figure 5.4: Figure showing the results of the MNCD algorithm (middle) and the improved algorithm (right) decomposition after an object has been rotated.

In figure 5.4 a rotation has been applied to the object before the image was given as an input to the MNCD and the improved algorithms. The object has been rotated 45° , 90° and 270° , and the results of the MNCD algorithm, and the improved method is shown above. As can be seen in the results, the MNCD algorithm gives the same results after every rotation, whereas the improved algorithm does not. Thus, this proves that the improved method is not rotation invariant.

Therefore in general, the improved method falls short when the invariance evaluation is applied to it. This is definitely an area of concern and future work will have to be done on the improved method to ensure that the algorithm will be invariant to rotation and distortion.

5.2.5 Parameter evaluation

In this subsection, the different parameters will be investigated and how they influence the results, if at all. In order to monitor the changes, each parameter will be changed and the accuracy, time and the number-of-parts produced will be looked at in order to determine the ideal values of the parameters. In the previous experiments, these values were kept constant as to have a fair experiment. The different parameters that will be looked at include the convexity threshold (ψ), the perception rule relation (β) and the visual naturalness (λ) parameters. The reason for looking

at these values are that these parameters have an influence on the MNCD results and thus the effects thereof should be investigated in the improved method.

ψ parameter

Accuracy evaluation

The ψ -parameter is used as the convexity threshold to allow an amount of convexity to be accepted. For this experiment, the parameter Ψ will be changed, while parameters λ and β will be kept constant, that is, $\lambda = 1$ and $\beta = 1$.

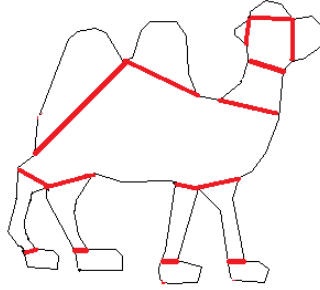


Figure 5.5: *Human shape decomposition result, also referred to as human ground truth, of a camel.*

The decomposition results are shown in figure 5.6. The human ground truth, or decomposition results are shown in figure 5.5. The results of the improved algorithm are shown in figure 5.6, for different values of ψ . As can be seen in figure 5.6, the red lines indicate cuts of the improved algorithm that agree to the human ground truth, while the black lines indicate cuts that do not agree to the human ground truth.

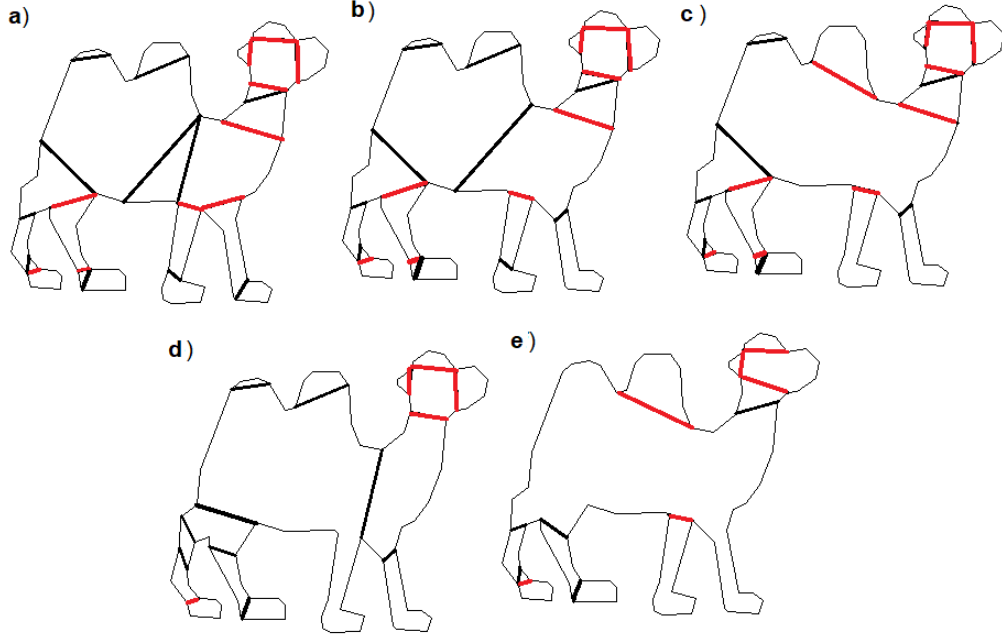


Figure 5.6: *Decomposition and accuracy in % results of Ψ -parameter, with (a) $\Psi = 0.01R$, (b) $\Psi = 0.3R$, (c) $\Psi = 0.5R$, (d) $\Psi = 0.75R$ and (e) $\Psi = 1R$, where R is the radius of the enclosing circle that surrounds the object and parameters $\beta = 1$, and $\lambda = 1$.*

The number of cuts produced as a result of the algorithm will be compared to figure 5.5 in order to obtain a percentage deviation of human ground truth, and thus an indication of accuracy. That is:

$$\%deviation_{hp} = \frac{|c_{hp} - c_{dm}|}{c_{hp}} \times 100 \quad (5.2)$$

Table 5.4: Table summarizing the results of changing the ψ -parameter value and how the accuracy results are influenced by changes of the ψ value .

Change of ψ - parameter vs. accuracy	
ψ value	% accuracy
0.01 R	66
0.30 R	60
0.50 R	66
0.75 R	33
1.00 R	40

As can be seen in figure 5.6 and in table 5.4, the lower the Ψ value, the more the number of parts there are and the higher the accuracy tends to be. It can also be noted that a small change can lead to different results. For example if you look at 5.6 (a) and (b), the Ψ value was changed from 0 to 0.01, and as can be seen there are fewer parts in (b) and the cuts also differ slightly

from (a). As found earlier, as the value of Ψ increases, the number of parts decrease along with the accuracy thereof. The number of part slowly decreases as Ψ increases. From the above results, it can be seen that the best value for Ψ is $0.01R \leq \Psi \leq 0.5R$, as the accuracy is still high, and the number of parts produced is not too many. The value used in the MNCD method was $\Psi = 0.03R$. As can be seen, the improved algorithm's value of Ψ falls within the boundaries chosen as well.

Time evaluation

In order to obtain an evaluation of the results, the time our method takes is recorded and the average time the MNCD-method takes was calculated. This was done because in Ren et al. [2] the average decomposition time was all that was given. From the given time, and the given time complexity, the average time (in seconds) could be determined. It is important to note that this is an average time based on time complexity, and will be different under different circumstances.

In order to calculate the average reduction in time, the following formula was used:

$$time \downarrow = \frac{time_{MNCD} - time_{NEW}}{time_{MNCD}} \times 100 \quad (5.3)$$

It is important to note that this time is in percentage (%). As can be seen in table 5.5 some of the pictures show a large improvement for example the picture of the cow. In the bug picture it can be seen that there is a incline in time reduction. The reason for this might be that there is less noise and thus less vertices in the picture of the horse than what there is in the picture of the bug - thus causing the DCE algorithm to take more time to evolve the bug than the horse. On average, it can be seen that the time was reduced by a total average of 59.65 %.

Table 5.5: Table showing the average reduction (%) in time for the pictures of the MPEG-7 dataset with $\lambda = 1$ and $\beta = 1$.

Total time reduction with change of ψ - parameter (%)					
Picture	$\psi = 0.005 R$	$\psi = 0.05 R$	$\psi = 0.1 R$	$\psi = 0.2 R$	$\psi = 0.5 R$
Cow	110.43	110.03	110.30	110.95	111.64
Elephant	199.62	199.08	184.40	199.36	199.70
Dog	55.00	52.42	53.60	55.50	55.81
Mouse	48.98	45.64	49.00	47.41	48.22
Bug	-40.53	-40.80	-40.00	-40.12	-39.46
Cat	75.50	76.86	85.40	77.75	76.35
Horse	-30.35	-30.17	-30.50	-29.92	-29.32
Average	59.81	59.01	58.89	60.13	60.42

In order to determine a relationship between time and Ψ , it can be seen from table 5.5 that the average time first declines, then increases. Although these inclines and declines are small amounts, out of the values chosen in the table above, it can be seen that the value of ψ should be chosen around $\psi = 0.5 \times R$. Here R indicates the radius of the bounding circle of the object at hand. This is used as to keep the value relative to the size of the object. It can also be seen that this parameter has a very small influence on the time it takes to decompose a shape, and it can thus

be concluded that the ψ -parameter has little to no influence on the time it takes to decompose a shape.

Number of parts evaluation

In this section the parameter Ψ is looked at and evaluated according to the number of parts produced by the improved algorithm after shape decomposition. That is, Ψ is set to 0.005, 0.05, 0.1, 0.2 and 0.5 and the number of parts after decomposition is then determined and recorded. This is done for the same 7 pictures of the MPEG-7 image dataset pictures that has been used in all of the experiments. The results of this experiment are shown in table 5.6.

Table 5.6: Table showing the total number of parts (NOP) for the pictures of the MPEG-7 dataset with $\lambda = 1$ and $\beta = 1$.

Total number of parts with change of ψ - parameter					
Picture	$\psi = 0.005 R$	$\psi = 0.05 R$	$\psi = 0.1 R$	$\psi = 0.2 R$	$\psi = 0.5 R$
Cow	16	16	15	14	12
Elephant	15	15	15	15	12
Dog	12	12	12	11	9
Mouse	10	10	10	10	8
Bug	16	16	15	15	13
Cat	12	12	12	12	12
Horse	15	14	14	13	10
Average	14	13	13	13	11

As can be seen in the table, the average number of parts does not show a drastic increase or decrease as the values of ψ is changed. It can however be said that the larger the ψ -parameter the less parts will be produced after decomposition indicating that this parameter will only influence the results when the parameter is given very large values. Thus, for a time improvement, keeping the ψ -parameter at values between $0.05R \leq \psi \leq 0.2R$ will produced the fastest results with more or less the same number of parts than experiments done earlier.

λ parameter

Accuracy evaluation

The λ -parameter can be classified as a visual naturalness parameter. Next the effects of the λ parameter will be compared. For this experiment the λ values will be changed while the parameter $\beta = 1$ and parameter $\Psi = 0.5R$ stays constant. Comparing the results to figure 5.5, and calculating the accuracy using equation 5.2, the decomposition results are shown in 5.7 below.

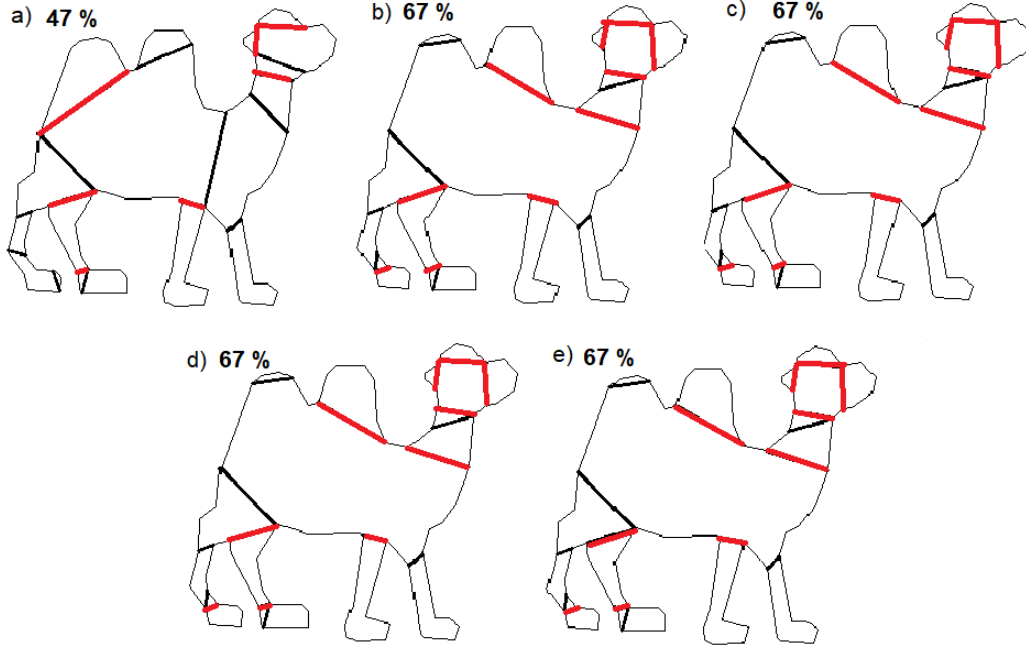


Figure 5.7: *Decomposition results and accuracy in % of λ -parameter, with (a) $\lambda = 0$, (b) $\lambda = 0.25$, (c) $\lambda = 0.5$, (d) $\lambda = 0.75$ and (e) $\lambda = 1$ with $\beta = 1$, and $\Psi = 0.5R$.*

As can be seen, the effects of the λ parameter is not as spectacular as that of the Ψ parameter. In figure 5.7, pictures a and b show some differences in cuts selection, whereas the last three pictures c to e show no change in cut selection. Thus a suitable lambda value for λ would be values that are $0.25 \leq \lambda \leq 1$. One last note to make is to realize that as the value of lambda increases, the number of parts stays constant in this specific picture. It is important to note that the cuts differ with some small changes and with smaller values of λ . With larger λ values the cuts stay the same.

Time evaluation

To be able to see the effects this parameter has, the same 7 images of the MPEG-7 dataset will be used as before and the time and number of parts produced after decomposition will be looked at. The time reduction formula that was used is the same as equation 5.3. As can be seen in table 5.7, the amount of time that the decomposition takes place with different lambda values are shown in percentages.

Table 5.7: Table showing the average reduction (%) in time for the pictures of the MPEG-7 dataset with $\Psi = 0.5R$ and $\beta = 1$.

Total time reduction with change of λ - parameter (%)					
Picture	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
Cow	107.81	106.03	110.30	109.97	109.74
Elephant	184.64	184.32	184.40	184.32	170.39
Dog	54.11	53.42	53.60	53.60	51.89
Mouse	48.27	49.30	49.00	49.89	49.73
Bug	-40.11	-41.01	-40.00	-39.76	-40.05
Cat	83.05	82.51	85.40	85.44	93.05
Horse	-29.12	-29.47	-30.50	-29.65	-31.97
Average	58.38	57.87	58.89	59.12	57.54

The average amount of percentage improvement can be given as 58.56%. Thus showing that the λ -parameter produces the fastest results between values $0.5 \leq \lambda \leq 0.75$. The change in values are so small that it can be neglected, which places the λ parameter specified in Ren *et al.* [2] with value between $0 \leq \lambda \leq \frac{1}{\sum_{i=1}^n w_i}$ into perspective.

Number of parts evaluation

In this section the parameter λ will be looked at and evaluated according to the number of parts. That is, λ was set to 0, 0.25, 0.5, 0.75 and 1 and the number of parts after decomposition was then determined and recorded. This was done for 8 pictures of the MPEG-7 image dataset pictures. The results of this experiment are shown in table 5.8.

Table 5.8: Table showing the total number of parts (NOP) for the pictures of the MPEG-7 dataset while changing the λ parameter, with $\Psi = 0.1R$ and $\beta = 1$.

Total number of parts with change of λ - parameter					
Picture	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
Cow	16	15	15	15	15
Elephant	15	15	15	15	15
Dog	12	12	12	12	12
Mouse	10	10	10	10	10
Bug	14	15	15	15	15
Cat	14	12	12	12	12
Horse	13	14	14	14	14
Average	13	13	13	13	13

As can be seen in the table, there is no change in the average number of parts produced for the different values of λ , and thus can be concluded that this parameter has no influence on the

number of parts produced after decomposition.

β parameter

Accuracy evaluation

The β -parameter is used as the perception rule relation parameter. That is, β is used to describe the relationship between the perception rules. Next the β -parameter changes will be monitored. Here the values of β will be changed while the parameters $\Psi = 0.5R$, and $\lambda = 0.5$ stay constant. The results are shown in figure 5.8. Comparing the results to figure 5.5, and calculating the accuracy using equation 5.2, the decomposition results are shown in figure 5.7 below.

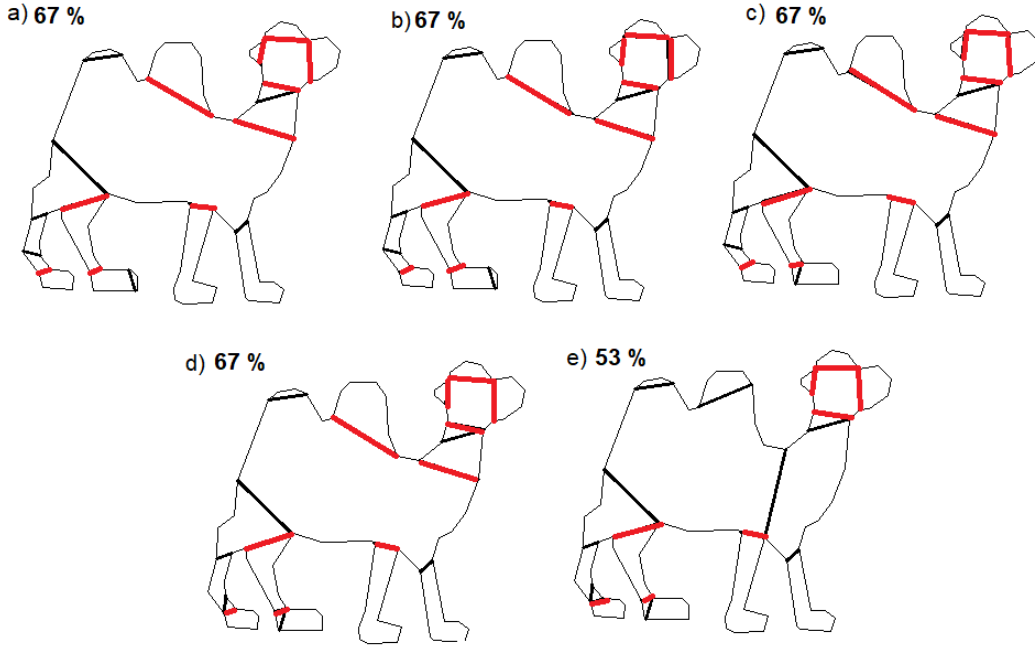


Figure 5.8: *Decomposition results and accuracy in % of β -parameter with (a) $\beta = 0$, (b) $\beta = 0.25$, (c) $\beta = 0.5$, (d) $\beta = 1$ and (e) $\beta = 3$ with $\lambda = 0.5$, and $\Psi = 0.5R$.*

As can be seen in figure 5.8, there is little change in the decomposition when the β parameter is changed. It can therefore be deduced that this parameter have little to no effect on the decomposition or the accuracy when the value of β is small. Thus for best accuracy results, β should be $0 \leq \beta \leq 1$.

Time evaluation

For this experiment, β was changed to values of 0.025, 0.5, 1, 1.5 and 3 respectively, and then the time it takes to decompose the picture was used to determine the time reduction percentage. This can be determined by using equation 5.3. The results are shown in table 5.9.

Table 5.9: Table showing the average reduction (%) in time for the pictures of the MPEG-7 dataset with $\Psi = 0.5R$ and $\lambda = 0.5$.

Total time reduction with change of β - parameter (%)					
Picture	$\beta = 0.025$	$\beta = 0.5$	$\beta = 1$	$\beta = 1.5$	$\beta = 3$
Cow	110.50	110.57	110.30	110.81	110.68
Elephant	199.42	199.08	184.40	199.16	199.45
Dog	55.49	55.00	53.60	54.12	55.35
Mouse	46.13	48.87	49.00	48.78	49.00
Bug	-41.05	-40.55	-40.00	-40.28	-40.28
Cat	77.53	78.42	85.40	76.94	76.15
Horse	-30.31	-30.77	-30.50	-31.09	-31.89
Average	59.67	60.09	58.89	59.78	59.78

As can be seen, the total average time reduction is 59.64%. It can also be seen that for some pictures there is an improvement in time reduction and in some there is a decline in time reduction.

Number of parts evaluation

In this section the parameter β will be looked at and evaluated according to the number of parts. That is, β was set to 0.025, 0.5, 1, 1.5 and 3 and the number of parts after decomposition was then determined and recorded. This was done for the same several images of the MPEG-7 image dataset pictures. The results of this experiment are shown in table 5.10.

Table 5.10: Table showing the total number of parts (NOP) for the pictures of the MPEG-7 dataset while changing the β parameter, with $\Psi = 0.5R$ and $\lambda = 0.5$.

Total number of parts with change of β - parameter					
Picture	$\beta = 0.025$	$\beta = 0.5$	$\beta = 1$	$\beta = 1.5$	$\beta = 3$
Cow	15	15	15	15	15
Elephant	15	15	15	15	15
Dog	12	12	12	12	12
Mouse	10	10	10	10	10
Bug	15	15	15	15	15
Cat	12	12	12	12	14
Horse	14	14	14	14	14
Average	13	13	13	13	14

As can be seen in the table, the average number of parts increases by one when $\beta = 3$. Thus, it can be said that for larger values of β the number of parts will increase, but looking at the results in the table, it would be safe to say that β has little to no influence on the number of parts that have been produced.

Combined parameter evaluation

In this experiment the different parameter will be changed to see how these combined effects change the decomposition and accuracy results. Here the values of the parameters will be as follows: $0R \leq \Psi \leq 0.3R$, $-0.25 \leq \lambda \leq 0.25$ and $\beta = 1$. This will result in nine decomposed pictures, and is shown with the accuracy in % in figure 5.9. The summary of the different parameter values for each decomposed picture is shown in table 5.11. This experiment is done to determine the best values that might be used for decomposition. These values are adapted to the picture of the camel, and might differ from the other pictures.

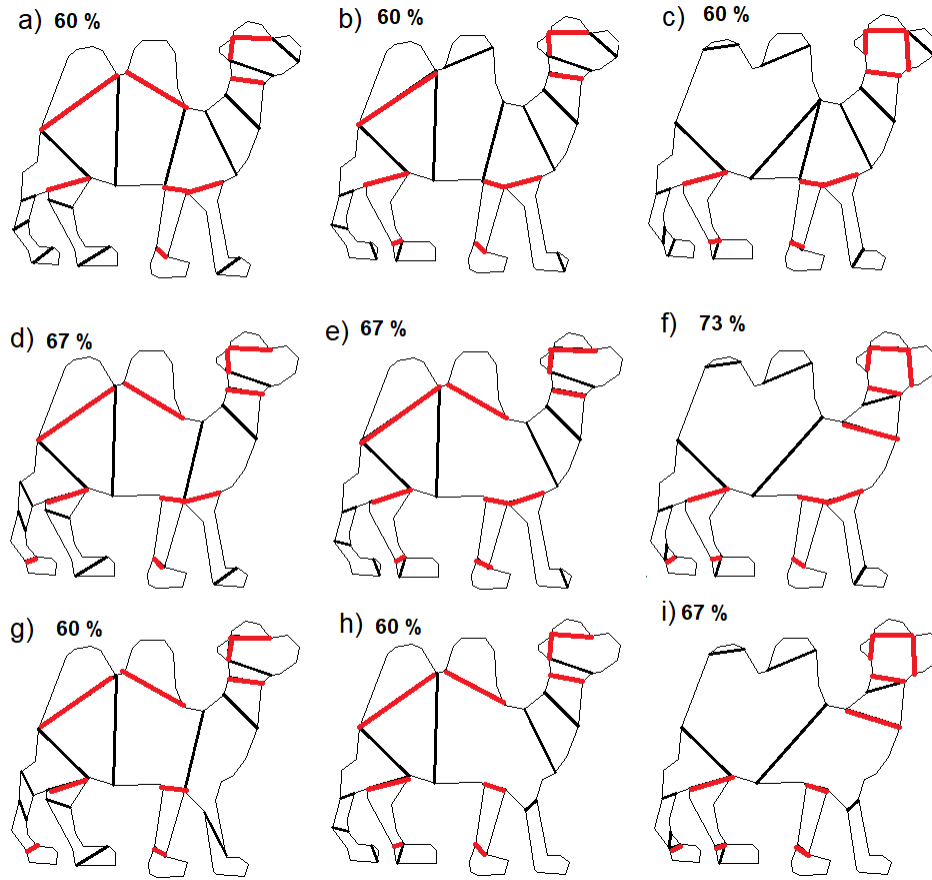


Figure 5.9: *Decomposition results of multi-parameters*

Table 5.11: Table showing the parameter values for each decomposed picture in figure 5.9.

Picture	ψ - values	λ - values	β - values	Accuracy (%)
a	0	-0.25	1	60
b	0	0	1	60
c	0	0.25	1	60
d	0.15	-0.25	1	67
e	0.15	0	1	67
f	0.15	0.25	1	73
g	0.3	-0.25	1	60
h	0.3	0	1	60
i	0.3	0.25	1	67

As can be seen in the results, the decomposition of either e or f looks the most natural. That is for values $\Psi = 0.15$, $0 \leq \lambda \leq 0.25$ and $\beta = 1$. In picture (e) the body of the camel is more precisely decomposed in terms of the ground truth. In (f), the camel feet and facial area are more precisely decomposed. Thus, looking at the accuracy percentages, (f) produces more accurate results, and these values were chosen to decompose the objects into simpler shapes. The decomposition results are shown in figure 5.10.

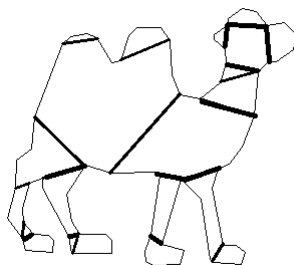


Figure 5.10: *Decomposition results of parameters $\Psi = 0.15R$, $\lambda = 25 \times 10^{-7}$, and $\beta = 1$.*

This results is close to natural decomposition in the sense that the head, legs, feet, humps and the body of the camel are all different parts. There are however some redundant cuts that could've been left out, for example by the neck, the middle cut is not necessary. And by the toes and the one heel there are also redundant cuts. This can also be as the result of using only a few vertices instead of all of them.

Now that the effects of the different parameters have been determined and the ideal values of these parameters have been found, the results of the improved method can be compared to the other methods, which will be discussed in the next section.

5.2.6 Comparison to other algorithms

In this subsection, the improved method will be compared to the other methods, as was done in chapter 2. The comparison will be done in the same way as was done before and the results are summarized in table 5.12 below.

Table 5.12: Table showing the evaluation results of the different shape decomposition algorithms 5.9.

	Method Values					
	Wang (PFSD)	Ren (MNCD)	Liu (CSD)	Lien (ACD)	Lu (AD)	New
Meaningful	3	4	0	3	2	2
Number of Parts	4	3	2	1	3	1
Time Complexity	2	0	1	3	4	2
Invariance	4	4	2	2	4	2
Type	2	4	4	4	0	4
Score	15	15	9	13	13	11

Thus, as can be seen in the table, the improved algorithm has made some improvements in some areas, while in other areas it did not. As can be seen, in time complexity the method improved quite significantly compared to the MNCD method, where in meaningfulness, number of parts and invariance it performs much like the other methods used to compare with. The decomposed results of the different methods are shown in figure 5.11.

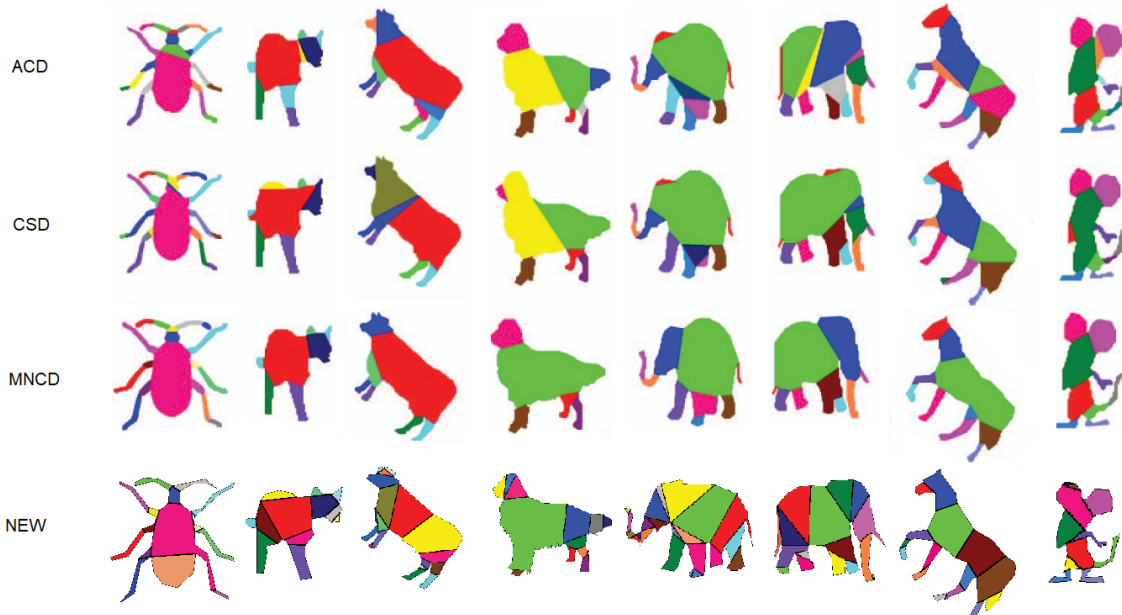


Figure 5.11: *Decomposition comparison with methods ACD, CSD, MNCD and our method. Parameters are set to $\Psi = 0.15R$, $\lambda = 0.25$, and $\beta = 1$.*

Here it can also be seen that the improved algorithms' decomposition results are not close to that of the ground truth. It can also be seen that the accuracy has declined from that of the MNCD algorithm.

5.3 Simple shape output results

In this section the simple shape output will be evaluated. Recall from section 4.4.5, that after shape decomposition is done, the parts will be simplified into more primitive shapes as to make classification of an object easier. Thus, simple shape output refers to the output after the complex shape has been decomposed into parts, and then these parts are simplified. For this experiment, the output of the different shapes will be looked at and evaluated in terms of the number of corners, and if the shape has been correctly identified.

For the shapes to be identified correctly another parameter, σ , is used. This parameter is used to determine the maximum distance from the contour at hand to the approximated contour and can be found in the last step of the improved algorithm as a parameter for the simplification of parts. This concept is illustrated in figure 5.12. In (a) σ is set to 10% of the perimeter of the shape and drawn in green is the approximate contour. In (b) σ is now 5% of the perimeter and in (c) its set to 1% of the perimeter. The resultant approximation contours are shown in (b) blue and (c) pink.

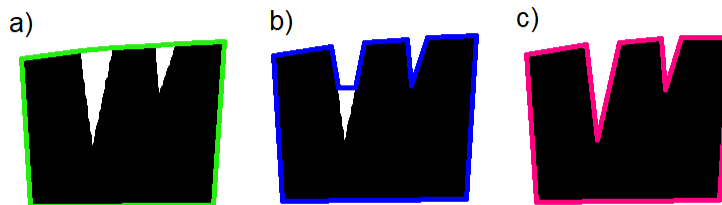


Figure 5.12: *Approximate contour results with changes in σ . In (a) $\sigma = 10\%$, (b) $\sigma = 5\%$ and (c) $\sigma = 1\%$ of the perimeter of the shape.*

If this value is chosen correctly, the simple shape output that is determined will be accurate and correct. In order to determine the optimal value, this σ value will be changed and the output is shown in figure 5.13. The parameter has been chosen as $\sigma = 0.006$, $\sigma = 0.01$, $\sigma = 0.025$, $\sigma = 0.03$, $\sigma = 0.06$ and $\sigma = 0.01$.

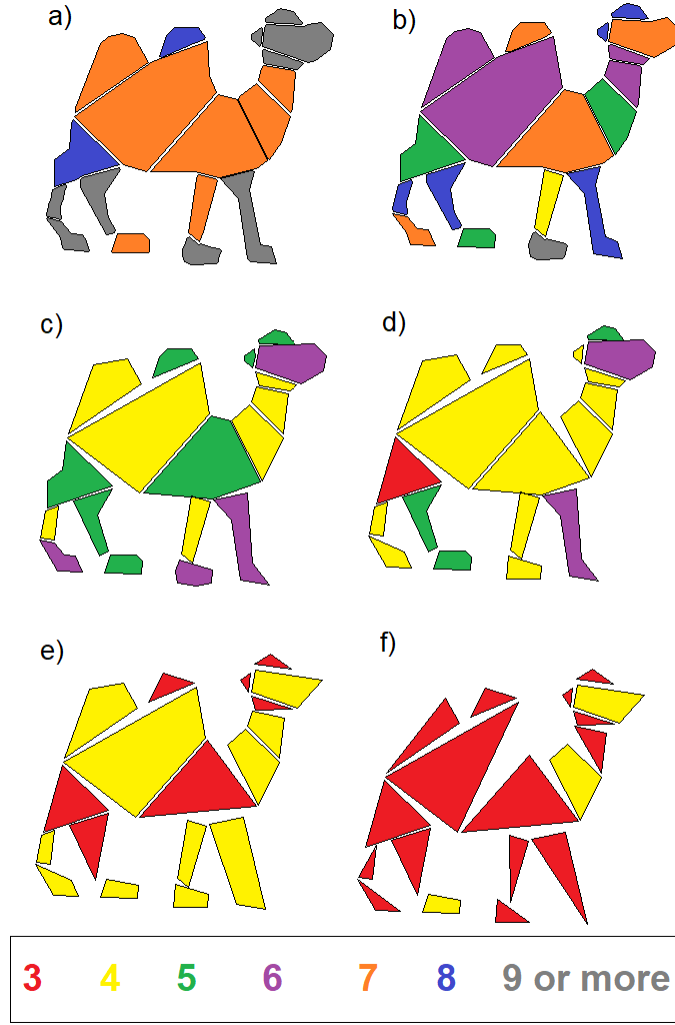


Figure 5.13: Simple shape output after σ has been changed to: (a) $\sigma = 0.006$, (b) $\sigma = 0.01$, (c) $\sigma = 0.025$, (d) $\sigma = 0.03$, (e) $\sigma = 0.06$ and (f) $\sigma = 0.1$

As can be seen in the figure, the legend is used to colour code the shape present in the decomposition result. For example, red is used to colour triangles, yellow for quadrilaterals, green for pentagons etc. As can be seen in the figure, the smaller the value of σ , the more accurate the simple shapes represent the original decomposed image. Thus, the larger σ creates more triangular approximation of the original image, but at the same time loses the effect to accurately still be able to tell what the picture represents.

Therefore looking at the different output values, either (c) or (d) can be used to represent the decomposed images in simple shape form. These two pictures represent the camel the best, without having any shapes that have more than six sides.

5.3.1 Conclusion

To conclude this chapter a short summary of the experiments and their results will be given. After all of the experiments was done, it was found that there is a direct proportional relation between time reduction and the Ψ parameter. This also means that is is inversely proportional to the time the method takes to solve the problem. It was also found that the λ and β parameters have no influence on the time it takes to decompose a shape. Furthermore, the complete opposite was found when it comes to the number of parts experiments. That is, it was found that the NOP is directly proportional to the time reduction, and thus inversely proportional to the time it takes for the method to decompose the shape. As before, it was found that λ and β has little to no influence on the NOP after decomposition. It was also found that, after comparing the improved method with the other methods, that an average of $\pm 58\%$ time reduction was achieved, but at the same time, the NOP increased. Thus there are more improvements that can be applied to this method, and will be discussed in chapter 6.

Chapter 6

Conclusion

6.1 Conclusion

The focus of this dissertation was to make an improvement on the time it takes to decompose a complex shape while the shape decomposition results, in terms of accuracy and the number of parts after decomposition, are kept the same. Thus previously proposed methods were looked at and compared in order to identify the method that needs the most improvement and at the same time should also be able to separate a complex shape into the minimal number of simple parts.

This was done by doing some research and then identify areas where shape decomposition plays an important role. After the research was done, a research question was formed and a literature review on what shape decomposition is, the different types of shape decomposition bases, some of the shape descriptors that is found and the time complexity of some decomposition methods, as well as the decomposition results of those methods, was done. After the different types of shape decomposition algorithms were compared, the MNCD method was identified as a method to be able to improve on.

It was found that the method best to improve on is the MNCD method, as it has a high time complexity and decomposed complex shapes into the minimal number of simple shapes while having high visually naturalness. As this method makes use of Morse functions to solve, an alternative to these functions has been proposed. Instead of making use of all the vertices of the contours as well, Harris Corner Detection (HCD) was used to make use of corners instead. But because HCD contains a lot of variables, Discrete Contour Evolution (DCE) was proposed to obtain fewer vertices and thus less cuts to decompose a shape. The concept of Mutex pairs was also implemented from Ren *et al.* [2] and different methods of how to determine mutex pairs are investigated.

Next some experiments were done, and the results were analysed. For this dissertation, the results were analysed according to the time improvement on the MNCD method, and an average of $\pm 58\%$ improvement was found. As to the number of parts, the accuracy and the invariance it was found that there was not a significant improvement, and in some cases the results could have been better.

6.2 Future improvements

As mentioned in chapter 5, the results of the time it takes to decompose a shape is improved by an average of $\pm 58\%$. Another way to evaluate the result is to draw up a time complexity function. In

order to implement the DCE, we know that the time complexity is: $O(N\log N)$, here the number of vertices in the original shape is represented by N . Then, to obtain the $C(S)$ will take $O(v^2)$, where v is the number of corners after Harris-Corner detection is implemented. In order to compute the number of mutex pairs, the time complexity is: $O(vr)$, where the number of notches are represented by r . In computing the $\mathbf{A}$ matrix $O(mn)$ is used where the number of mutex pairs is represented by m and n the number of cuts. But because $N \gg v, r, m$ and n , we can write the time complexity as: $O(N\log N + v^2 + vr + mn) = O(N\log N)$.

The first and most important area to improve on is the accuracy. If the accuracy can be kept the same as the MNCD-method, this method will be very useful in many areas. Thus, in order to improve on accuracy, some mathematical improvements can be made to the DCE, Harris corner and the Delaunay triangulation detection algorithms. Although standard, some of these algorithms can be tweaked in order to produce different results, and as such, more accurate results. Thus, for future work, some in depth study if the different algorithms can be done in order to identify areas of improvement and to get results closer to human perception.

The next possible improvement that can be brought up is an improvement on how to determine the Mutex pairs. The Morse functions take up a lot of time, and using only the corner points is too simple, and thus the mutex pairs can have a great influence on the decomposition results. A suggestion that I would make is to obtain a medial axis, and through doing that, obtain mutex areas, instead of points, that will describe possible areas of decomposition, and any point closer to a midpoint of an area, but be described accordingly. Another suggestion might be to look at machine learning algorithms in order to identify areas that can be classified as mutex areas and in such a way improve the speed of decomposition over time.

Another improvement that can be implemented, is invariant improvements, as the experiment results show that this method is not distortion invariant. In order to improve on these properties, more research needs to be done on why they are invariant and how to ensure that it will stay invariant.

List of references

- [1] Y. Lu, J. M. Lien, M. Ghosh, and N. M. Amato, “ α -Decomposition of polygons,” *Computers and Graphics (Pergamon)*, vol. 36, no. 5, pp. 466–476, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.cag.2012.03.018>
- [2] Z. Ren, J. Yuan, and W. Liu, “Minimum near-convex shape decomposition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2546–2552, 2013.
- [3] M. Sonka, V. Hlavac, and R. Boyle, “Image Processing, Analysis, and Machine Vision,” *Thomson Learning*, p. 555, 2008.
- [4] R. J. Qureshi, J.-Y. Ramel, and H. Cardot, “Graph based Shapes Representation and Recognition,” in *Lecture Notes in Computer Science*, 2007, vol. 4538/2007, pp. 49–60. [Online]. Available: <https://projet.liris.cnrs.fr/m2disco/pub/Congres/2007-GBR/papers/poster1.pdf>
- [5] J. Ahmad, M. Sajjad, I. Mehmood, S. Rho, and S. Wook, “Describing Colors , Textures and Shapes for Content Based Image Retrieval A Survey,” *JOURNAL OF PLATFORM TECHNOLOGY*, vol. 2, no. 4, pp. 34–48, 2014.
- [6] J. Zeng, R. Lakaemper, X. Yang, and X. Li, “2D shape decomposition based on combined skeleton-boundary features,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5359 LNCS, no. PART 2, pp. 682–691, 2008.
- [7] J. De Winter and J. Wagemans, “Segmentation of object outlines into parts: A large-scale integrative study,” *Cognition*, vol. 99, no. 3, pp. 275–325, 2006.
- [8] I. Pitas and A. N. Venetsanopoulos, “Morphological Shape Decomposition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 38–45, 1990.
- [9] J.-M. Lien, J. Keyser, and N. M. Amato, “Simultaneous shape decomposition and skeletonization,” *Proceedings of the 2006 ACM symposium on Solid and physical modeling - SPM '06*, p. 219, 2006. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1128888.1128919>
- [10] C. Wang, W. Liu, Z. Lai, and H. Wang, “Perceptually friendly shape decomposition by resolving segmentation points with minimum cost,” *Journal of Visual Communication and Image Representation*, vol. 24, no. 3, pp. 270–282, 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.jvcir.2013.01.001>

- [11] Y. Lu, J. M. Lien, M. Ghosh, and N. M. Amato, " α -Decomposition of polygons," *Computers and Graphics (Pergamon)*, vol. 36, no. 5, pp. 466–476, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.cag.2012.03.018>
- [12] H. Liu, W. Liu, and L. J. Latecki, "Convex shape decomposition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 97–104, 2010.
- [13] J. Page, "Convex Polygon Definition - Math Open Reference," 2011. [Online]. Available: <https://www.mathopenref.com/polygonconvex.html>
<http://www.mathopenref.com/polygonconvex.html>
- [14] C. Wang, W. Liu, Z. Lai, and H. Wang, "Perceptually friendly shape decomposition by resolving segmentation points with minimum cost," *Journal of Visual Communication and Image Representation*, vol. 24, no. 3, pp. 270–282, 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.jvcir.2013.01.001>
- [15] H. Liu, W. Liu, and L. J. Latecki, "Convex shape decomposition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. May 2014, pp. 97–104, 2010.
- [16] J. M. Lien and N. M. Amato, "Approximate convex decomposition of polyhedra and its applications," *Computer Aided Geometric Design*, vol. 25, no. 7, pp. 503–522, 2008.
- [17] T. Jiang, Z. Dong, C. Ma, and Y. Wang, "Toward perception-based shape decomposition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7725 LNCS, no. PART 2, pp. 188–201, 2013.
- [18] G. Mallas and E. Bas, "Computer Vision Project 1," *Methodology*, no. 1, pp. 1–15, 2009. [Online]. Available: https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj2/html/lyi33/
- [19] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed., M. Loukides, Ed. Sebastopol: O'Reilly Media, 2008.
- [20] J. A. Suykens, "Introduction to Machine Learning," in *Introduction to Machine Learning*. London: MIT Press, 2014, pp. 765–773. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/B9780123965028000139>
- [21] G. Liu, Z. Xi, and J. M. Lien, "Dual-space decomposition of 2D complex shapes," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4154–4161, jun 2014. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909925>
- [22] C. Ma, Z. Dong, T. Jiang, Y. Wang, and W. Gao, "A method of perceptual-based shape decomposition," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 873–880, 2013.
- [23] D. D. Hoffman and M. Singh, "Salience of visual parts," *Cognition*, vol. 63, no. 1, pp. 29–78, 1997.

- [24] J. Erickson and S. Har-Peled, “Optimally Cutting a Surface into a Disk,” in *Discrete and Computational Geometry*, vol. 31, no. 1. New York, New York, USA: ACM Press, 2004, pp. 37–59. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=513400.513430>
- [25] Z. Ren, J. Yuan, C. Li, and W. Liu, “Minimum near-convex decomposition for robust shape representation,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 303–310, 2011.
- [26] C. Wang, Z. Lai, and H. Wang, “Hand gesture recognition based on perceptual shape decomposition with a Kinect camera,” *IEICE Transactions on Information and Systems*, vol. E96-D, no. 9, pp. 2147–2151, 2013.
- [27] M. Singh, G. D. Seyranian, and D. D. Hoffman, “Parsing silhouettes: The short-cut rule,” *Perception and Psychophysics*, vol. 61, no. 4, pp. 636–660, 1999.
- [28] N. Papanelopoulous, “Planar shape decomposition made simple,” *Bmvc2015*, pp. 1–12, 2015.
- [29] K. Leonard, G. Morin, S. Hahmann, and A. Carlier, “A 2D shape structure for decomposition and part similarity,” in *Proceedings - International Conference on Pattern Recognition*, 2017, pp. 3216–3221.
- [30] I. Biederman, “Recognition by components: A theory of human image understanding,” *Psychological Review*, vol. 94, no. 2, pp. 115–117, 1987.
- [31] A. V. Naik, K. W. Regan, and D. Sivakumar, “On Quasilinear Time Complexity Theory,” SUNY - Buffalo, Tech. Rep., 1999. [Online]. Available: <https://cse.buffalo.edu/regan/papers/pdf/NRS95.pdf>
- [32] M. Keil and J. Snoeyink, “on the Time Bound for Convex Decomposition of Simple Polygons,” *International Journal of Computational Geometry & Applications*, vol. 12, no. 03, pp. 181–192, 2002.
- [33] H. Rom and G. Médioni, “Hierarchical Decomposition and Axial Shape Description,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 973–981, 1993.
- [34] F. Aurenhammer, “Weighted skeletons and fixed-share decomposition,” *Computational Geometry: Theory and Applications*, vol. 40, no. 2, pp. 93–101, 2008.
- [35] S. Loncaric, “A survey of shape analysis techniques,” University of Zagreb, Zagreb, Tech. Rep. 8, 1998.
- [36] C. Martinez-Ortiz, “2D and 3D Shape Descriptors,” *Eric.Exeter.Ac.Uk*, no. 12, pp. 1–14, 2010.
- [37] O. E. Badawy and M. Kamel, “Matching Concavity Trees,” University of Waterloo, Waterloo, Tech. Rep., 2004.
- [38] G. Liu, Z. Xi, and J. M. Lien, “Dual-space decomposition of 2D complex shapes,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. June 2014, pp. 4154–4161, 2014.

- [39] Y. Mingqiang, K. Kidiyo, and R. Joseph, "A Survey of Shape Feature Extraction Techniques," *Pattern Recognition Techniques, Technology and Applications*, 2008.
- [40] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, no. 1, pp. 1–19, 2004.
- [41] E. R. Davies, *Computer Vision, Fifth Edition: Principles, Algorithms, Applications, Learning*, 5th ed. London: Academic Press, 2018.
- [42] P. Blanke, "Medial axis," pp. 27–28, 2009. [Online]. Available: https://commons.wikimedia.org/wiki/File:Medial_axis_example_2d.svg
- [43] S. Velasco-forero, S. V.-f. I.-c. Opening, and A. M. Morphology, "Inner-Cheeger Opening and Applications To cite this version : HAL Id : hal-01250280," Mines Paris Tech, Paris, Tech. Rep. May 2015, 2016.
- [44] E. H. Cohen and M. Singh, "Geometric determinants of shape segmentation: Tests using segment identification," *Vision Research*, vol. 47, no. 22, pp. 2825–2840, 2007.
- [45] S. Bouagar and S. Larabi, "Discriminative outlines parts for shape retrieval," *Journal of Visual Communication and Image Representation*, vol. 33, pp. 149–164, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.jvcir.2015.08.019>
- [46] M. R. Daliri and V. Torre, "Classification of silhouettes using contour fragments," *Computer Vision and Image Understanding*, vol. 113, no. 9, pp. 1017–1025, 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.cviu.2009.05.001>
- [47] T. Biedl, M. Held, S. Huber, D. Kaaser, and P. Palfrader, "Reprint of: Weighted straight skeletons in the plane," *Computational Geometry: Theory and Applications*, vol. 48, no. 5, pp. 429–442, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.comgeo.2015.01.004>
- [48] Y. Lu, "Path Planning in Similar Environments," Ph.D. dissertation, Dalian University of Technology, 2013.
- [49] D. Avis and G. T. Toussaint, "An efficient algorithm for decomposing a polygon into star-shaped polygons," *Pattern Recognition*, vol. 13, no. 6, pp. 395–398, 1981.
- [50] L. J. Latecki and R. Lakämper, "Convexity Rule for Shape Decomposition Based on Discrete Contour Evolution," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 441–454, 1999.
- [51] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 35, no. 1, pp. 13–32, 1999.
- [52] K. Siddiqi and B. Kimia, "Parts of Visual Form: Computational Aspects," Providence, pp. 75–81, 1993.
- [53] C. Aslan, A. Erdem, E. Erdem, and S. Tari, "Disconnected skeleton: Shape at its absolute scale," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 12, pp. 2188–2203, 2008.

- [54] X. Jianning, “MORPHOLOGICAL DECOMPOSITION OF 2 D BINARY SHAPES INTO MODESTLY OVERLAPPED DISK COMPONENTS (TueAmPO6) Menu,” *IEEE*, p. 6, 2005.
- [55] A. Sharma, “Time and Space Complexity Tutorials & Notes — Basic Programming — HackerEarth,” 2016. [Online]. Available: <https://www.hackerearth.com/practice/basic-programming/complexity-analysis/time-and-space-complexity/tutorial/>
- [56] M.-K. Hu, “Visual {P}attern {R}ecognition by {M}oment {I}nvariants,” *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 179–187, 1962. [Online]. Available: <http://www.sci.utah.edu/gerig/CS7960-S2010/handouts/Hu.pdf>
- [57] M. Krause, “What is translation invariance in computer vision and convolutional neural network?” 2016. [Online]. Available: <https://stats.stackexchange.com/questions/208936/what-is-translation-invariance-in-computer-vision-and-convolutional-neural-netwo>
- [58] L. D. Stroebe and R. D. Zakia, “The Focal encyclopedia of photography,” *Choice Reviews Online*, vol. 31, no. 04, pp. 31–1855–31–1855, 2013.
- [59] L. Alvarez, L. Gomez, and J. R. Sendra, “Algebraic Lens Distortion Model Estimation,” *Image Processing On Line*, vol. 1, 2010. [Online]. Available: <http://dx.doi.org/10.5201/ipol.2010.ags-alde>
- [60] S. L. Miller and D. Childers, “Numerical Methods for Evaluating the Q-Function,” in *Probability and Random Processes*, 2012, pp. 587–591. [Online]. Available: <http://elsevier.com>
- [61] M. Sadeqi, R. C. Holte, and S. Zilles, “Using Coarse State Space Abstractions to Detect Mutex Pairs,” pp. 1–8, 2013. [Online]. Available: <papers://d471b97a-e92c-44c2-8562-4efc271c8c1b/Paper/p143>
- [62] Jérôme Lang, “International Joint Conferences on Artificial Intelligence (IJCAI),” in *International Joint Conferences on Artificial Intelligence*, J. Lang, Ed. Stockholm: International Joint Conferences on Artificial Intelligence, 2017.
- [63] P. Moulon, P. Monasse, and R. Marlet, “Computer Vision – ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part IV.” Springer, 2013, pp. 257–270. [Online]. Available: https://books.google.co.za/books?id=FiS5BQAAQBAJ&dq=mux+pairs&source=gbs_navlinks_s http://dx.doi.org/10.1007/978-3-642-37447-0_20
- [64] A. Nishimura, “Classification of Two-dimensional Manifolds Using Morse Theory,” pp. 1–48, 2010.
- [65] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, “Loops in Reeb graphs of 2-manifolds,” *Proceedings of the nineteenth annual symposium on Computational geometry*, pp. 344–350, 2003.
- [66] A. KUNDU, “Introduction to Reeb Graphs and Contour Trees Quotient topology and space :.” [Online]. Available: <http://web.cse.ohio-state.edu/wang.1016/courses/788/Lecs/lec15-abhisek.pdf>

- [67] S. Qin, X. Zhu, Y. Yang, and Y. Jiang, “Real-time hand gesture recognition from depth images using convex shape decomposition method,” *Journal of Signal Processing Systems*, vol. 74, no. 1, pp. 47–58, 2014.
- [68] H. Liu, W. Liu, and L. J. Latecki, “Convex shape decomposition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 97–104, 2010.
- [69] N. C. Subhas, “Introduction Linear Programming Integer Programming,” Tech. Rep. [Online]. Available: <https://www.isical.ac.in/~arijit/courses/autumn2016/ILP-Lecture-1.pdf>
- [70] J. K. Karlof, *Integer programming : theory and practice*. Taylor & Francis, 2006.
- [71] Y. Morenko, “Continuous and discrete optimization techniques for some problems in industrial engineering and materials design,” Ph.D. dissertation, 2015. [Online]. Available: <https://doi.org/10.17077/etd.q5rqf58m> <https://ir.uiowa.edu/etd/1991>
- [72] J. W. Chinneck, “Chapter 13: Binary and Mixed-Integer Programming,” *Practical Optimization: a Gentle Introduction*, pp. 1–11, 2004. [Online]. Available: <http://www.sce.carleton.ca/faculty/chinneck/po.html>
- [73] IBM Knowledge Center, “what does cplex do at each node.” [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/UsrMan/topics/preface/whatdoes.html
- [74] Lindo Systems Inc, “LINGO: The Modeling Language and Optimizer,” *Chicago, Illinois*, p. 869, 2018. [Online]. Available: <http://www.lindo.com> <https://www.lindo.com/index.php/products/lingo-and-optimization-modeling>
- [75] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Proceedings of the Alvey Vision Conference 1988*, pp. 23.1–23.6, 1988. [Online]. Available: <http://www.bmva.org/bmvc/1988/avc-88-023.html>
- [76] L. J. Latecki and R. Lakamper, “Shape Similarity Measure Based on Correspondence of Visual Parts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1–6, 2000.
- [77] L. Latecki and R. Lakämper, “Polygon evolution by vertex deletion,” *SCALE-SPACE '99 Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pp. 398–409, 2010.
- [78] M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf, “Computational Geometry,” *Springer*, pp. 191–218, 2008.
- [79] M. Kline, “Calculus: An Intuitive and Physical Approach,” in *Calculus: An Intuitive and Physical Approach*. Dover Publications, 1998, p. 960.

Appendix A

Article

Decomposition of complex two-dimensional shapes into simple convex shapes

ZANRICH VAN RENSBURG*

North-West University
zzanrich@gmail.com

November 20, 2018

Abstract

Decomposing a complex shape into visually significant parts comes naturally for humans, and turns out to be very useful in areas such as shape analysis, shape matching, recognition, topology extraction, collision detection and other geometric processing methods[1]. After analysis it was found that the Minimum Near-Convex Method(MNCD)[2] is one of the most promising algorithm currently available that shows room for improvement. The focus of this dissertation is to make an improvement on the time it takes to decompose a complex shape, while keeping the decomposition(number of parts) results the same. One improvement that was implemented was to neglect the Morse function, as this takes a long time to execute. Another improvement was to make use of Delaunay Triangulation(DT) instead of considering all of the vertices, as no overlapping will take place and the need for the non-overlapping matrix $\mathbf{B}$ is no longer necessary. Experimental results show that an average time reduction of 15%, but an increase in the number of parts. Thus there is an improvement made on the duration of the algorithm, but there is room to improve on the total amount of parts obtained after decomposition.

Keywords:

shape decomposition, complex shapes, simple shapes, convex shapes, Delaunay Triangulation(DT), Minimum Near-Convex Decomposition(MNCD), Discrete Contour Evolution(DCE), shape simplification, time complexity, optimization, parts

I. INTRODUCTION

Identification in image processing is a rather large field, and can in itself include other fields of study. The three basic parts of image processing and pattern recognition include pre-processing, feature extraction and classification[23]. In order to be able to classify objects, a classifier requires features that has been extracted in the feature extraction process. If a feature is chosen properly it will not take a classifier long to classify an object. Therefore for this report, feature extraction was rather looked at and how possible improvements can be made to this section.

Upon investigation it was found that shape decomposition is one of the better ways to approach object recognition. This is because of how humans identify complex shapes as humans tend to break down such shapes into simpler, or more recognisable ones[3, 4]. The human cognitive system is very complex and trying to portray this to a computational system is not an easy task[5]. Thus, shape decomposition is known for taking a long time on a computer, especially if the shape becomes more complex.

Thus, the problem at hand will be to identify or classify objects at a reasonable speed. The main idea here is to decompose a complex shape of an object into simpler more classifiable shapes in a short amount of time. This is done in order to make the classification process shorter, and thus also the whole recognition process. This then leads to an

investigative question, which after some literature has been review, will help us shape a concrete investigative question. In shape-related areas such as computer vision, graphics and scientific data visualization, shape decomposition has been considered as an important problem[5]. Different methods of shape decomposition can generally be categorized into two classes. The first class is motivated by psychological studies[6], while the second is by geometric constraints[7].

In Psychological studies, a complex shape is decomposed into natural parts[8, 9]. The definition of natural parts is dependent on human conceptuality and therefore has no confirmed explanation. There does, however, exist several fundamental rules of perception that has been developed from cognitive science principles. Of which the most well known is the *minima rule*, *short-cut rule* and *limbs and neck rules*.

The geometric studies aim to decompose shapes into geometrically related parts [10]. The most popular geometric device that is used is convexity due to the fact that most convex parts have decent geometrical and topological properties. It is also an important constraint, as convexity plays a role in the human perception[11].

In order to measure the execution of this category, two indexes are looked at namely, time complexity and the number of decomposed parts. Time complexity is defined as the computational complexity that is used to describe the amount of time it takes for an algorithm to execute[12]. In [13] they use a Hierarchical Decomposition and Axial Shape Description

*A thank you or further information

Table 1: Table showing a summary of all the decomposition methods.

Method:	PFSD(Wang et al)	MNCD(Ren et al. 2011)	APSD(Hu et al.)	MNCD(Ren et al. 2013)	GBD(Kim et al.)
Advantages	Deformation invariant Distortion invariant Partly occlusion invariant	Distortion invariant Deformation invariant Visual Naturalness minum #parts	Decomposes with holes Saves Material	Distortion invariant Deformation invariant Visual Naturalness minimum #parts	Scaling invariant Rotation invariant Noise invariant Deformation invariant Occlusion invariant Visual naturalness
Disadvantages	Unnecessary Parts	Slow	Mostly for 3D-printing Very slow Not optimal solution	Slow, but not as slow as MNCD-1	Slow due to Morphological operation
Average #parts	8.25	9.13	3.63	9.13	4.78
Perception Rules	Minima Salient	Minima Short-cut		Minima Short-cut	
Parameters	β - balancing weight T_2 - largest rel. measure	ψ - convexity threshold λ - visual naturalness	n - no. of sample points m - no. of sampled base directions k - no. of block generations y^* - paramidity threshold	ψ - convexity threshold λ - visual naturalness	t_L - threshold values t_G - threshold values
Solutions	Binary Programming Problem	Global Optimal Algorithm	Pyramidality	Binary Integer Linear Programming	Irrative methods using CMD and RDS
Constraints		non-overlapping convexity		non-overlapping convexity	split merging
Based	Perceptual	Geometric	Geometric	Geometric	Geometric

method and the time complexity that was found is $O(N \log N)$, where N is the number of boundary points. Furthermore Ren, Yaun and Liu proposed a Minimum Near-Convex Decomposition(MNCD) method in [2]. They found that the time complexity of their algorithm can be written as $O(n^2)$.

The second execution measure is the number of parts. Here Lien[14] proposed a method where a 2D-shape is decomposed into the minimal amount of strictly convex parts which produces a large amount of decomposed parts. In order to overcome this problem, Lien and Amato [15] proposed an approximate convex decomposition method. Despite the improvement, there still remains two unsolved problems, being redundant parts are produces and it is difficult to obtain visual naturalness[2]. To solve these problems, Ren et al. in their MNCD method[2] break down complex shapes into a minimal amount of "near-convex" components as mentioned earlier.

Although there has been a lot of work done on improving the visual naturalness of shape decomposition, there is still a gap in the amount of time it takes to do so. This might be problematic in areas where instant recognition is required. This then leads us to our investigative question which is which decomposition method would decompose a complex shape into the least amount of simple shapes in a shorter amount of time?

The layout of the rest of the paper is as follows: The background study will be introduced in Section 2, where related work will be discussed. In section 3 an overview of the problem will be looked at, followed by Section 4 where the implementation will be discussed. This is then followed by the experimental results and a discussion thereof in Section 5. Finally in Section 6 a conclusion will be drawn an future work discussed.

II. LITERATURE OVERVIEW

i. Background

Part-based representation can be defined as the representation of a shape or an object in a number of its decomposed 'natural' parts[16]. Objects can have many different features like colour, texture, shape etc. that can be used for recognition. Of all these the most distinguished feature that can be detected is shape[17]. Decomposing a shape can lead to a better analysis, as well as an improved understanding of a shape by simplifying it into simpler parts[2, 11, 18]. Shape decomposition is used widely in image processes that include shape recognition and recovery[8, 2], skeletonization[19, 20, 21] and path planning[2, 22]. Possible examples of shape decomposition are decompositions into convex, spiral -and monotone polygons[23].

Most of the known shape decomposition methods can be classified into two classes. The first class is motivated by psychological studies[6], while the second is by geometric constraints[7]. The first class is proposes to break down objects into natural parts[8, 9]. Natural parts being defined as dependent on the cognition system of humans and has no verifiable explanation. There does exist several fundamental perceptual rules that have been developed from cognitive science principles. Some of these rules include the *minima rule*, the *short-cut rule* and the *limbs-and-necks rule*. De Winter and Wagemans[3] did a large scale study on the decomposition of object outlines into parts and found that the minima rule has the greatest influence on segmentation of shapes, which is followed by the short-cut rule and the limbs and neck rule. The second class is driven by geometric descriptors and aims to decompose shapes into geometrically related parts [10]. The most popular geometric device that is used is convexity, as convexity plays a role in the human perception[11], as it has been found that humans by nature tend to decom-

Table 2: Table showing a summary of all the decomposition methods.

Method:	PSD(Ma et al.)	CSD(Liu et al.)	CMSR(Luo et al.)	ACD(Lien et al.)	AD(Lu et al.)
Advantages	High Visual Naturalness Decomposed with holes		Noise invariant Occlusion invariant Articulation invariant Rotation invariant	Decomposes with holes	Persistent convex parts Topological noise invariant Transformation invariant Deformation invariant Visual naturalness Fast decomposition
Disadvantages	Redundant parts Slow	Slow due to Morse functions Redundant parts	To many parameters	Slow due to iterative processes	No perception rules used
Average #parts	6.88	10.75	6.67	12.50	8.86
Perception Rules	Short-cut Minima Convexity	Short-cut	Minima Short-cut	Minima Short-cut	
Parameters	ϵ - concavity tolerance a - weight of cut income b - part-similarity	t - no. of Morse function ϵ - concavity tolerance	t_{DCE} - DCE stop parameter n_d - No. of directions t_{h2} - associated neighborhood	τ - allowable concavity	α - diameter of convolving circle
Solutions	Quadratically Constrained Quadratic Program	Integer Linear Programming	Greedy Algorithm	Greedy Algorithm	Constrained Optimization Strategy
Constraints		convexity			
Based	Perceptual	Perceptual and Geometric	Perceptual	Geometric	Geometric

pose shapes into visual parts that are convex. In order to measure the execution of this category, time complexity and the number of decomposed parts are used[]. This class can be summarized as methods that makes use of mathematical expressions and tend to take less time. This will also be an important point to consider when comparing different decomposition methods, as the aim is to improve on existing method's time complexity.

In a survey done in 2008 by [24] of shape feature extraction techniques, it is said that a descriptor attempts to quantify shapes in ways that agree to human intuition. It is important to note that shape descriptors are divided into two main categories, the first category is *contour based*, while the second category is *region based*. As their names suggest one is about the contour while the other is about the whole shape. Some of the contour descriptors include chain code representation, graph-based representation, bending energy, convexity, shape signature etc.[]. Some of the region descriptors include convex hull, medial axis, geometric moments, grid descriptors etc.[]

ii. Literature review

Time complexity can be defined as the time taken by an algorithm to run as a function of the size of the input[25]. In [19], Aurenhammer introduced a weighted skeleton and fixed-share decomposition concept(WSFD). For this algorithm, a runtime of $O(m \log n)$ is used, where n is the number of sides of a convex polygon and m is the number of points that forms part of the set S that is enclosed by P , and m -number of points can be selected such that $|region(e_i) \cap S| \geq m_i$, for $i = 1, \dots, n$. Wang et al. [26] proposed a shape decomposition strategy that is focused on the relationship between the part cuts and the segment points called the Perceptually Friendly Shape Decomposition(PFSD). For their algorithm, the time complexity is $O(N \log N)$, where N is the number of boundary points.

Table 3: Table showing the score of all the decomposition methods.

Methods	Score
MNCD(Ren et al)	0.13
GBD(Kim et al.)	0.78
PSD(Ma et al.)	0.88
PFSD(Wang et al)	1.25
CMSR(Luo et al.)	1.67
AD(Lu et al.)	2.86
ACD(Lien et al.)	6.88
APSD(Hu et al.)	8.63
CSD(Liu et al.)	10.38

In [13], Rom and Medioni uses a Hierarchical Decomposition and Axial Shape Description(HDASD) method to recognise parts. Here a complexity of $O(n^2)$ was found, where n is the number of circular arcs in the approximation of a shape contour.

Lu et al. proposed an α -decomposition(AD) of polygons[27] where the time complexity is $O(1)$, which means that the time it takes to run the algorithm is not dependant on the size of the input.

Ren, Yaun and Liu proposed a Minimum Near-Convex Decomposition(MNCD) method in [2]. Their time complexity can be written as $O(n^2)$. In [28], Keil and Snoeyink obtained in their work, the Convex Decomposition of Simple Shapes(CDSS), a time complexity of $O(n + r^2 \min\{r^2, n\})$ with n vertices and r notches.

Comparing the time complexities with each other it was found that the MNCD/HDASD methods are the most complex followed by SSDS, PFSD, CDSS, WSFD and lastly AD. **Number of parts** after decomposition is the next evaluation process that will be discussed. The advantages of Wang et al.(PFSD)[26] is that this method is robust to occlusion, deformation and distortion and they also made use of the minima rule to obtain parts close to human perception.

In Ren et al. (MNCD)[2] the advantages that this method holds is that it is unrefined to shape deformation and local distortions and it also makes use of the minima and short-cut perception rules to decompose the shape. They also make use of binary integer linear programming(BILP) to obtain the optimal number of cuts.

In Liu et al.[5] they propose convex shape decomposition(CSD). Their method is based on the Morse theory and multiple Morse functions. Furthermore, they use integer linear programming (ILP) to optimize the solution. Of the perception rules, they only make use of the short-cut rule. One disadvantage that this method holds is that it is also time consuming due to the Morse functions used.

In [15], Lien and Amato proposed an approximate convex decomposition method(ACD). Their algorithm makes use of iteratively removing the most significant non-convex feature thus taking longer to decompose. This then produces a hierarchical representation that provides a series of increasingly convex decompositions. Advantages that this method holds is that it can be used on shapes with and without holes, but also needs user input, that can produce strange results when chosen incorrectly.

A summary of different method that where considered is given in tables 1 and 2. In order to obtain the best possible results, all of the methods were given scores based on their advantages, disadvantages, perceptual rules used, number of parts, parameters, optimization technique and the decomposition based method. The results are shown in table 3.

The smallest scores are then the better methods to consider. It is also important to take note that although these scores are determined, the time complexity of these methods as well as the number of parts needs to be taken into consideration.

Looking at table 3, it was decided to use methods MNCD, ACD and CSD. This was decided because of all the methods considered, these methods made use of perceptual rules, they are geometrically based and they all use optimization techniques to obtain the optimal cuts.

Plotting these methods' time complexities against each other??, one can see that the CSD method has the higher time complexity at lower values, and becomes less complex than MNCD at larger values. These values are quite close to each other, and thus the decomposition results will also be looked at.

The decomposition results of the different methods can be seen in figure 1. It can be seen in figure 1, that the ACD method produces the most amount of redundant parts, followed by CSD lastly MNCD. The average number of parts of, rounding off, gives us: 13, 11 and 10 parts. Thus if a choice is to be made in terms of the least amount of parts, the method of choice will be MNCD, as it produce the least amount of parts as well as the most visually natural parts.

Looking at the decomposition results of the three methods, one can see that ACD that there exist a lot of unnecessary parts, for example when looking at the cow's decomposition, there exists cuts in the body that is totally unnecessary. When looking at CSD, the same picture produces better results, but

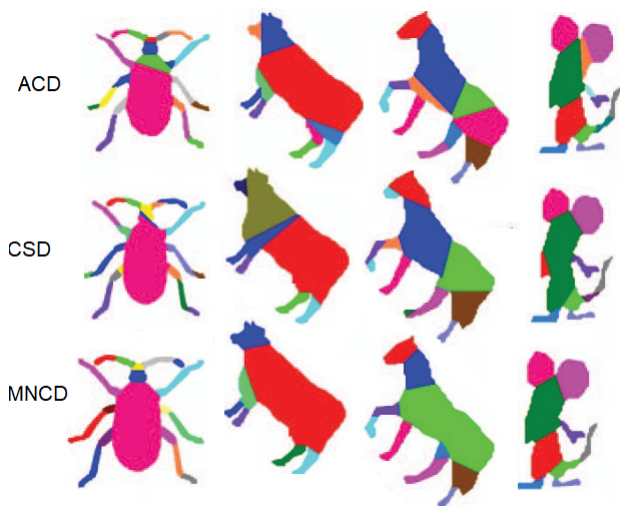


Figure 1: Decomposition comparison of methods ACD, CSD and MNCD in terms of their number of parts.

there is still some extra parts. The MNCD produces more visual natural parts, but also has an extra parts cut by the hind leg.

The MNCD will therefore be considered based on the visual naturalness of its decomposition. It is important to note that one major disadvantage of this method is that is very slow. The Minimum Near-Convex Decomposition(MNCD) method can break down a complex polygon into the least amount of simple shapes, and needs improvement on the amount of time it takes to do so.

III. MY APPROACH OVERVIEW

Algorithm overview

Algorithm 1 shows the overall procedure of the proposed method. Input to the algorithm is an image containing a complex shape. The first step is then to obtain the Discrete Contour Evolution (DCE). Here the shape is first simplified before further use. This is then used in conjunction with Harris corner detection to determine the most important points or corners of the object at hand. Once these points are obtained, the Delaunay triangulation(DT) is implemented, which will ensure convex cuts that are optimal. The mutex-pairs are computed next, followed by matrix $\mathbf{A}$ and then the cost-function $\mathbf{w}$. These matrices are then optimized using Binary Integer Linear Programming(BILP), which will return the final optimal cut vector. Lastly the simplest convex shapes are fit to the decomposed object in order to reveal simpler less complex shapes.

i. Discrete Contour Evolution(DCE)

Consider S to be a closed complex polygon (that is, a polygon that is concave and has more than three vertices). A vertex of S will be denoted as $v(S)$. A sequence of polygons, $S = (S^0, \dots, S^m)$ are produced until $|v(S^m)| \leq 3$, with the discrete

Algorithm 1 Algorithm for complex shape decomposition**Input:** A complex shape(S), and concave threshold(ψ)**Output:** Decomposed shape P_i into simpler convex shapes

```

1: Obtain DCE and optimal number of vertices  $J(S)$ 
2: Compute corners using Harris-Corner detection,  $J(S) \rightarrow H(S)$ 
3: Compute cut set using Delaunay triangulation,  $C(S)$ 
4: Compute  $\psi$ -mutex set of  $H(S) \rightarrow M^\psi(S)$ 
5: for each  $mp_i$  in  $M(S)$  do
6:   for each  $cut_j$  in  $C(S)$  do
7:     check whether  $cut_j$  separates  $mp_i \rightarrow a_{ij}$ 
8:   end for
9: end for
10: for each  $cut_j$  in  $C(S)$  do
11:   obtain distance and curvature to determine cost  $\rightarrow w_i$ 
12: end for
13: Solve the optimization problem in Eq. ??  $\rightarrow P_i$ 
14: for each  $P_i$  in  $P(S)$  do
15:   Fit simplest convex shape  $\rightarrow CP_i$ 
16: end for

```

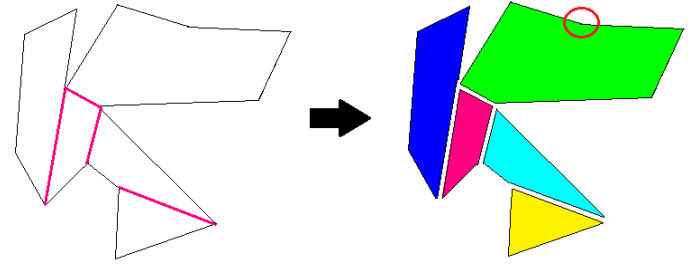


Figure 3: Picture demonstrating that near-convex decomposition does not decompose a complex shape into strict convex parts. As can be seen in the decomposed picture on the right, the red circle indicates a concave vertex, thus the green part is not convex, but near-convex.

ii. Finding corners

A classical approach to find corners in an image is to use the Harris feature detector. Harris looks at the average directional intensity change in a small window around a point of interest in order to define the detection of corners in an image. The average intensity change, when considering a displacement vector $(\Delta x, \Delta y)$, is given by[30]:

$$f(x, y) = \sum_{(x_k, y_k) \in W} (I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y))^2 \quad (2)$$

The Harris Corner Detection algorithm can be summed up in these five steps:

1. Convert image to a grayscale image
2. Calculate the spacial derivative
3. Constructing the variable matrix M
4. Calculate the Harris response
5. Suppress the non-maximum colour image to grayscale

These steps are implemented by using an library, as the Harris Corner Detection algorithm has been developed for optimal solution, and rewriting this algorithm will be a waste of time.

iii. Delaunay Triangulation(DT)

The aim of DT within this algorithm is to create a set of cuts that will not overlap each other, and to ensure that the cuts will be convex of nature[31]. Many algorithms currently exists that can compute the DT of complex shapes. Several algorithms have difficult internal details but are very effective. The following summary shows the steps of the less complex algorithms:

1. Create the outer triangle and begin at one of its intersection points - this ensures a explicit external beginning point.
2. Add an inner point; then remove those triangulations whose circum-circles contains that point.
3. Re-draw the triangles, ensuring to include the new point in the circum-circles of the just erased triangles.

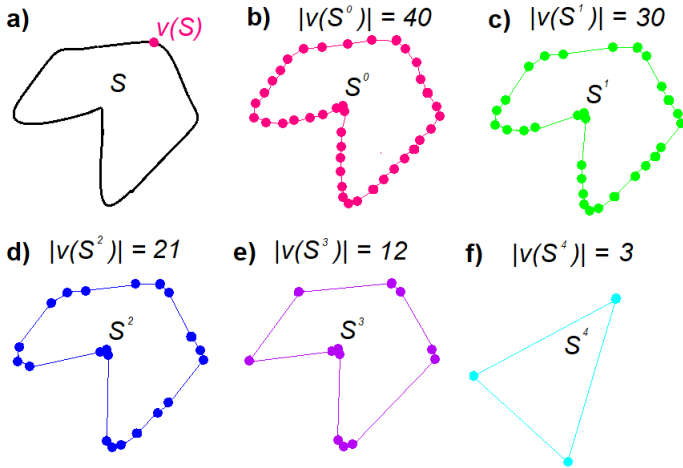


Figure 2: Process showing how the DCE process works.

contour evolution(DCE) technique[29]. Here $|\cdot|$ indicates the number of vertices. This is demonstrated in figure 2. In order to determine the DCE of a complex shape, a relevance measure K given by Eq.1:

$$K(\beta, l_1, l_2) = \frac{\beta l_1 l_2}{l_1 + l_2} \quad (1)$$

where β is the turn angle at the common vertex v in the shape S^i . Furthermore l_1 and l_2 are the lengths of $\overline{vu}$ and $\overline{vw}$. Both of these lengths are to be normalized to the total length of S . After every evolution step, the amount of intersection points decrement by at least one, which will cause termination of the process when the number of vertices ≤ 3 . A method of determining a stopping criteria will be discussed in section IV.

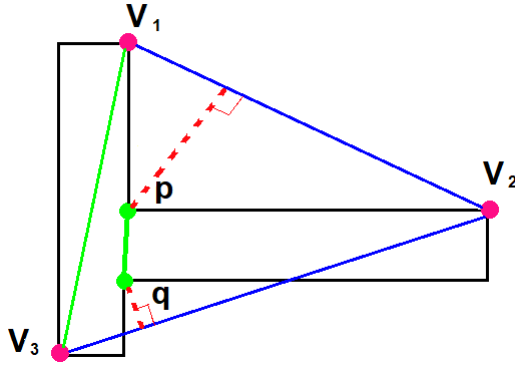


Figure 4: A figure demonstrating that any two vertices pq or v_1v_3 with a line connecting them that lies completely within the contour, is considered a cut (shown in green). The line v_1v_2 is found outside the concave part of the contour and line v_2, v_3 intersect with the contour, which forms mutex pairs (shown in blue).

4. Repeat steps two and three until there is no more points left to add.

Since this method has been used many times before, there exists some standard DT algorithms that can be used to determine the DT of a set of points, without having to do it from first principles.

iv. Decomposition of shape

Here the decomposition method used by the MNCD method will be discussed. Each decomposed part in a near-convex decomposition is not always found to be convex. This concept is illustrated in figure 3.

A formal definition to decompose a shape S into a ψ -near-convex decomposition can be given as [2]: for some decomposition, $D_\psi(S)$, containing only ψ -near-convex parts and contains no overlapping parts, can be written as:

$$D_\psi(S) = \{P_i \mid \cup_i P_i = S, \forall_{i \neq j} P_i \cap P_j = \emptyset, \text{concave}(P_i) \leq \psi\} \quad (3)$$

where P_i denotes the decomposed part. As can be seen in Eq. 3, there exists two constraints. The first is the *non-overlapping constraint* denoted by $\forall_{i \neq j} P_i \cap P_j = \emptyset$ and the second is the *convexity constraint* denoted by $\text{concave}(P_i) \leq \psi$. The concavity of P_i can be described by $\text{concave}(P_i)$.

A group of cuts form a partition, P_i . A cut can be defined as follows: If for any two points that lie on the border, say q and p , and the line joining these two points are located inside the border, then the line qp is called a cut [8]. This is demonstrated in figure 4. The complete candidate cut set in shape S is denoted by $C(S)$.

In order to measure $\text{concave}(P_i)$, the shape feature *mutex pairs* proposed in [8] will be used: For any two points, v_1 and v_2 , on the contour of a shape, if the line connecting points v_1 and v_2 locates outside the contour or it intersects with the contour, then (v_1, v_2) is known as a **mutex pair** (mutually exclusive pair). As can be seen in Fig. 4, (v_1, v_2) and (v_2, v_3) are two

mutex pairs. The maximal concavity of each mutex pair can be used to determine the concavity of each partition P_i :

$$\text{concave}(P_i) = \max_{(v_1, v_2) \in P_i} \{\text{concave}_m(v_1, v_2)\} \quad (4)$$

where (v_1, v_2) denotes the mutex pair in part P_i and $\text{concave}_m(v_1, v_2)$ is the concavity of the same mutex pair. Hence, we can obtain the concavity of a part, $\text{concave}(P_i)$, by measuring all of the mutex pairs' concavity, $\text{concave}_m(v_1, v_2)$, in P_i . Thus, $\text{concave}_m(v_1, v_2)$ is defined as: The maximum orthogonal length between a line (v_1v_2) and the closest related concave contour is defined as the concavity of that mutex pair.

As can be seen in figure 4, the red dotted line indicate how the concavity can be measured and shows the concavity measures of $\text{concave}_m(v_1, v_2)$ and $\text{concave}_m(v_2, v_3)$. It can be seen that $\text{concave}_m(v_1, v_2) > \text{concave}_m(v_2, v_3)$.

Minimum-Near Convex Decomposition (MNCD)

In a shape S , for a total of n potential cuts, that is $C(S) = \{cut_1, \dots, cut_n\}$, a subset that consist out these cuts, written as $C'(S) \subseteq C(S)$, is known as the final decomposition. For each cut_i in $C(S)$, a binary variable x_i is assigned where [8]:

$$x_i = \begin{cases} 0, & \text{if } cut_i \notin C'(S) \\ 1, & \text{if } cut_i \in C'(S) \end{cases} \quad (5)$$

Thus the cut selection or rejection from $C(S)$ is represented by a binary vector $x_{n \times 1} = (x_1, x_2, \dots, x_n)^T$. Thus the cut selection or rejection from $C(S)$ is represented by a binary vector $x_{n \times 1} = (x_1, x_2, \dots, x_n)^T$. By using perception rules to minimizing the total amount of cuts, and with the two limitations in equation 3, the ψ -MNCD can be expressed as a BILP problem [2]:

$$\begin{aligned} \min \quad & \|x\|_0 + \lambda w^T x \\ \text{s.t.} \quad & Ax \geq 1 \\ & Bx \leq 1 \\ & x \in \{0, 1\}^n \end{aligned} \quad (6)$$

here the amount of chosen cuts in $C'(S)$ is represented by the zero-norm $\|x\|_0$ of the vector x . Furthermore in order to control the visual naturalness, $\lambda \geq 0$, is introduced to $w^T x$, so that higher visual naturalness is favoured. The variable λ will be discussed in a later subsection [2].

Visual naturalness

The *short-cut rule* [11] and the *minima rule* [32] are both made use of to obtain visual naturalness. To calculate this visual naturalness, a cost to each cut is given, $cut_i \in C(S)$ [2]:

$$w_{v_1v_3} = \frac{\text{dist}[v_1v_3]}{1 + \beta |\min\{\text{cur}(v_1), 0\} + \min\{\text{cur}(v_3), 0\}|} \quad (7)$$

where v_1 and v_3 represents vertices and a possible cut represented by v_1v_3 forms part of the set $C(S)$. Furthermore the normalized distance between vertices v_1 and v_3 is $\text{dist}[v_1v_3]$ which represents the *short-cut rule*. The normalized curvature

of the vertices v_1 and v_3 , are shown as $cur(v_1)$ and $cur(v_3)$, and represents the *minima rule*. The cost of n candidate cuts is denoted by $\mathbf{w}_{n \times 1} = (w_1, w_2, \dots, w_n)^T$.

Convexity constraint

As mentioned earlier, to see to the convexity constraint $\forall(P_i), concave(P_i) \leq \psi$, the mutex pairs with concavities larger than ψ need to be divided into different parts. In order to achieve this the ψ -mutex set of S , $M^\psi(S)$ is first created. Let us assume that in the ψ -mutex set, $M^\psi(S) = \{mp_1, \dots, mp_m\}$, there are m mutex pairs. Then for each mutex pair in $M^\psi(S)$, mp_i , and for every cut that can separate it, the set $C'(S)$ must contain at least one cut. Thus, for every mp_i , this gives a restraint:

$$\sum_{j=1}^n a_{ij}x_j \geq 1, \text{ where } a_{ij} = \begin{cases} 1, & \text{if } mp_i \in M'_j \\ 0, & \text{if } mp_i \notin M'_j \end{cases} \quad (8)$$

Where $\mathbf{A}_{m \times n}$ denotes $(a_{ij} | i = 1, \dots, m; j = 1, \dots, n)$ and $\mathbf{1}_{m \times 1}$ denotes $(1, \dots, 1)^T$. Considering m , all of the mutex pairs in $M^\psi(S)$, one of the convexity constraints, $\mathbf{Ax} \geq \mathbf{1}$ in (6) is obtained.

Non overlapping constraint

In the function $C(S)$, for any of the possible cuts it might happen that two cuts can intersect. Suppose cut_i and cut_j meet at the k^{th} intersection, the in matrix $\mathbf{B}$ the k^{th} row can be defined as:

$$b_{kw} = \begin{cases} 1, & \text{if } w = i \text{ or } j \\ 0, & \text{if otherwise} \end{cases} \quad (9)$$

To guarantee that the non-overlapping constraint, $\forall_{i \neq j} P_i \cap P_j = \emptyset$, in (3) is met, intersection of the selected cuts in $C'(S)$ may not takes place. That is: $\forall_{g=1, \dots, t} \sum_{j=1}^n b_{gw}x_w \leq 1$. Thus the constraint of intersection can be written as: $\mathbf{Bx} \leq \mathbf{1}$, where $\mathbf{1}_{t \times 1} = (1, 1, \dots, 1)^T$.

Binary Integer Linear Programming(BILP)

An optimization of the expression created in (6) can efficiently be implemented using BILP. The binary nature of vector $\mathbf{x}$ can be used to rewrite (6) in a linear form: $\|\mathbf{x}\|_0 + \lambda \mathbf{w}^T \mathbf{x} = (\mathbf{1}^T + \lambda \mathbf{w}^T) \mathbf{x}$ where $\mathbf{1}_{n \times 1} = (1, \dots, 1)^T$.

A matrix can be defined as $\mathbf{D}_{(m+t) \times n}$ such that $\mathbf{D} = \begin{bmatrix} -\mathbf{A} \\ \mathbf{B} \end{bmatrix}$, and a vector $\mathbf{u}_{(m+t) \times 1}$ such that $\mathbf{u} = \begin{bmatrix} -\mathbf{1} \\ \mathbf{1} \end{bmatrix}$, where $-\mathbf{1}_{m \times 1} = (-1, -1, \dots, -1)^T$ and $\mathbf{1}_{t \times 1} = (1, 1, \dots, 1)^T$. Thus representing (6) as a BILP problem:

$$\begin{aligned} & \min(\mathbf{1}^T + \lambda \mathbf{w}^T) \mathbf{x} \\ & \text{s.t. : } \mathbf{Dx} \leq \mathbf{u} \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned} \quad (10)$$

By making use of discrete optimization techniques, for example integer relaxation, GLPK, CPLEX or Lingo, the above BILP problem can be solved. In their work[2] CPLEX was used.

IV.IMPROVED IMPLEMENTATION

i.Stopping criteria for DCE

In their latest work Ren et al.[2] decided to discard cuts where both ends are convex vertices as these cuts will not separate mutex pairs. They have also made use of Morse functions to solve the problem. In order to use less points and therefore decrease the time it takes to decompose a shape, DCE is used to obtain the vertices with the most contribution to the overall shape.

The problem with equation 1 is that no stopping parameter is specified, and so the evolution will continue until three vertices are left. In order to solve this problem, an algorithm was created to calculate the stopping criteria. In order to better explain the algorithm, figure 5 has been added and a explanation on how to determine the stopping criteria is to follow.

As can be seen in (a) the original shape is given as input. In (b) the contour has been obtained, and in (c) the original area $A_{original}$, shown in grey shade, and original length $L_{original}$, shown as the number of pink vertices. The next step is then to determine the vertices with the least amount of contribution (shown as cyan dots) (d). These vertices are then removed, and new lines are drawn to the remaining vertices (e). Also shown in (e), although very small, is the error $error(I)$ in area (shown in dark blue). Now the new area in (f) can be calculated as $newArea(I)$ and the new length is $newlen(I)$. Next, the area ratio $E(I) = \frac{eSum(I)}{Area(I)}$ and the length ratio $S(I) = \frac{newlen(I)}{len(I)}$ are calculated. Thus the first entry in $J(I) = E(I) + S(I)$ can be calculated. This process is then repeated until the length is equal to three. Repeating steps (d)-(f) we find in (g)-(i) the new area and length, and as can be seen in (h) the error is becoming larger with a decrease in number of vertices. After each process has been calculated, the J is then plotted against the number of points that has been removed from the original length. In order to determine the stopping criteria, the minimum point will be considered, shown in green circle in (j), as that is where the shape will still keep all of the information of the original image, with the least amount of vertices.

It is known that the DCE will delete one vertex at a time, which will result in a length of $n - 1$ for each time the loop is executed. Thus to define it mathematically:

$$newC_i = \begin{cases} contour_i, & \text{if } c_i \neq \min\{C(S)\} \\ \emptyset, & \text{if } c_i = \min\{C(S)\} \end{cases} \quad (11)$$

Where $new\mathbf{C}_{(n-1) \times 1} = (newc_0, newc_1, \dots, newc_{n-1})^T$ is a vector. Once $new\mathbf{C}$ is determined, the new shape is drawn on the clear image, $N(I)$, which will exclude the deleted vertex. To determine A_{diff} the first step would be to determine the error between the original image, $I(S)$, and the new image, $N(S)$. This is achieved by an exclusive disjunction (XOR) of the images with each other. Thus the error, $E(S)$ can be

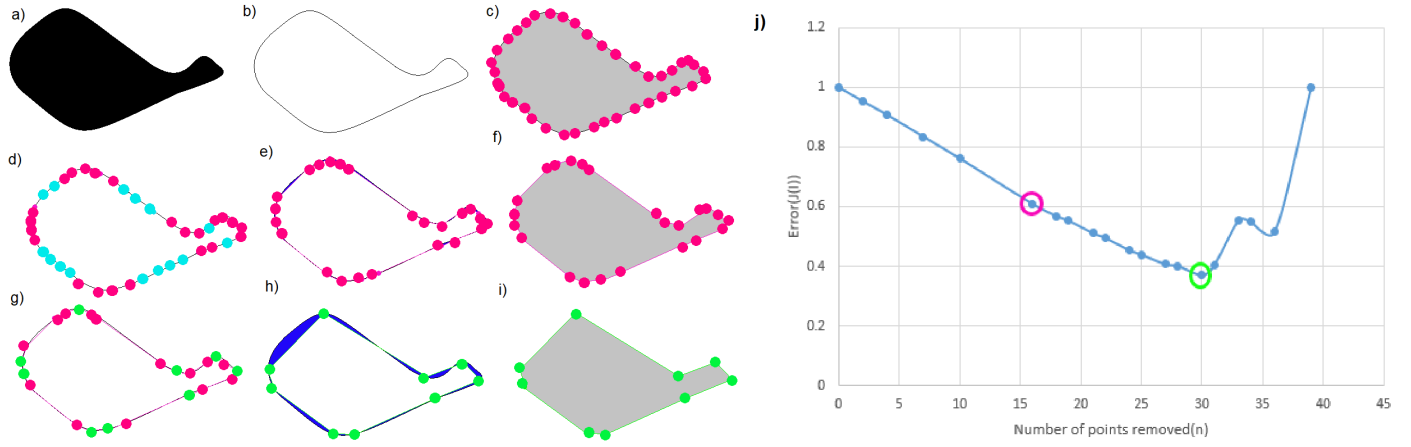


Figure 5: Algorithm ?? explained step-by-step. Plot of $J(I)$ vs. Number of parts removed shown in (j). The pink circle and green circle indicates the DCE process between (d-e) and (g-i) respectively. The green circle also indicates the stopping criteria, which is 9 vertices. plot

determined by:

$$E(S) = I(S) \oplus N(S) \quad (12)$$

In order to determine the difference in area A_{diff} , all of the values in $E(S)$ must be summed:

$$A_{diff} = \sum_{i=0}^n \sum_{j=0}^m E_{ij} \quad (13)$$

Where m and n shows dimensions of the original image. Next the ratios of the areas and the lengths are determined:

$$A_{ratio} = \frac{A_{diff}}{A_{original}} \quad (14)$$

and

$$L_{ratio} = \frac{L_{new}}{L_{original}} \quad (15)$$

Next the J vector is determined:

$$J(S) = A_{ratio} + L_{ratio} \quad (16)$$

Lastly, the minimum value of J is determined and the index of this value subtracted from the original length, $L_{original}$, to determine the stopping criteria, sc , for the evolution: Let i be such that $J_i = \min\{J\}$

$$sc = L_{original} - i \quad (17)$$

ii. Corners

It was decided in order to save computation time instead of using all of the vertices, that only the most important vertices will be used.

Concavity

After a point has been identified, two additional points are drawn on the contour an equal distance away from the point. It is then needed to determine the index of the point in

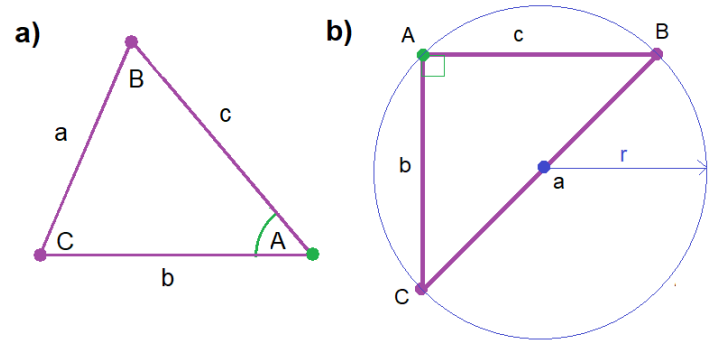


Figure 6: (a) shows the corner that has been detected (green) shows that two points, B and C, approximately ten pixels away are also used to determine the angle of the corner (A) by using the cosine rule. (b) shows how two points, B and C, are used to determine the curvature of the corner (A)

the contour vector, in order to determine the index that is ten intervals away. Thus, following the description of figure 6(a), mathematically: Let i be such that $contour_i = point(A)$ then:

$$point(B) = contour_{(i+10)} \text{ and } point(C) = contour_{(i-10)} \quad (18)$$

Now that the points B and C is determined, the cosine rule can be used to determine the angle of A. That is:

$$\cos(A) = \frac{b^2 + c^2 - a^2}{2bc} \quad (19)$$

but value of a is unknown. Thus using the distance formula:

$$a = \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2} \quad (20)$$

the angle A can be determined. To determine if the angle is convex or concave, we look at the side, a , opposite of the angle A . If this line lies outside the contour, the corner is concave. To determine if a is outside the contour an OpenCV

function called the **pointPolygonTest**. This function takes as input a coordinate pair to be tested. Thus a coordinate pair has to be given and the best one will be the point between B and C . Thus using the midpoint formula:

$$M = \left(\frac{x_b + x_c}{2}, \frac{y_b + y_c}{2} \right) \quad (21)$$

Thus using the OpenCV function:

$$dist = \text{pointPolygonTest}(M, \text{Contour}, \text{false}) \quad (22)$$

it will return a +1 when M is inside or a -1 when M is outside the contour. Referring to figure 6 we find that:

$$\angle A = \begin{cases} \angle A, & \text{if } dist = 1 \\ 2\pi - \angle A, & \text{if } dist = -1 \end{cases} \quad (23)$$

Curvature

Recall from equation 7 that the curvature of a point is to be used as part of the minima rule[8]. In order to determine the curvature of a line, the reciprocal of the radius can be used[33]. Thus if we have three points, $A(a_1, a_2)$, $B(b_1, b_2)$ and $C(c_1, c_2)$, the equation of the circle can be determined by solving the following expression:

$$\begin{vmatrix} (x^2 + y^2) & x & y & 1 \\ (a_1^2 + a_2^2) & a_1 & a_2 & 1 \\ (b_1^2 + b_2^2) & b_1 & b_2 & 1 \\ (c_1^2 + c_2^2) & c_1 & c_2 & 1 \end{vmatrix} = 0 \quad (24)$$

Using Laplace expansion and c_i to represent the coefficients of each of the first row terms above we find:

$$c_1 = a_1 \begin{vmatrix} b_2 & 1 \\ c_2 & 1 \end{vmatrix} - a_2 \begin{vmatrix} b_1 & 1 \\ c_1 & 1 \end{vmatrix} + 1 \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix} \quad (25)$$

similarly:

$$c_2 = (a_1^2 + a_2^2) \begin{vmatrix} b_2 & 1 \\ c_2 & 1 \end{vmatrix} - a_2 \begin{vmatrix} (b_1^2 + b_2^2) & 1 \\ (c_1^2 + c_2^2) & 1 \end{vmatrix} + 1 \begin{vmatrix} (b_1^2 + b_2^2) & b_2 \\ (c_1^2 + c_2^2) & c_2 \end{vmatrix} \quad (26)$$

$$c_3 = (a_1^2 + a_2^2) \begin{vmatrix} b_1 & 1 \\ c_1 & 1 \end{vmatrix} - a_1 \begin{vmatrix} (b_1^2 + b_2^2) & 1 \\ (c_1^2 + c_2^2) & 1 \end{vmatrix} + 1 \begin{vmatrix} (b_1^2 + b_2^2) & b_1 \\ (c_1^2 + c_2^2) & c_1 \end{vmatrix} \quad (27)$$

and

$$c_4 = (a_1^2 + a_2^2) \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix} - a_1 \begin{vmatrix} (b_1^2 + b_2^2) & b_2 \\ (c_1^2 + c_2^2) & c_2 \end{vmatrix} + a_2 \begin{vmatrix} (b_1^2 + b_2^2) & b_1 \\ (c_1^2 + c_2^2) & c_1 \end{vmatrix} \quad (28)$$

Thus simplifying the Laplace expansion of equation 24 to:

$$c_1 (x^2 + y^2) - c_2 x + c_3 y - c_4 = 0 \quad (29)$$

Now by completing the squares, equation 29 can be re-written in the standard circle equation form:

$$\left(x - \left(\frac{c_2}{2} \right) \right)^2 + \left(y - \left(\frac{c_3}{2} \right) \right)^2 = \frac{2c_4 + c_2^2 + c_3^2}{2c_1} \quad (30)$$

From which the radius r can be determined as follows:

$$r = \sqrt{\frac{2c_4 + c_2^2 + c_3^2}{2c_1}} \quad (31)$$

Finally, the curvature can be determined by using:

$$\kappa = \frac{1}{r} \quad (32)$$

As with the convexity, the curvature is given a negative sign if the midpoint, M lies inside the contour, and a positive sign if it lies outside the contour. That is:

$$\kappa = \begin{cases} -\frac{1}{r}, & \text{if } dist = 1 \\ \frac{1}{r}, & \text{if } dist = -1 \end{cases} \quad (33)$$

iii. Discrete lines

In order to determine each point of a line, a line iterator was created using Bresenham's line algorithm. Taking $A = (a_x, b_y)$ and $B = (b_x, b_y)$ as the two input points, then the first step will be to obtain the absolute value of the difference between these two points:

$$\Delta X_{abs} = |\Delta X| \text{ where } \Delta X = b_x - a_x \quad (34)$$

and

$$\Delta Y_{abs} = |\Delta Y| \text{ where } \Delta Y = b_y - a_y \quad (35)$$

The next step would be to determine if the line is vertical $b_x = a_x$, horizontal $b_y = a_y$, or diagonal $b_x \neq a_x$ and/or $b_y \neq a_y$. This is followed by a test to see which x coordinate and which y -coordinate is larger.

The Bresenham's algorithm then follows a simple set of rules to determine the next value of the vector. Therefore, the matrix can be mathematically described when considering the following conditions as:

If $a_x = b_x$ and $a_y > b_y$ then:

$$\sum_{i=1}^3 \sum_{j=0}^n = \text{line}_{ij} \text{ where } \begin{cases} \text{line}_{1j} = a_x \\ \text{line}_{2j} = j \\ \text{line}_{3j} = I(a_x, j) \end{cases} \text{ if } \forall j \in \text{Image} \quad (36)$$

If $a_y = b_y$ and $a_x > b_x$ then 36 becomes:

$$\sum_{i=1}^3 \sum_{j=0}^n = \text{line}_{ij} \text{ where } \begin{cases} \text{line}_{1j} = j \\ \text{line}_{2j} = a_y \\ \text{line}_{3j} = I(j, a_y) \end{cases} \text{ if } \forall j \in \text{Image} \quad (37)$$

where the intensity thereof is shown as $I(x, y)$. Lastly, if $a_x \neq b_x$ and $a_y \neq b_y$, the first test would be to check the slope. If the slope is steep, that is: $\Delta Y_{abs} > \Delta X_{abs}$ we can calculate the steep slope by:

$$m_{ss} = \frac{\Delta X}{\Delta Y} \quad (38)$$

else the slope is calculated as normal by:

$$m_s = \frac{\Delta Y}{\Delta X} \quad (39)$$

Knowing this, equation 36 becomes:

$$\sum_{i=1}^3 \sum_{j=0}^n = \text{line}_{ij} \text{ where } \begin{cases} \text{line}_{1j} = m_{ss}(j - a_y) + a_x \\ \text{line}_{2j} = j \\ \text{line}_{3j} = I(a_x, j) \end{cases} \text{ if } \forall j \in \text{Image} \quad (40)$$

and equation 37 becomes:

$$\sum_{i=1}^3 \sum_{j=0}^n = \text{line}_{ij} \text{ where } \begin{cases} \text{line}_{1j} = j \\ \text{line}_{2j} = m_s(j - a_x) + a_y \\ \text{line}_{3j} = I(j, a_y) \end{cases} \text{ if } \forall j \in \text{Image} \quad (41)$$

Now that the **line** is determined, we can use this information in conjunction with the function **pointPolygonTest** mentioned earlier to determine if a line lies inside, outside or on the contour.

iv.Cut set(C(S))

Non-overlapping constraint

In order to avoid the creation of the **B** matrix(see equation ??) completely, Delaunay Triangulation(DT) was brought into consideration. Since one of the constraints to the equation 6 require that the cuts are not to overlap, the DT seemed to be a fast and effective alternative. In order to apply DT in the project, OpenCV was used. This function requires one to give a set of points that the DT can be applied to. These points are determined in the previous step by using DCE and corner detection algorithms. The only down side to using DT is that some possible cut lines lie on the contour, and for all the possible triangles, there might be multiple lines on top of each other.

Removing lines

In order to remove lines that lie outside the shape, on on the contour, the same principle as discussed earlier in this section will be used. Each point on the line will be sent through the **pointPolygonTest** function and a negative, 0 or positive distance will be returned if the point is outside, on top of or inside of the contour. Thus to determine where the line lies, we calculate the average value of all of the points on the line:

$$d_{average} = \frac{\sum_{i=0}^m \text{dist}_i}{m} \quad (42)$$

If the value of $d_{average} < -1$ then the line lies outside, if $d_{average} > 1$ it lies inside and if $-1 \leq d_{average} \leq 1$ the line lies on the contour.

v.Mutex pairs

After the mutex pairs have been identified, the concavity of each mutex pair must be determined. As described in section ?? the concavity is determined by taking the maximum perpendicular length of each line to the contour. In order to determine which point is perpendicular, we look at the line equation:

$$y = mx + c \quad (43)$$

assume point $A(a_1, a_2)$ and point $B(b_1, b_2)$ are the two corner points of the mutex pair. Then the slope of the line connecting the two mutex points can be written as:

$$m_{AB} = \frac{b_2 - a_2}{b_1 - a_1} \quad (44)$$

we can then determine the equation for the line AB by using the "point-slope" equation and replacing the values and by choosing point $A(a_1, a_2)$ we find:

$$y = \frac{b_2 - a_2}{b_1 - a_1}(x - a_1) + a_2 \quad (45)$$

we know that for two lines to be perpendicular, $m_{AB} \times m_{CD} = -1$ must be true. Thus knowing that the furthest point of a mutex pair will always be a concave point, we know the coordinates of $C(c_1, c_2)$, and can thus determine the coordinates of $D(d_1, d_2)$, by obtaining the equation of line BD and equating that equation to equation 45:

$$\frac{b_2 - a_2}{b_1 - a_1}(x - a_1) + a_2 = -\frac{b_1 - a_1}{b_2 - a_2}(x - c_1) + c_2 \quad (46)$$

to obtain d_1 , the equation can be rewritten as:

$$d_1 = -\frac{\frac{a_1 b_1 - b_1 c_1}{a_2 - c_2} + \frac{a_1 a_2 - a_1 c_2}{a_1 - c_1} + b_2 - a_2}{\frac{a_2 - c_2}{a_1 - c_1} - \frac{c_1 - a_1}{a_2 - c_2}} \quad (47)$$

and thus d_2 can be found using equation ??:

$$d_2 = -\frac{b_1 - a_1}{b_2 - a_2}(d_1 - c_1) + c_2 \quad (48)$$

once these points are found, the distance formula can be used to determine the concavity of each mutex pair by:

$$\text{concavity}_i = \sqrt{(c_{1i} - d_{1i})^2 + (c_{2i} - d_{2i})^2} \text{ for } \{i | i = 0, 1, \dots, m\} \quad (49)$$

where m is the number of mutex pairs.

vi. Ψ -concavity

The ψ parameter is used to determine the degree of concavity that the user wants, and which to ignore. In essence the greater this parameter, the more decomposed parts will be delivered as all of the remaining parts will be strictly convex, and in our case we only want near-convex parts.

In order for the parts to be ψ -concave, the concavity as determined in equation 49 will be used as a measure. Thus the final mutex pair set, $\mathbf{M}_{final}(S)$, will be determined as follows:

$$m_{final_j} = \begin{cases} m_i, & \text{if } \text{concavity}_i \geq \psi \\ \emptyset, & \text{if otherwise} \end{cases} \quad (50)$$

for all mutex pairs in $\mathbf{M}(S)$, where $i = \{0, 1, 2, \dots, m\}$ and $j = \{0, 1, 2, \dots, m_f\}$ where m_f is the amount of mutex pairs that satisfies the condition $\text{concavity}_i \geq \psi$.

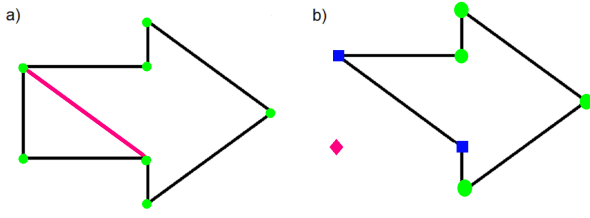


Figure 7: a) Part shown with one possible cut, in magenta. b) shows the one part that is cut and the green circle corner points that lie on the cut part, the blue square corner points that form part of the cut coordinates and the magenta diamond point that is outside the cut part.

vii. **A**-matrix

As it was explained earlier in section III, the **A** matrix is essentially a matrix used to describe whether a cut in $\mathbf{C}(S)$ can separate a mutex pair in $\mathbf{M}_{final}(S)$. In order to determine if a cut separates a mutex pair the first step is to draw each cut separately, and split the original part into two parts. The next step then is to only draw one of the two parts. This is then used with the corner points to determine which points are on the shape and which are not. These two steps are depicted in figure 7.

In a) it can be seen that the first cut in the cut-set $\mathbf{C}(S)$ is considered, shown in the magenta line. In b) the part is cut into two parts and only one part is shown. The part corner points on the part is then indicated as green circles, while the corners on the cut is blue squares and the corner out of the part is shown in magenta diamond.

For each corner point that lies on the contour of the cut part, a one is assigned, while the rest of the corner points, including the points that form part of the cut, $S = (s_1; s_2)$ and $E = (e_1; e_2)$ are assigned zeros. These values are then stored in a "look up" matrix, $L(S)$. Therefore the look-up matrix can be created by:

$$l_i = \begin{cases} 1, & \text{if } dist = 0 \text{ and endpoints} \neq S \text{ or } E \\ 0, & \text{otherwise} \end{cases} \quad (51)$$

thus yielding a vector $L(S)$, with size n where n is the number of corner points. After this vector has been created, the **A** matrix can be determined column by column.

This is done by checking the coordinate pairs of all of the mutex pairs. For each mutex pair, mp_i , there will be two sets of coordinates to indicate the mutex pair. Thus, for each mutex pair, the first coordinate pair, $A(x_a, y_a)$ is looked up in the look-up vector, giving us the first half, denoted as $A_{i,1a}$. If the value is one in the look-up vector, it will also be one in $A_{i,1a}$. The second coordinate pair, $B(x_b, y_b)$ is then looked up, and we obtain the second half of the column, denoted as $A_{i,1b}$. These two vectors are then exclusive disjunctive or better known as 'XOR'-ed with each other to reveal the first column of **A**, denoted by $\mathbf{A}_{i,1}$. Thus, mathematically it can be written

as:

$$a_{i1a} = \begin{cases} 1, & \text{if } A(x_a, y_a) \in \mathbf{L}(S) \\ 0, & \text{otherwise} \end{cases} \quad (52)$$

and

$$a_{i1b} = \begin{cases} 1, & \text{if } A(x_b, y_b) \in \mathbf{L}(S) \\ 0, & \text{otherwise} \end{cases} \quad (53)$$

Now

$$a_{i1} = a_{i1a} \oplus a_{i1b} \quad (54)$$

These equations are specifically used to determine the first column of the **A** matrix. Thus if $i = 0, 1, 2, \dots, m$ where m is the amount of mutex pairs, and if $j = 0, 1, 2, \dots, n$ where n is the number of cuts, then equation 54 becomes :

$$a_{ij} = a_{ija} \oplus a_{ijb} \quad (55)$$

Which will give an **A** matrix of size $m \times n$.

viii. Cost function($W(S)$)

The next part to determine is the cost function. As mentioned in section III, the cost function is determined using the equation 7. Thus to determine the numerator the distance formula is used as mentioned in section III:

$$dist(pq) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2} \quad (56)$$

Next, the curvature is determined using equation 33. It is important to note that in equation 7 in the minimum curvature is to be determined, $\min\{cur(p), 0\}$. Mathematically, this can be described as:

$$cur_{min} = \begin{cases} cur(p), & \text{if } \kappa > 0 \\ 0, & \text{otherwise} \end{cases} \quad (57)$$

In this case, the curvature, $cur(x)$, is either 0 when the corner is concave, that is $\geq 180^\circ$, or it is actual curvature value, κ when it is convex. The parameter β is set to one as the relation between the numerator and the denominator want to be kept equal.

ix. Binary Integer Linear Programming (BILP)

Recall that the binary integer linear programming (BILP) is used to obtain the optimal cuts by using the **A** matrix. The optimization problem can be seen in section III. In our case, there will be no **B** matrix created, and thus $\mathbf{D} = -\mathbf{A}$. The same goes for **u**, becoming $\mathbf{u} = -\mathbf{1}$. Thus, equation (10) can be rewritten as:

$$\begin{aligned} \min(\mathbf{1}^T + \lambda \mathbf{w}^T) \mathbf{x} \\ \text{s.t. : } \mathbf{A} \mathbf{x} \geq \mathbf{1} \\ \mathbf{x} \in \{0, 1\}^n \end{aligned} \quad (58)$$

where $\mathbf{A}_{m \times n}$ and vector $\mathbf{1}_{m \times 1}$, where $\mathbf{1}_{m \times 1} = (1, 1, \dots, 1)^T$ and the number of mutex pairs is shown by m and the number of cuts by n . The next variable to determine is the parameter λ . This parameter is defined as $0 \leq \lambda \leq \frac{1}{\sum_{i=1}^n w_i}$. This parameter

will also be changed and results compared with each other in section V. Now that all of the values are known, the above BILP problem can be solved. In order to do this, a python library called CVXOPT was used.

x. Identifying simpler shapes

Here *OpenCV* contains another function that can be used called *approxPolyDP(contour, epsilon, True)*. As input this function requires a contour, obtained from the previously mentioned function, and an epsilon value, as well as a true or false, which indicates if the approximation polygon is a closed shape or not. The epsilon value is the maximum distance from the contour to the approximated contour. This value, if chosen correctly, can produce the desired outcome.

In order to determine the epsilon value, another *OpenCV* function called *arcLength(contour, True)* is used. As input the contour as mentioned previously is given, as well as a true or false input indicating whether a contour is closed (a polygon) or open (a curve). This function returns the perimeter of the contour, and if a parameter σ is chosen in such a way that $\epsilon = \sigma \times \text{perimeter}$, then that epsilon will produce the wanted outcome. This parameter, σ will be evaluated in chapter ??.

It is important to note that this method will only be used to identify the number of sides the shape has, and won't be used to identify specific shapes, for example, a right-angle triangle and an acute triangle will both be classified as a triangle. The same goes for a square, rectangle and parallelogram, as they will all be classified as quadrilaterals.

Each shape is then classified according to the amount of points the approximate contour vector has. Mathematically, it can be written as:

$$\text{shape}_i = \begin{cases} \text{triangle}, & \text{if } \text{len}(\text{contour}_i) = 3 \\ \text{quadrilateral}, & \text{if } \text{len}(\text{contour}_i) = 4 \\ \text{pentagon}, & \text{if } \text{len}(\text{contour}_i) = 5 \\ \text{hexagon}, & \text{if } \text{len}(\text{contour}_i) = 6 \\ \text{check}, & \text{otherwise} \end{cases} \quad (59)$$

V. EXPERIMENTS

For the experiments, the MPEG-7 shape dataset will be used. The experimental set-up consists of 19 pictures each tested according to speed (time in seconds) and number of parts. This will then be compared to the MNCD method which this paper is trying to improve on. Each parameter will be changed while the others are kept constant in order to determine the effects of the different parameters on the accuracy and the speed. Recall from section ?? that there are three main parameters that might have different outcomes on the results. The first, and most important being Ψ . This is the concavity measure and is used to determine how convex a part is to be after decomposition. Next is the β parameter, that is used in the cost-function $\mathbf{w}$ to determine the relationship between the *short-cut rule* and the *minima rule*. Lastly we

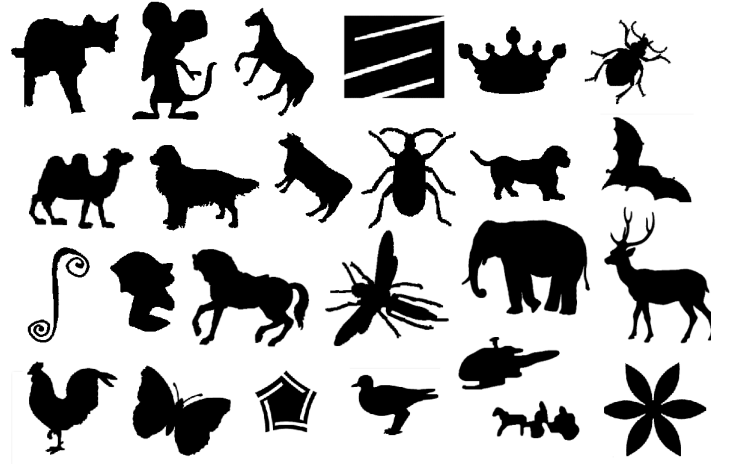


Figure 8: Example figures that forms part of the MPEG-7 shape dataset.

have the parameter λ which is used to determine the visual naturalness of the results. After the experiments were done, it was found that the only parameter that has an influence on the time as well as the number of parts is Ψ .

i. Time evaluation

For each parameter Ψ , β and λ the average time of each decomposition was taken. That is, for each picture, the decomposition was run three times and then the average time calculated. In order to calculate the average reduction in time, the following formula was used:

$$\text{time} \downarrow = \frac{\text{time}_{\text{MNCD}} - \text{time}_{\text{NEW}}}{\text{time}_{\text{MNCD}}} \times 100 \quad (60)$$

It is important to note that this time is in percentage(%). On average, it can be seen that the time was reduced by a total average of 15.33%, 14.92% and 15% for Ψ , λ and β . After changing these parameters, it was found that the relationship between the Ψ and the amount of time reduction is directly proportional to each other. Mathematically it can be written as:

$$\Psi \propto \text{time} \downarrow \quad (61)$$

It can then also be rewritten as:

$$\Psi \propto \frac{1}{\text{time}_{\text{new}}} \quad (62)$$

since:

$$\text{time}_{\text{new}} \propto \frac{1}{\text{time} \downarrow} \quad (63)$$

The results of the time vs. Ψ parameter is shown in table 4.

ii. Number of Parts(NOP) evaluation

In this section, the parameters Ψ , λ and β will be changed for each picture, and the number of parts produced after decomposition will be recorded in order to determine the relationship between the number of parts and the different

Table 4: Table showing the average reduction(%) in time for the pictures of the MPEG-7 dataset with $\lambda = 1$ and $\beta = 1$.

	$\psi = 0,005$ R	$\psi = 0,05$ R	$\psi = 0,1$ R	$\psi = 0,2$ R	$\psi = 0,5$ R
MPEG-7 dataset	New↓(%)	New↓(%)	New↓(%)	New↓(%)	New↓(%)
bat	72.76	71.72	72.84	72.77	72.98
bird	39.17	36.61	38.96	39.26	38.87
butterfly	28.72	27.34	28.35	28.42	28.30
camel	-30.83	-31.04	-30.43	-30.52	-30.02
carraige	16.73	15.68	17.81	17.41	18.64
cattle	49.42	49.24	49.36	49.65	49.96
chicken	46.46	45.99	46.05	46.18	46.24
chopper	-8.52	-10.05	-5.88	-7.30	-5.45
crown	38.19	38.88	43.20	39.33	38.62
deer	-42.06	-42.18	-42.92	-41.98	-41.79
dog	35.83	34.15	34.92	36.16	36.36
elephant	70.18	69.99	64.83	70.09	70.21
fly	7.90	7.80	12.39	8.04	8.41
horse	-43.65	-43.38	-43.86	-43.02	-42.17
horseshoe	33.11	32.00	30.34	32.90	33.05
lizzard	-9.61	-10.02	-10.58	-9.81	-9.42
misk	31.35	31.55	32.86	32.06	33.23
mickey	21.85	20.36	21.86	21.15	21.51
spring	-65.88	-65.83	-70.45	-66.07	-65.52
Average↓(%)	15.32	14.67	15.24	15.51	15.89

parameters. Here it was also found that the only parameter that really has an influence on the number of parts is the Ψ parameter. The results of the number of parts vs the Ψ are shown in table 5. Looking at the results, we can derive a general relation for NOP and the parameters:

$$NOP \propto \frac{1}{\Psi} \quad (64)$$

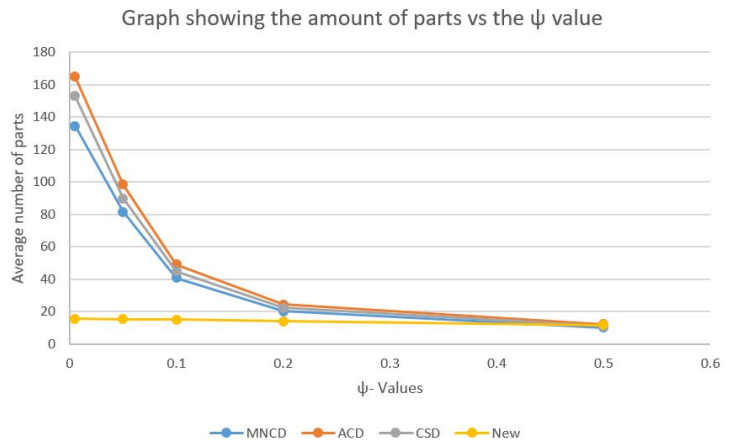
iii.Number of parts comparison to other methods

In this section the number of parts obtained by the new method will be compared and discussed to the number of parts delivered by the other methods. The other methods that the comparisons are made to is the Approximate Contour Decomposition(ACD)[15], the Contour Shape Decomposition(CSD)[5] and the Minimum Near-Convex Decomposition(MNCD)[2]. For this experiment, mainly the Ψ -parameter value will be changed and the values compared. The results of the NOP of the different methods are shown in figure 9. As one can see in the figure, all of the methods, as the Ψ -parameter increases, the number of parts decreases. The relationship that the number of parts decrease is different for the ACD, CSD and MNCD methods. For these three methods, the relation is of a logarithmic nature, while for our method, the relation is more linear of nature.

This phenomenon can then be used to explain the large percentage improvement. The large percentage of improvement is due to the fact that the relation between the Ψ -parameter and the number of parts are generally logarithmic of nature. The main reason why our method is not logarithmic but linear instead, is because of the number of vertices used to decompose the pictures. Where all of the other methods consider all of the vertices when deciding to decompose

Table 5: Table showing the total number of parts(NOP) for the pictures of the MPEG-7 dataset with $\lambda = 1$ and $\beta = 1$.

ψ	0.005	0.05	0.1	0.2	0.5
	New NOP	New NOP	New NOP	New NOP	New NOP
bat	12	12	11	11	5
bird	13	12	12	9	8
butterfly	11	10	9	9	9
camel	20	20	19	19	16
carraige	10	10	10	9	7
cattle	23	23	22	20	17
chicken	6	6	6	6	6
chopper	5	5	5	5	4
crown	7	7	7	7	7
deer	34	34	34	30	22
dog	9	9	9	8	7
elephant	21	21	21	21	17
fly	18	18	18	18	13
horse	24	23	23	21	17
horseshoe	7	7	7	7	7
lizzard	23	23	22	19	15
mickey	10	10	10	10	8
misk	10	10	10	9	8
spring	33	33	32	32	30
Average	15.58	15.42	15.11	14.21	11.74

**Figure 9:** Graph showing the graph of the different methods and Number of parts when Ψ -parameter has been changed, with $\beta = 1$, and $\lambda = 1$.

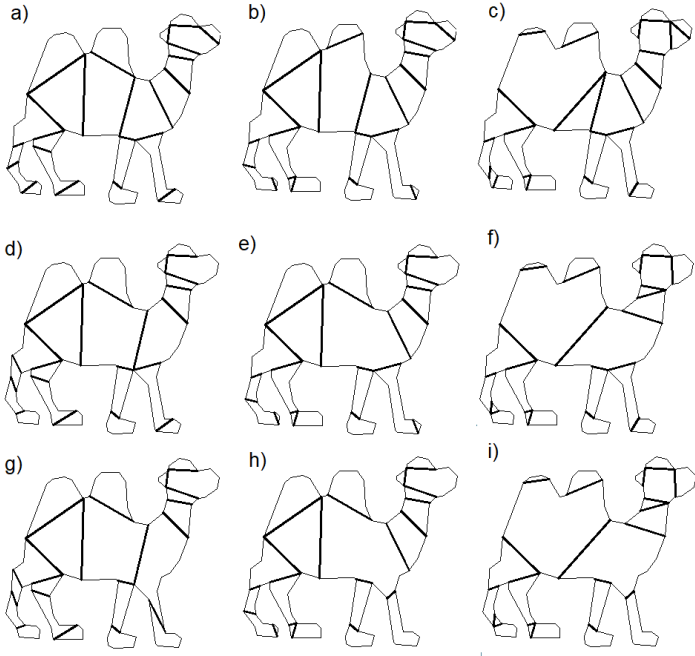


Figure 10: Decomposition results of multi-parameters

a picture, we have only used the amount of vertices that the DCE suggest as output. That is, instead of using for example 2000 vertices we only considered the 40 vertices that contribute the most to the shape.

iv. Decomposition results

In this experiment the different parameter will be changed to see if these combined effects change on the decomposition results. Here the values of the parameters will be as follows: $0R \leq \Psi \leq 0.3R$, $-0.25 \leq \lambda \leq 0.25$ and $\beta = 1$. This will result in nine decomposed pictures, and is shown in figure 10.

The summary of the different parameter values for each decomposed picture is shown in table 6. This experiment is done to determine the best values that might be used for decomposition. These values are adapted to the picture of the camel, and might differ from the other pictures. The other pictures will be tested with the same parameter values later in this section. As can be seen in the results, the decomposition of either e or f looks the most natural. That is for values $\Psi = 0.15$, $0 \leq \lambda \leq 0.25$ and $\beta = 1$. In picture e, the body of the camel is more precisely decomposed in terms of natural decomposition. In f, the camel feet and facial area are more precisely decomposed. In order to see what the effect of λ will be, λ was tested more, while keeping the other values the same. The final value found with the best decomposition is $\lambda = 25 \times 10^{-7}$. This value is very small, but shows the most promising results. For the comparison with the other methods, the following pictures was used: beetle, cat, cattle, dog, elephant, horse and rat. Here the results will be compared to the ACD, CSD and MNCD methods. The results are shown in figure 11. When looking at the results, one can

Table 6: Table showing the parameter values for each decomposed picture in figure 10.

Picture	Ψ -values	λ -values	β -value
a	0	-0.25	1
b	0	0	1
c	0	0.25	1
d	0.15	-0.25	1
e	0.15	0	1
f	0.15	0.25	1
g	0.3	-0.25	1
h	0.3	0	1
i	0.3	0.25	1

see that the new method has more decomposed parts than the other methods. This is to be expected as this method was developed to improve the amount of time it takes to decompose, and the keep the decomposing accuracy. In this case, the accuracy has decreased, but the time of decomposition has increased. This results might also be due to the fact that the Ψ value here compared to the other values are quite big, and the λ value is much smaller. These values were picked based on the results of the camel in figure ?? . These values might be different for every picture considered, and thus the decomposition results produced show unnecessary cuts.

Next characteristics will be looked at, where the most important of these are rotation- and distortion invariance. In order to test if the new method is distortion invariant, the 5-winged pentagon picture is used. This picture is then to undergo different distortions and then decomposed to see if the results are the same as that for the normal picture. The results of the rotation can be seen in figure 12.

As can be seen in the results, for every distortion, there is different decomposition results. This then leads to the conclusion that our method is not distortion invariant. This can be due to the DCE that is used to determine corners, and when distortions are applied, the DCE includes corners that could be discarded.

Next rotation invariance will be tested. The same picture will be used to conduct this test. This test was done by taking the picture, rotating it by 45° and then decomposing the shape. This was done for 0° , 45° , 90° and 135° . The results are shown in figure 13.

As can be seen in the results, the decomposition after rotation produces better results than that of the distortion. After the rotation is applied, there is one or two redundant cuts as a result of the rotation. Since it is only one or two more, it can be said that this method is rotation invariant. **Simple shape output**

For the shapes to be identified correctly another parameter, σ , is used. This parameter is used to determine the maximum distance from the contour at hand to the approximated contour.

If this value is chosen correctly, the simple shape output that

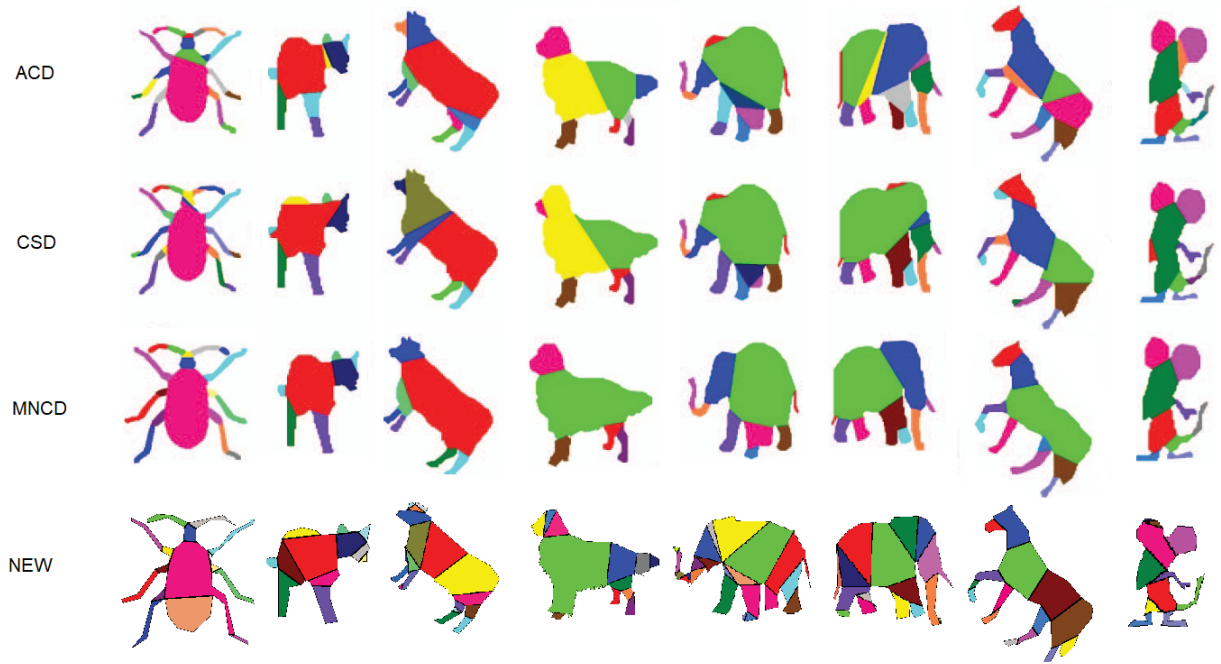


Figure 11: Decomposition comparison with methods ACD, CSD, MNCD and our method. Parameters are set to $\Psi = 0.15R$, $\lambda = 25 \times 10^{-7}$, and $\beta = 1$.

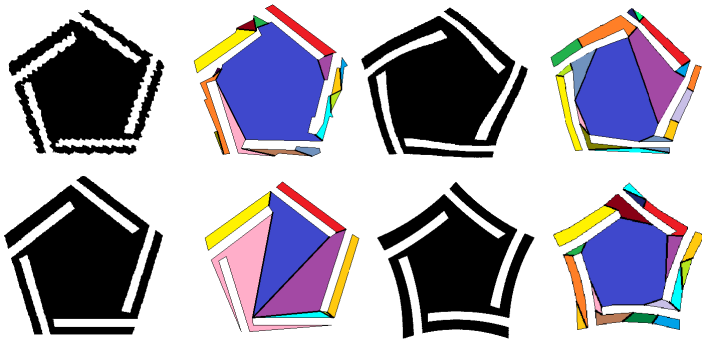


Figure 12: Decomposition results after different distortions are applied

is determined will be accurate and correct. In order to determine the optimal value, this σ value will be changed and the output is shown in figure 14. The parameter has been chosen as $\sigma = 0.006, \sigma = 0.01, \sigma = 0.025, \sigma = 0.03, \sigma = 0.06$ and $\sigma = 0.01$. As can be seen in the figure, the legend is used to colour code the shape present in the decomposition result. For example, red is used to colour triangles, yellow for quadrilaterals, green for pentagons etc. As can be seen in the figure, the smaller the value of σ , the more accurate the simple shapes represent the original decomposed image. Thus, the larger σ creates more triangular approximation of the original image, but at the same time loses the effect to accurately still be able to tell what the picture represents.

VI.CONCLUSION

i.Conclusion

The focus of this dissertation was to make an improvement on the time it takes to decompose a complex shape, while keeping the decomposition results the same. Thus previously proposed methods were looked at and compared in order to identify the method that needs the most improvement and at the same time should also be able to separate a complex shape into the minimal amount of simple parts.

This was done by doing some research and then to identify areas where shape decomposition plays an important role. After the research was done, an investigative question was formed and a literature review on what shape decomposition is, the different types of shape decomposition bases, some of the shape descriptors that is found and the time complexity of

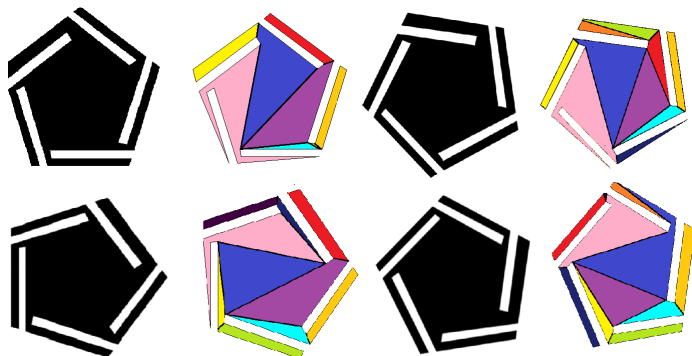


Figure 13: Decomposition results after different angles are applied

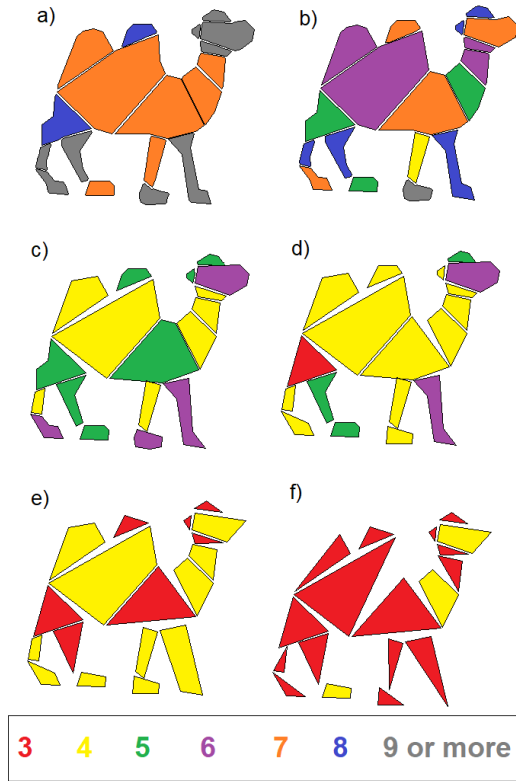


Figure 14: Simple shape output after σ has been changed to: (a) $\sigma = 0.006$, (b) $\sigma = 0.01$, (c) $\sigma = 0.025$, (d) $\sigma = 0.03$, (e) $\sigma = 0.06$ and (f) $\sigma = 0.1$

some decomposition methods, as well as the decomposition results of those methods, was done. After these methods was compared, a method was identified that could be improved. It was found that the method best to improve on is the MNCD method, as it has a high time complexity and decomposed complex shapes into the minimal number of simple shapes while having high visual naturalness. As this method makes use of Morse functions to solve, an alternative to these functions has been proposed. Instead of making use of all the vertices of the contours as well, the corners was used instead. But because corners can be defined out of many parameters, Discrete Contour Evolution(DCE) was also proposed to obtain less vertices to cut a shape from. The concept of Mutex pairs was also introduced and different methods of how to determine mutex pairs are investigated.

Lastly some experiments was done, and the results were analysed. For this dissertation, the results was analysed according to the time improvement on the MNCD method, and an average of 15% improvement was found. As to the number of parts, it was found that there was not a significant improvement, and in some cases the decomposition could have been better.

ii.Future work

As mentioned in chapter ??, the results of the time it takes to decompose a shape is improved by an average of 15.326 %. Another way to evaluate the result is to draw up a time complexity function. In order to implement the DCE, we know that the time complexity is: $O(N \log N)$, here the amount of vertices in the original shape is represented by N . Then, to obtain the $C(S)$ will take $O(v^2)$, where v is the number of corners after Harris-Corner detection is implemented. In order to compute the amount mutex pairs, the time complexity is: $O(vr)$, where the amount of notches are represented by r . In computing the A matrix $O(mn)$ is used where the amount of mutex pairs is represented by m and n the number of cuts. But because $N \gg v, r, m$ and n , we can write the time complexity as: $O(N \log N + v^2 + vr + mn) = O(N \log N)$.

If one looks at figure ??, Wang et al. has a curve of $O(N \log N)$, which can be seen as an improvement on the Ren et al. method, because the time complexity increase at a much lower rate that that of Ren. In order to further improve on the speed of this method, the DCE method can be revised and improvements brought to. One suggestion would be to instead of running the DCE algorithm twice, do it once and automatically determine a stop criteria while going through the algorithm.

The next possible improvement that can be brought up is an improvement on how to determine the Mutex pairs. The Morse functions take up a lot of time, and using only the corner points is too simple, and thus the mutex pairs can have a great influence on the decomposition results. A suggestion that I would make is to obtain a medial axis, and through doing that, obtain mutex areas, instead of points, that will describe possible areas of decomposition, and any point closer to a midpoint of an area, but be described accordingly. Another improvement that can be implemented, is invariant improvements, as the experiment results show that this method is not distortion invariant. In order to improve on these properties, more research needs to be done on why they are variant and how to ensure that it will stay invariant.

REFERENCES

- [1] Y. Lu, J. M. Lien, M. Ghosh, and N. M. Amato, " α -Decomposition of polygons," *Computers and Graphics (Pergamon)*, vol. 36, no. 5, pp. 466–476, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.cag.2012.03.018>
- [2] Z. Ren, J. Yuan, and W. Liu, "Minimum near-convex shape decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2546–2552, 2013.
- [3] J. De Winter and J. Wagemans, "Segmentation of object outlines into parts: A large-scale integrative study," *Cognition*, vol. 99, no. 3, pp. 275–325, 2006.
- [4] G. Liu, Z. Xi, and J. M. Lien, "Dual-space decomposition of 2D complex shapes," *Proceedings of the IEEE*

- Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4154–4161, jun 2014. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909925>
- [5] H. Liu, W. Liu, and L. J. Latecki, "Convex shape decomposition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. May 2014, pp. 97–104, 2010.
- [6] D. D. Hoffman and M. Singh, "Saliency of visual parts," *Cognition*, vol. 63, no. 1, pp. 29–78, 1997.
- [7] J. Erickson and S. Har-Peled, "Optimally Cutting a Surface into a Disk," in *Discrete and Computational Geometry*, vol. 31, no. 1. New York, New York, USA: ACM Press, 2004, pp. 37–59. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=513400.513430>
- [8] Z. Ren, J. Yuan, C. Li, and W. Liu, "Minimum near-convex decomposition for robust shape representation," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 303–310, 2011.
- [9] C. Wang, Z. Lai, and H. Wang, "Hand gesture recognition based on perceptual shape decomposition with a Kinect camera," *IEICE Transactions on Information and Systems*, vol. E96-D, no. 9, pp. 2147–2151, 2013.
- [10] K. Leonard, G. Morin, S. Hahmann, and A. Carlier, "A 2D shape structure for decomposition and part similarity," in *Proceedings - International Conference on Pattern Recognition*, 2017, pp. 3216–3221.
- [11] I. Biederman, "Recognition by components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 2, pp. 115–117, 1987.
- [12] A. V. Naik, K. W. Regan, and D. Sivakumar, "On Quasilinear Time Complexity Theory," SUNY - Buffalo, Tech. Rep., 1999. [Online]. Available: <https://cse.buffalo.edu/~regan/papers/pdf/NRS95.pdf>
- [13] H. Rom and G. Médioni, "Hierarchical Decomposition and Axial Shape Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 973–981, 1993.
- [14] J.-M. Lien, J. Keyser, and N. M. Amato, "Simultaneous shape decomposition and skeletonization," *Proceedings of the 2006 ACM symposium on Solid and physical modeling - SPM '06*, p. 219, 2006. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1128888.1128919>
- [15] J. M. Lien and N. M. Amato, "Approximate convex decomposition of polyhedra and its applications," *Computer Aided Geometric Design*, vol. 25, no. 7, pp. 503–522, 2008.
- [16] S. Velasco-forero, S. V.-f. I.-c. Opening, and A. M. Morphology, "Inner-Cheeger Opening and Applications To cite this version : HAL Id : hal-01250280," Mines Paris Tech, Paris, Tech. Rep. May 2015, 2016.
- [17] M. R. Daliri and V. Torre, "Classification of silhouettes using contour fragments," *Computer Vision and Image Understanding*, vol. 113, no. 9, pp. 1017–1025, 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.cviu.2009.05.001>
- [18] E. H. Cohen and M. Singh, "Geometric determinants of shape segmentation: Tests using segment identification," *Vision Research*, vol. 47, no. 22, pp. 2825–2840, 2007.
- [19] F. Aurenhammer, "Weighted skeletons and fixed-share decomposition," *Computational Geometry: Theory and Applications*, vol. 40, no. 2, pp. 93–101, 2008.
- [20] T. Jiang, Z. Dong, C. Ma, and Y. Wang, "Toward perception-based shape decomposition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7725 LNCS, no. PART 2, pp. 188–201, 2013.
- [21] T. Biedl, M. Held, S. Huber, D. Kaaser, and P. Palfrader, "Reprint of: Weighted straight skeletons in the plane," *Computational Geometry: Theory and Applications*, vol. 48, no. 5, pp. 429–442, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.comgeo.2015.01.004>
- [22] Y. Lu, "Path Planning in Similar Environments," Ph.D. dissertation, Dalian University of Technology, 2013.
- [23] D. Avis and G. T. Toussaint, "An efficient algorithm for decomposing a polygon into star-shaped polygons," *Pattern Recognition*, vol. 13, no. 6, pp. 395–398, 1981.
- [24] Y. Mingqiang, K. Kidiyo, and R. Joseph, "A Survey of Shape Feature Extraction Techniques," *Pattern Recognition Techniques, Technology and Applications*, 2008.
- [25] A. Sharma, "Time and Space Complexity Tutorials & Notes | Basic Programming | HackerEarth," 2016. [Online]. Available: <https://www.hackerearth.com/practice/basic-programming/complexity-analysis/time-and-space-complexity/tutorial/>
- [26] C. Wang, W. Liu, Z. Lai, and H. Wang, "Perceptually friendly shape decomposition by resolving segmentation points with minimum cost," *Journal of Visual Communication and Image Representation*, vol. 24, no. 3, pp. 270–282, 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.jvcir.2013.01.001>
- [27] Y. Lu, J. M. Lien, M. Ghosh, and N. M. Amato, " α -Decomposition of polygons," *Computers and Graphics (Pergamon)*, vol. 36, no. 5, pp. 466–476, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.cag.2012.03.018>

- [28] M. Keil and J. Snoeyink, "on the Time Bound for Convex Decomposition of Simple Polygons," *International Journal of Computational Geometry & Applications*, vol. 12, no. 03, pp. 181–192, 2002.
- [29] L. Latecki and R. Lakämper, "Polygon evolution by vertex deletion," *SCALE-SPACE '99 Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pp. 398–409, 2010.
- [30] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proceedings of the Alvey Vision Conference 1988*, pp. 23.1–23.6, 1988. [Online]. Available: <http://www.bmva.org/bmvc/1988/avc-88-023.html>
- [31] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed., M. Loukides, Ed. Sebastopol: O'Reilly Media, 2008.
- [32] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 35, no. 1, pp. 13–32, 1999.
- [33] M. Kline, "Calculus: An Intuitive and Physical Approach," in *Calculus: An Intuitive and Physical Approach*. Dover Publications, 1998, p. 960.

Appendix B

Human Perception Experiment

B.1 Human Perception Experiment

In this appendix, the human perception experiment will be discussed. It was decided to do a human perception experiment in order to determine the quality of a shape decomposition algorithm, and to evaluate the results of the new algorithm. In order for this experiment to be successful, the scientific method will be followed and a short investigation will be done to obtain the results.

B.1.1 The problem

The problem at hand is that there is no qualitative way to evaluate the quality of a shape decomposition algorithm. This is mostly due to the fact that shape decomposition algorithms try to decompose shapes into the most natural way that humans would decompose a shape into. The problem here lies with the fact that human perception differs from human to human, and is therefore very hard to define. Thus, in order to create a more concrete way to determine if a shape decomposition algorithm is more likely to decompose a complex shape like a human would, human perception needs to be investigated. This then leads us to the investigative question.

B.1.2 Investigative question

Can the quality of an algorithm be determined by looking at deviations from human shape decomposition?

B.1.3 Hypothesis

The quality of an algorithm can be determined by looking at deviations from humans shape decomposition.

B.1.4 Experiments

Questionnaire

In order to determine the cuts that can be considered as that of human perception, a questionnaire was set-up. In this questionnaire, several objects of the MPEG-7 dataset was placed on paper and candidates were asked to decompose these complex shapes into simple primitive shapes. The questionnaires were given to a group of people representing the population of South-Africa. An example of the questionnaire is shown in figure B.1. Once completed, every questionnaire was taken and all of the information recorded. The information extracted is the geological information, the human perception cuts and the number of parts after shape decomposition. Each one of these topics will be discussed next.

Shape Decomposition Questionnaire

Name: _____

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Figure B.1: *Picture showing the questionnaire to evaluate human perception.*

B.1.5 Capture of results

Geological information

In this subsection the geological information will be discussed. This is done to ensure fair results, and to provide an estimate of the expected population. In total there was a 100 questionnaires answered. Of the 100 questionnaires, 80 % were African, 8 % Caucasian, 3 % Indian, 1 % Asian and 8% were other races. This closely resembles the South-African population which is 79 % African, 9% Caucasian, 2 % Indian, 1 % Asian and 9 % other races [1]. To summarize the geological information for this questionnaire, the results are shown in pie charts in figure B.2 below. The

gender is misrepresenting the population in that the questionnaire asked 68 % females and 32 % males, whereas the population is 51 % female and 49 % male. Lastly, the age groups are closely related to the population in that the age groups 0-14 has 31 %, 15-64 has 65 % and > 65 has 4 % compared to the population where 0-14 is 31 %, 15-64 is 64 % and > 65 is 5 %. Thus from these statistics it can be seen that the questionnaire group closely represents the population and can be used to draw informative conclusion as to how humans decompose complex shapes.

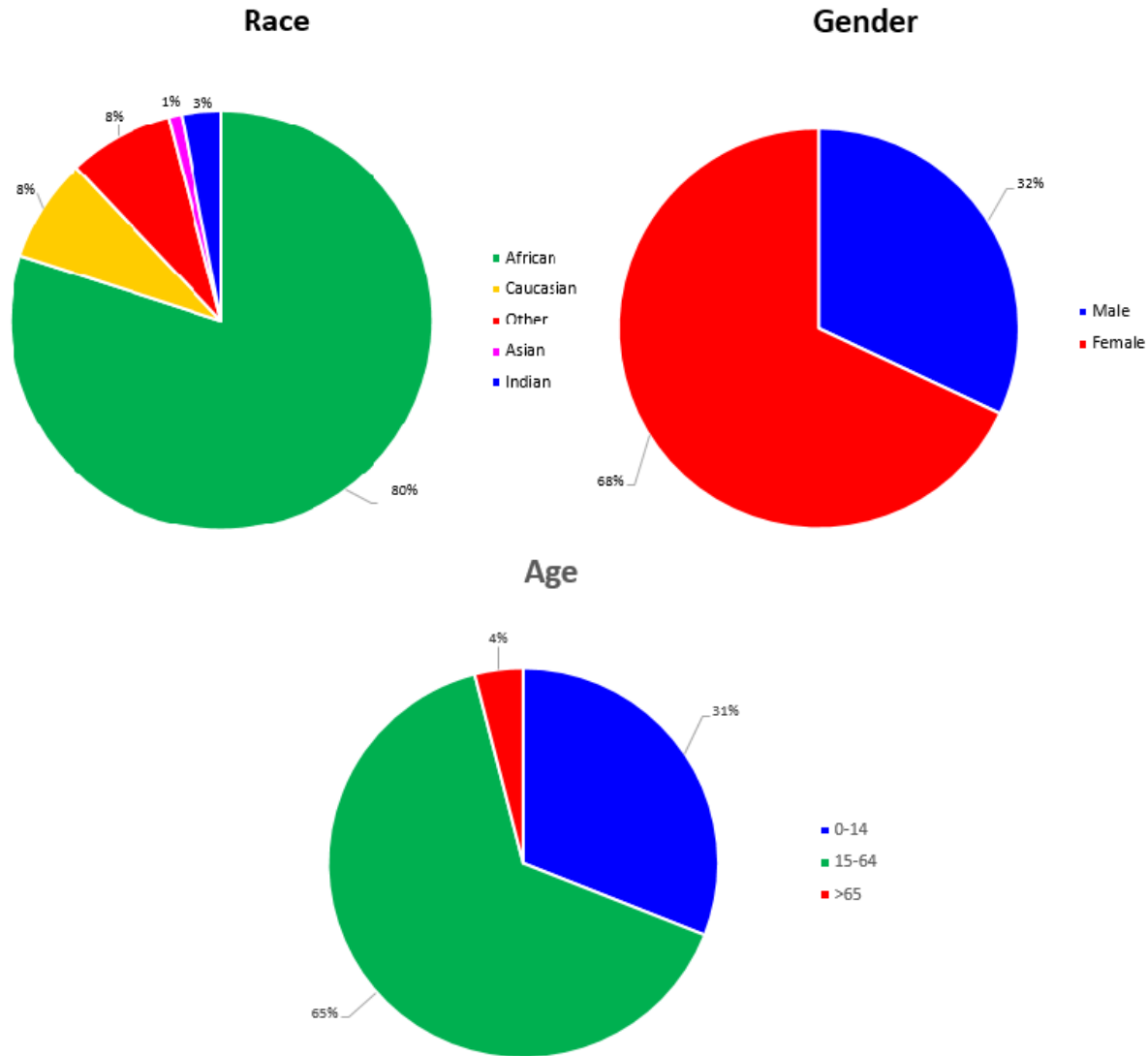


Figure B.2: Picture showing the questionnaire to evaluate human perception.

Human perception cuts

For every picture in the questionnaire the same picture, described as the main picture, was used to record the cuts of every questionnaire on it. Each cut is drawn in very low transparency, and thus will cause any repeated cuts to become less transparent and more visible. Therefore, the cuts

with the least transparency will be chosen as the final human perception cuts. The results will be shown below with a short discussion of each.

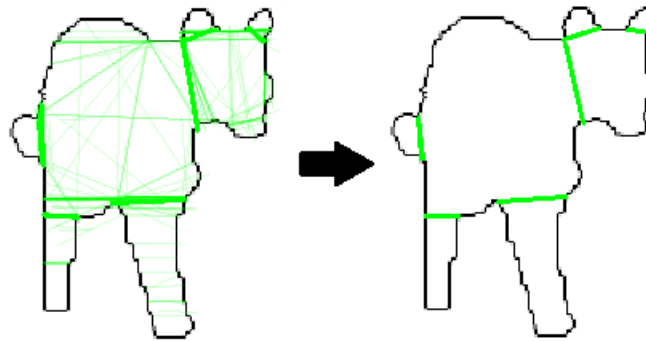


Figure B.3: *Picture showing the questionnaire to evaluate human perception.*

The first picture is of a cat and the results are shown in figure B.3. As can be seen in the picture, a lot of cuts of different parts are given. The final chosen cuts are shown to the right as these cuts where the most prominent of all the cuts on the cat. As can be seen, the ears, head, tail, body and legs have been decomposed, which is an expected result when human perception is 'thought of'.

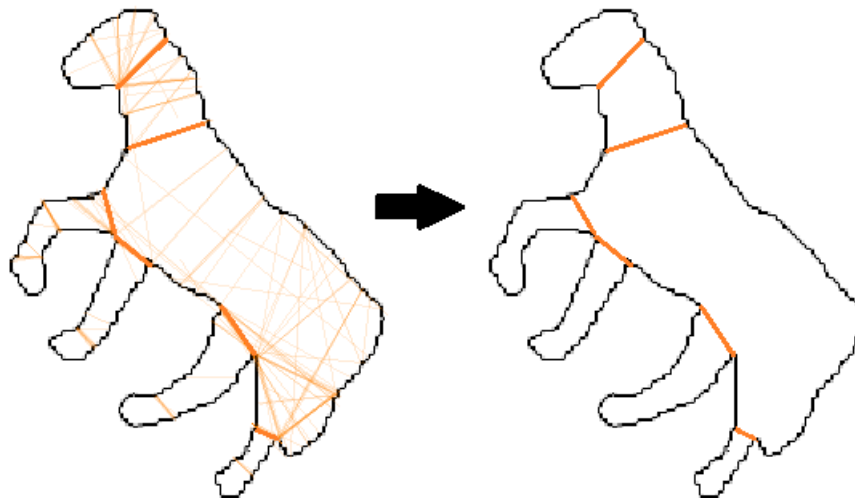


Figure B.4: *Picture showing the questionnaire to evaluate human perception.*

The second picture is of a horse and the results are shown in figure B.4. The final chosen cuts are where the most prominent of all the cuts as can be seen. As can be seen the head, neck, body and legs have been decomposed, which is an expected result when human perception is 'thought of'. The tail and the hooves however was not prominent cuts, but can also be expected to be shown as a human perception result.

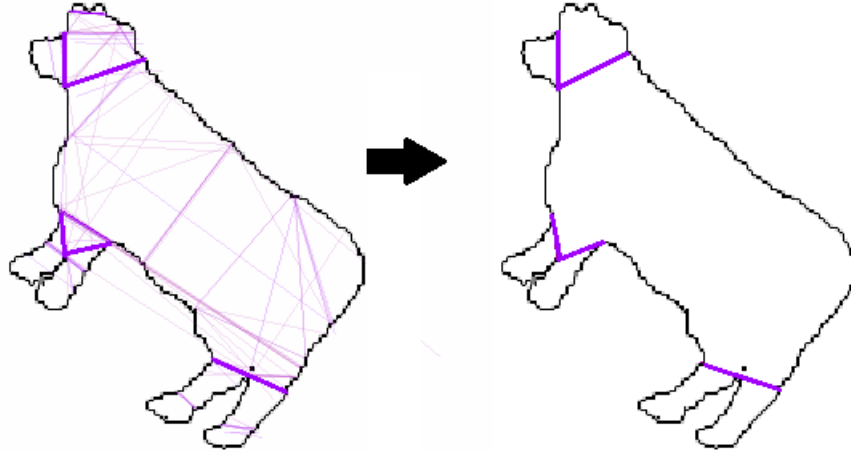


Figure B.5: *Picture showing the questionnaire to evaluate human perception.*

The third picture is of a cow and the results are shown in figure B.5. The final chosen cuts are where the most prominent of all the cuts as can be seen. As can be seen the head, mouth, body and legs have been decomposed, which is an expected result when human perception is 'thought of'. The front legs show three possible cuts, but it has been decided to leave the top cut as it was not as prominent as the other two, and it will result in an unnecessary small part.

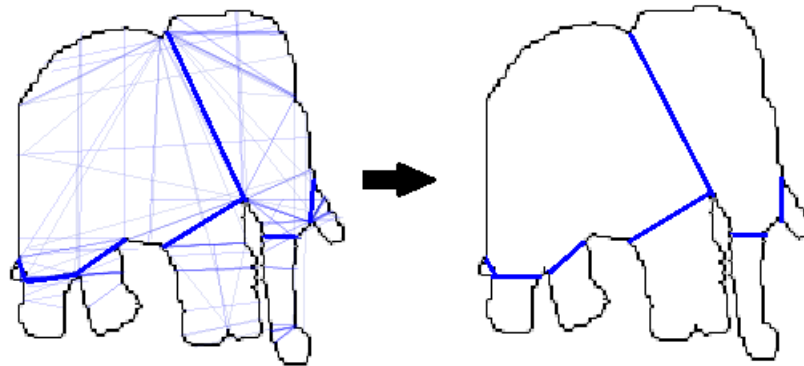


Figure B.6: *Picture showing the questionnaire to evaluate human perception.*

The forth picture is of a elephant and the results are shown in figure B.6. The final chosen cuts are where the most prominent of all the cuts as can be seen. Also the head, trunk, tusks, body, tail and legs have been decomposed, which is an expected result when thinking of human perception. The front legs in this case one can think would be split as this is what is expected, but the results show that most humans tend to decompose the front legs as a while part.

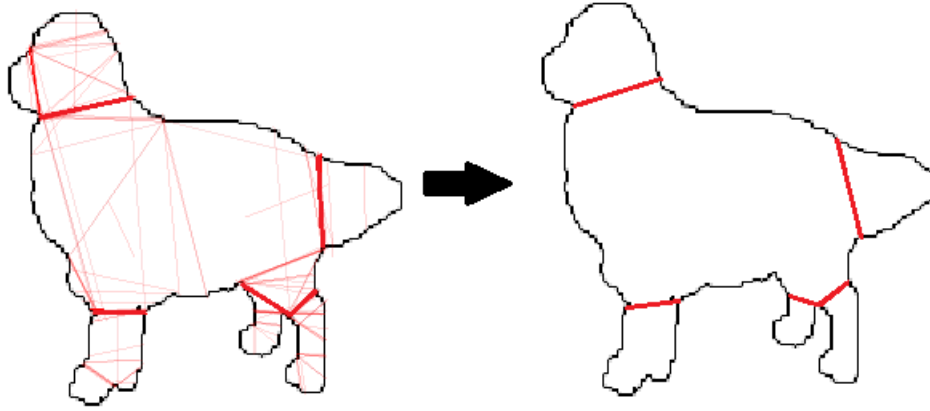


Figure B.7: *Picture showing the questionnaire to evaluate human perception.*

The fifth picture is of a dog and the results are shown in figure B.7. The final chosen cuts are selected as the most prominent of all the cuts that can be seen. As can be seen the head, body, tails and legs have been decomposed, which is an expected result when human perception is 'thought of'. The mouth might have also been an expected part, but due to the other cuts being more prominent, this was left out.

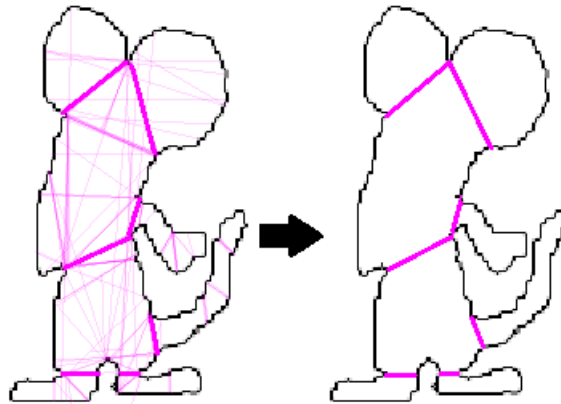


Figure B.8: *Picture showing the questionnaire to evaluate human perception.*

The sixth picture is of a mouse and the results are shown in figure B.8. The final chosen cuts are selected as the most prominent of all the cuts that can be seen. As can be seen the head, body, tail, arm, ears and legs have been decomposed, which is an expected result when human perception is applied.

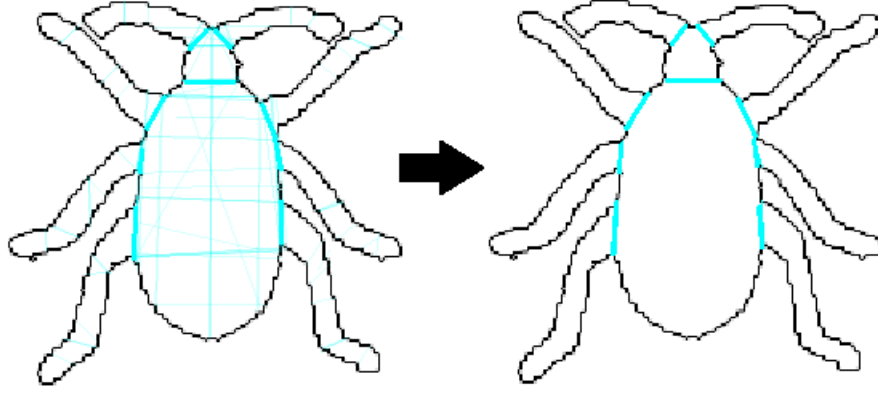


Figure B.9: *Picture showing the questionnaire to evaluate human perception.*

The seventh picture is of a bug and the results are shown in figure B.9. The final chosen cuts are selected as the most prominent of all the cuts that can be seen. As can be seen the head, body, antennas and legs have been decomposed, which is an expected result when human perception is applied.

As can be seen from the results, there exist a lot of possible cuts, and these differ with each questionnaire. Thus it can be concluded that shape decomposition done by human perception is a difficult concept to illustrate as each person would do it differently. Hence this questionnaire to see which cuts are more prominent and most likely to be chosen when a human is asked to decompose an object. These results show that there are general areas that most people tend to cut an object - especially animal object- into , which includes mostly legs, arms, head and body. These results can now be used to determine when an algorithm is close to human perception or not. To do this, the amount of cuts of an algorithm that lie on more or less the same area then the cuts of human perception will be considered a cut that agrees with human perception. Thus, the total amount of cuts that then agrees to human perception subtracted from the total amount of human perception cuts, divided by the total amount of human perception cuts times 100 will give a percentage of cuts that the algorithm being tested agrees with that of human perception cuts, and thus a accuracy measure. That is:

$$\%accuracy_{hp} = \frac{|c_{hp} - c_a|}{c_{hp}} \times 100 \quad (B.1)$$

where c_{hp} represents the total number of human perception cuts and c_a represents that total number of cuts of the algorithm that agree to the human perception cuts. Therefore as can be seen, the higher this percentage, the greater the algorithm relates to human perception.

Number of parts

In this subsection, the total number of parts after decomposition will be discussed. In order to evaluate the number of parts, the number of parts produced of each picture in each questionnaire was recorded and the average and the standard deviation of each picture was calculated. The summary of this is given in table B.1 below.

Table B.1: Table showing the average and standard deviation of the number of parts after decomposition of human perception

Average and standard deviation of number of parts							
	Cow	Elephant	Dog	Mouse	Bug	Cat	Horse
Average	7	8	6	8	10	7	7
Std	2	1	1	2	3	1	2

As can be seen from the table, the picture of the bug has the highest number of standard deviation, which indicates that of all of the pictures this picture is most likely to produce a lot more or a lot less parts than the average number of parts. This also then indicates that this picture will most likely produce different decomposition results from person to person. The other results show rather promising standard deviations and can thus be used as a measure to determine how far an algorithm deviates from the human perception decomposition.

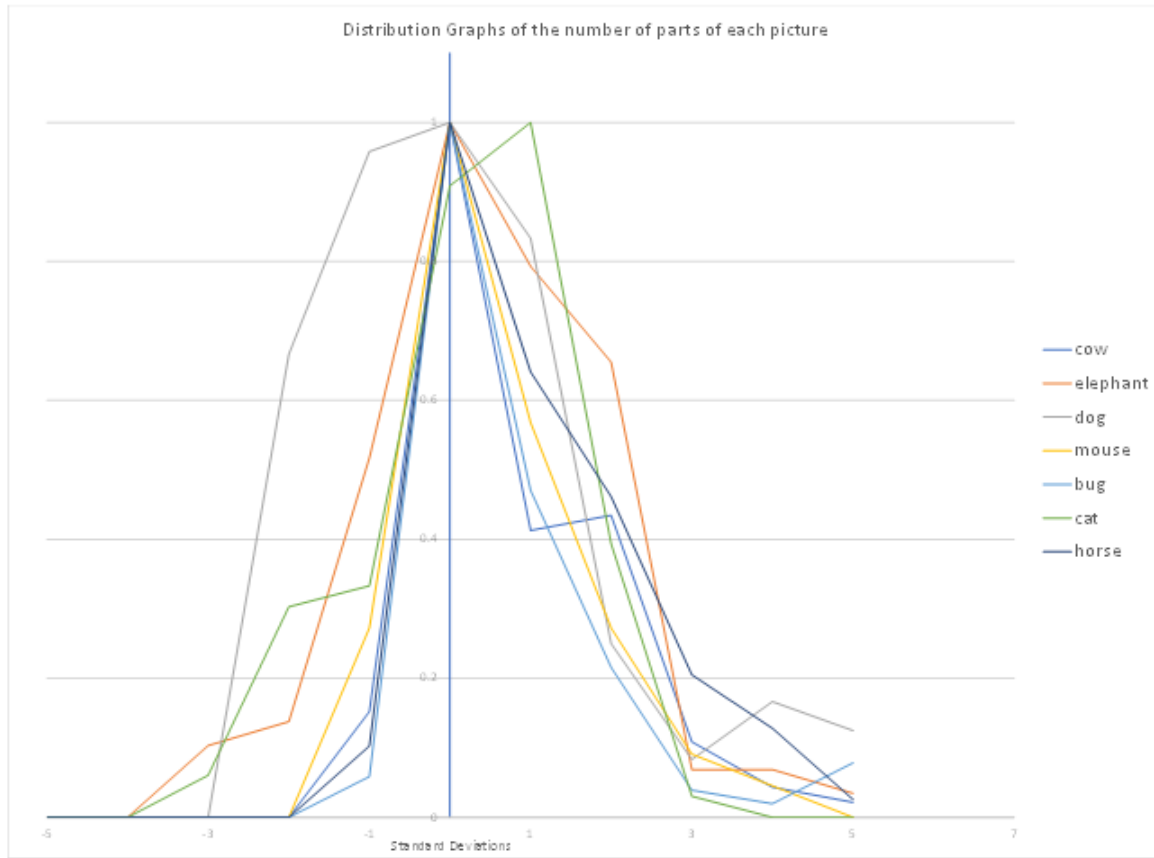


Figure B.10: Picture showing the questionnaire to evaluate human perception.

To evaluate the quality of the questionnaires, the normalization curves of each picture was drawn to determine if the data that is collected lies around the norm. The results of the graph

can be seen in figure B.10. As can be seen, 6 out of the 7 pictures' results can be considered as normally distributed, where the cat lies a bit of centre. This is most probably due to one or two questionnaires where the cat was decomposed into a lot of parts or only a few parts as opposed to the average 7 parts that is expected. Thus, as can be seen from these results, it can be justified that the number of parts produced can be a good criterion to determine the quality of an algorithm.

Thus, as a measure of how accurate an algorithm relates to human perception, another measure would be to count the number of parts produced after decomposition, and then determine how many standard deviations the results differ from the human perception results. In general, a standard deviation of more then four is classified as not being a good measure, as statistically 99.9 % of the results should lie within 3 standard deviations from the mean or the average value. Therefore, to test how closely an object from the picture relates to human perception, one can simply count the number of parts, and determine how many standard deviations it is from the mean number of human perception. Therefore mathematically it can be determined: perception cuts, and thus a accuracy measure. That is:

$$standarddeviation_{al} = \frac{|NOP_{hp} - NOP_a|}{std_{hp}} \quad (B.2)$$

where NOP_{hp} is the average number of parts produced after shape decomposition by human perception, NOP_a is the number of parts produced by the algorithm the is begin tested and std_{hp} which is the standard deviation which is determined by this questionnaire. This value should be rounded and the value obtained will indicate the number of standard deviations the current algorithm deviates from human perception.

B.1.6 Conclusion

In this experiment the human perception was looked in order to determine how humans will decompose complex shapes. This was done to determine the quality of algorithms and to have a measure of how close an algorithm can relate to human perception. Now that human perception results have been obtained there is a way of measuring the quality of the algorithms produced as well as to compare result obtained out of this dissertation.

Appendix C

Questionnaires

Shape Decomposition Questionnaire

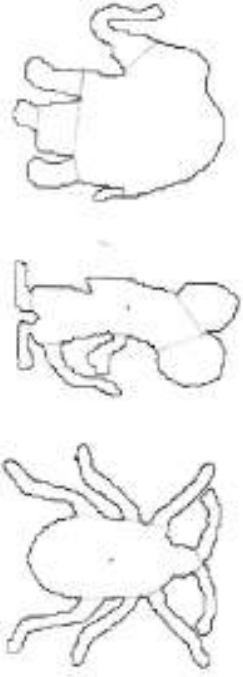
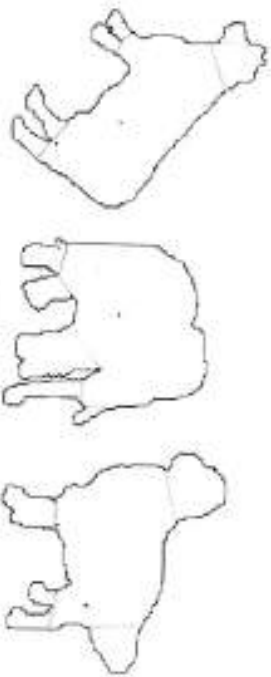
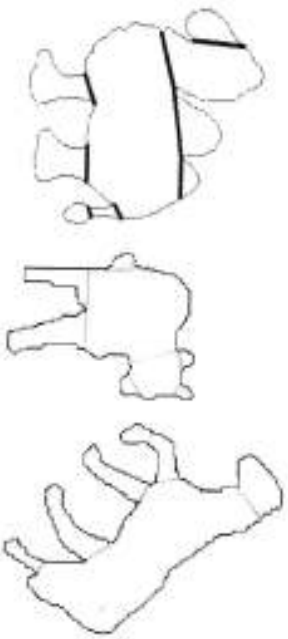
Name: Siapho Kanya

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Ladique Xanana

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

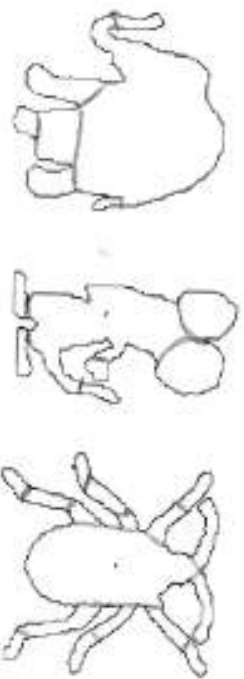
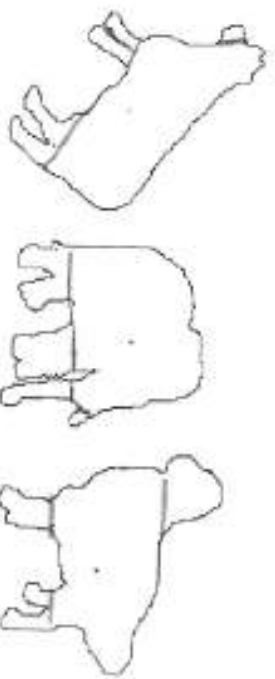
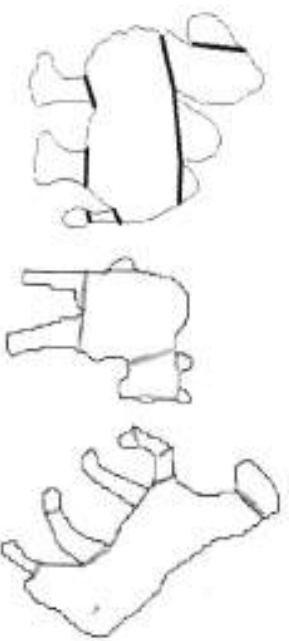
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: S Theodora Nette

Age: 14

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

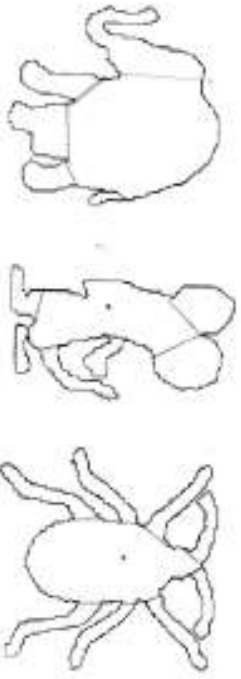
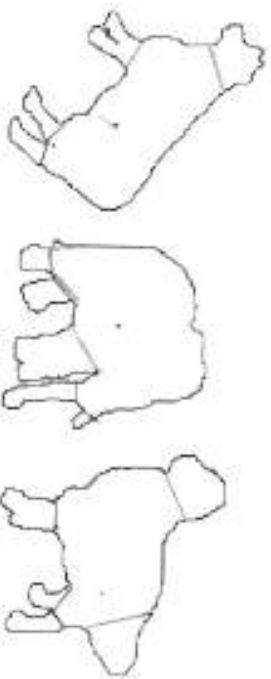
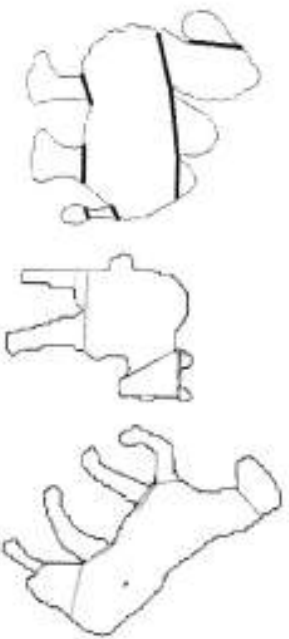
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

VaneSSa

Age:

☒ ≤20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

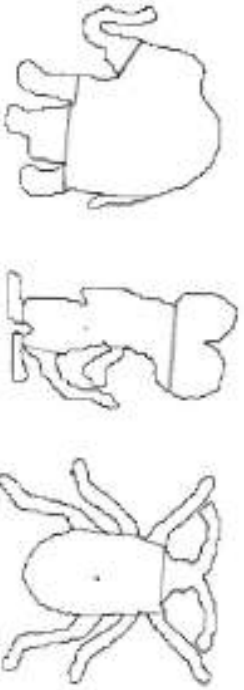
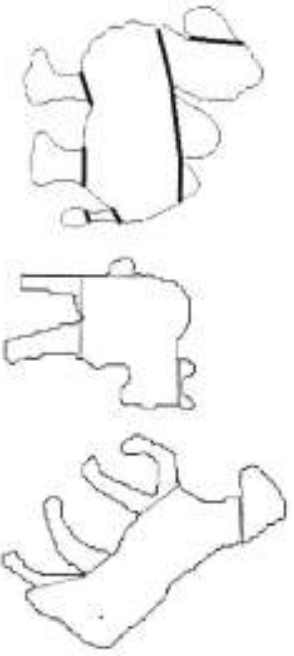
Gender:

☐ Male ☒ Female

Race:

☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

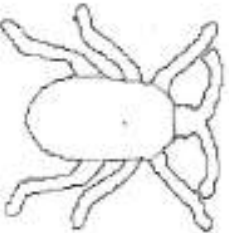
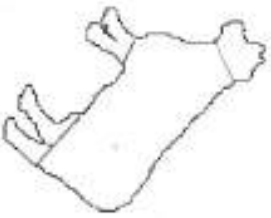
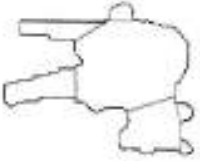
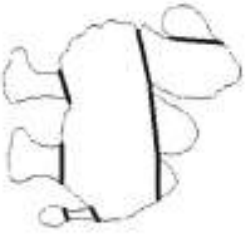
Name: Nogilio Wiles

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

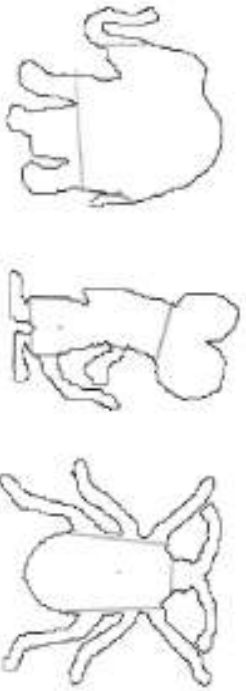
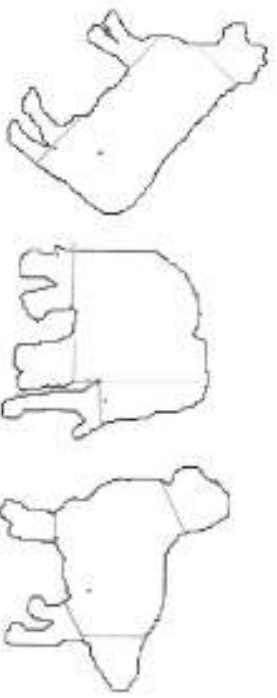
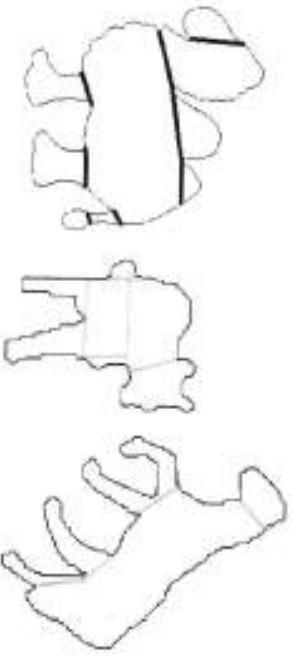
Name: Kennetho. Moroti

Age: 30 | 20-30 | 30-40 | 40-50 | 50-60 | >60

Gender: Male | Female

Race: African | Indian | Caucasian | Asian | Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Amelia Cross

Age:

24 | 20-30 | 30-40 | 40-50 | 50-60 | >60

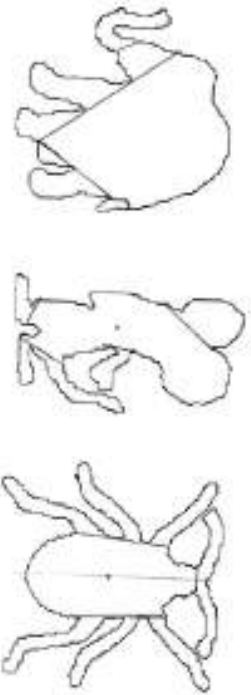
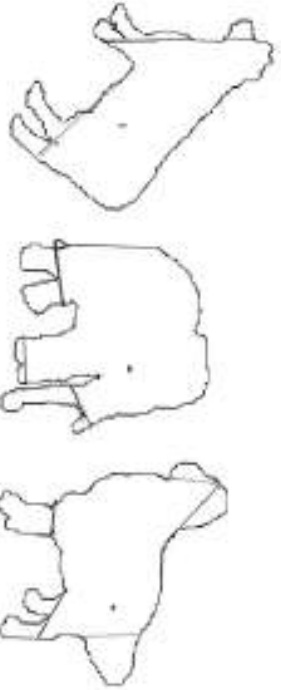
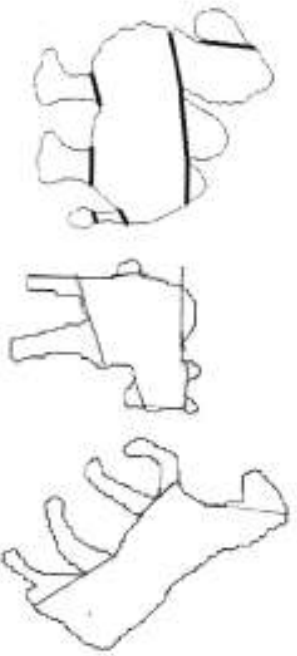
Gender:

Male | Female

Race:

African | Indian | Caucasian | Asian | Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

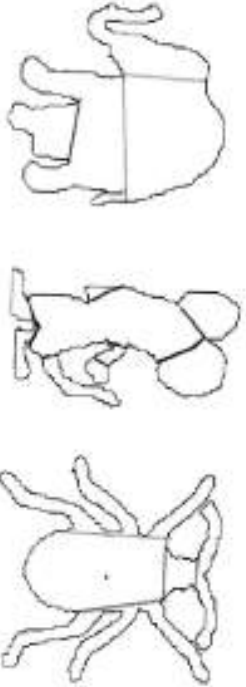
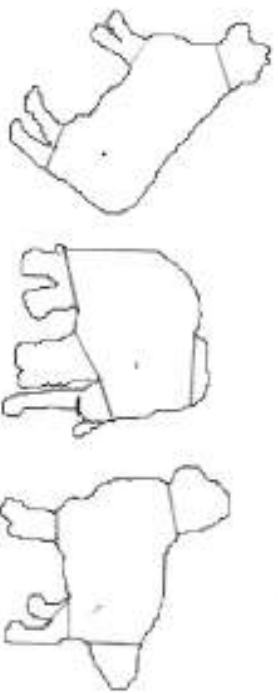
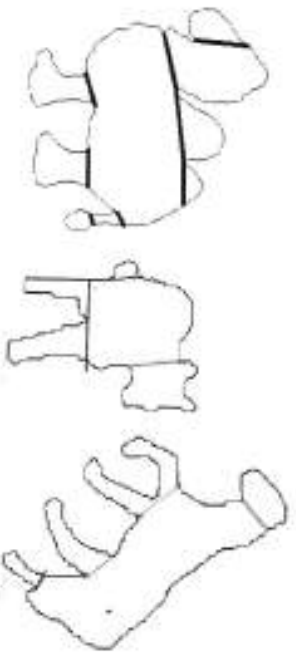
Name: SEGUNO TO MASHITSHI

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Long Mahtijir

Age:

<20 20-30 30-40 40-50 50-60 >60

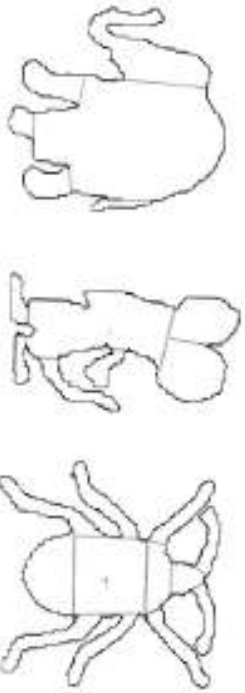
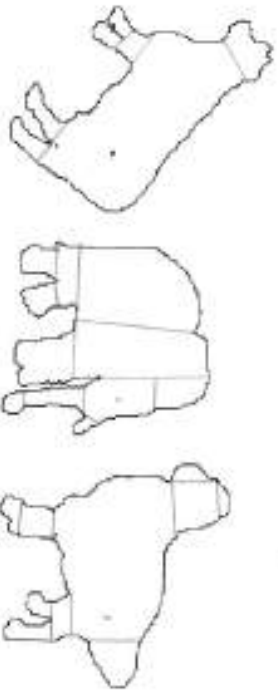
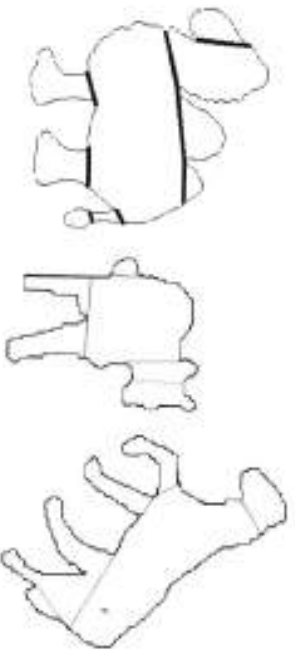
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

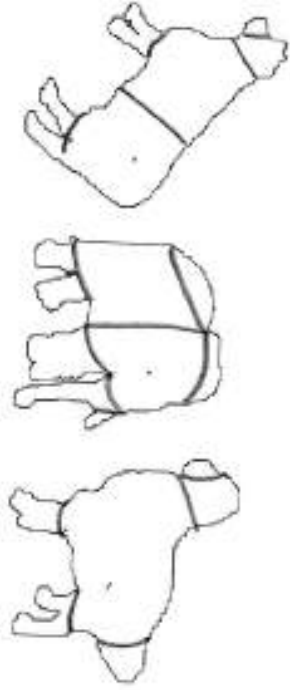
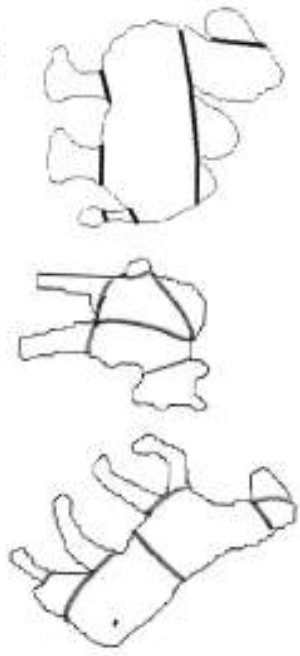
Name: Timothy McFeters

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Pulane Evelyn

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

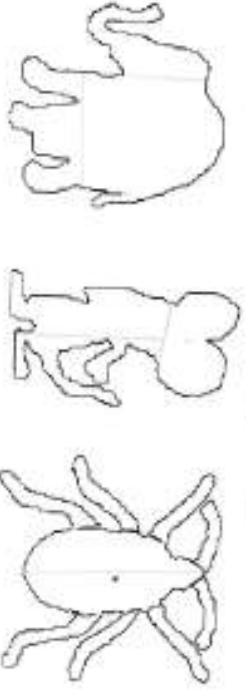
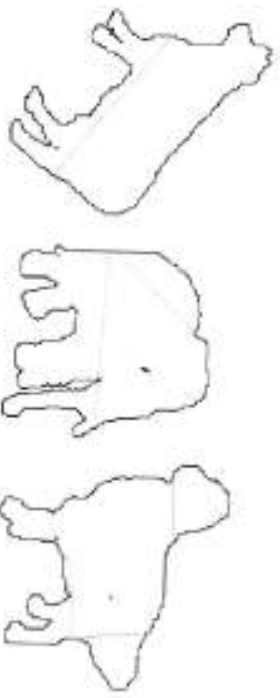
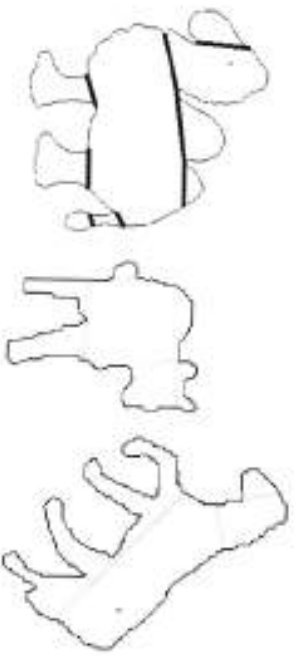
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

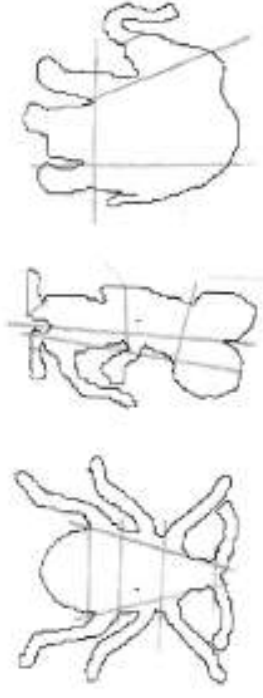
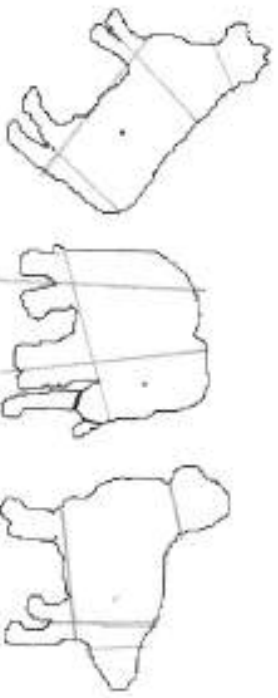
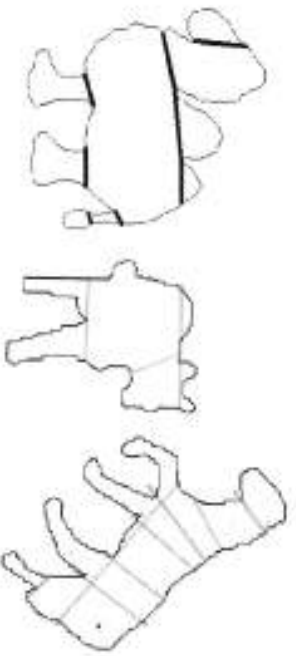
Name: Sisile Mampitso

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Vanishetha Kuriya

Age:

☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

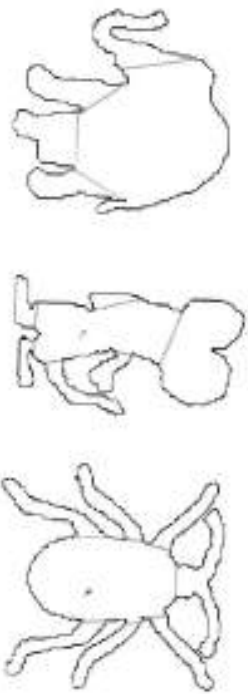
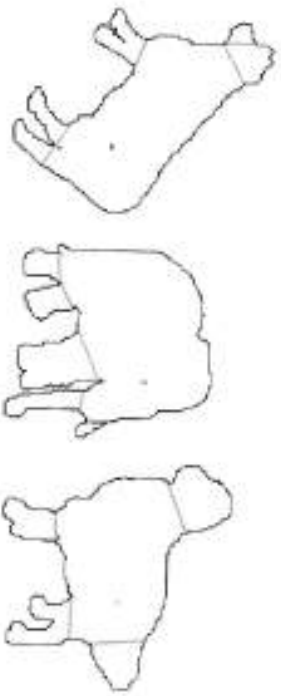
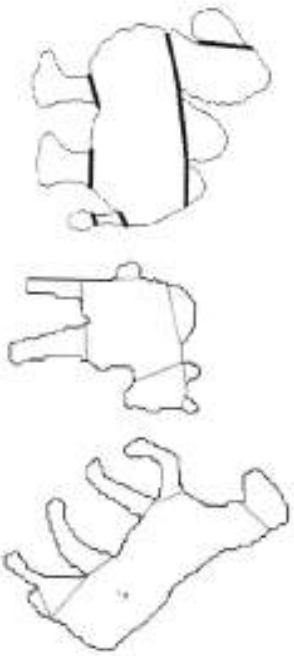
Gender:

☒ Male ☐ Female

Race:

☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Crebetse Maphangela

Age:

<20 20-30 30-40 40-50 50-60 >60

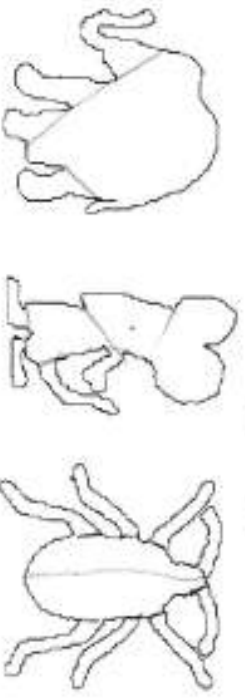
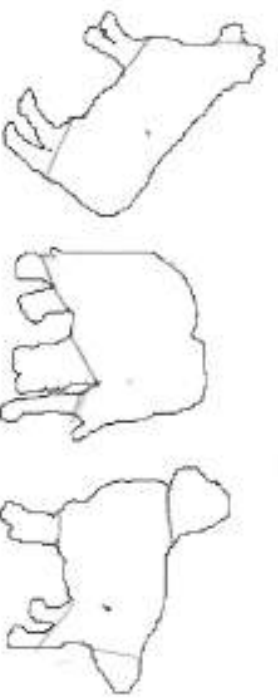
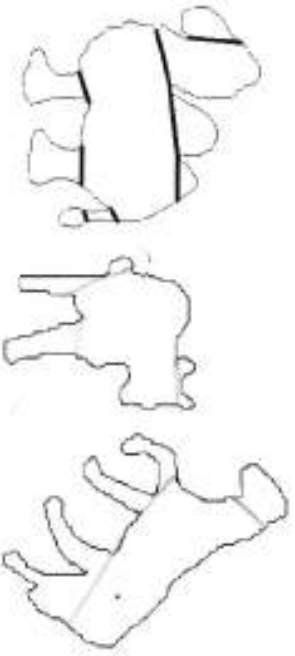
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

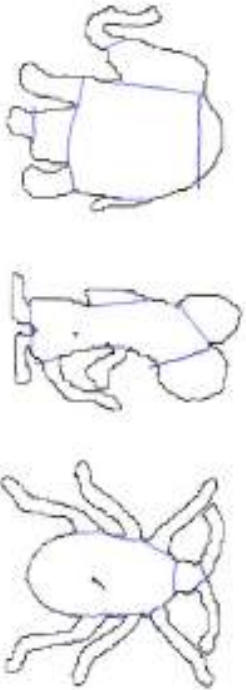
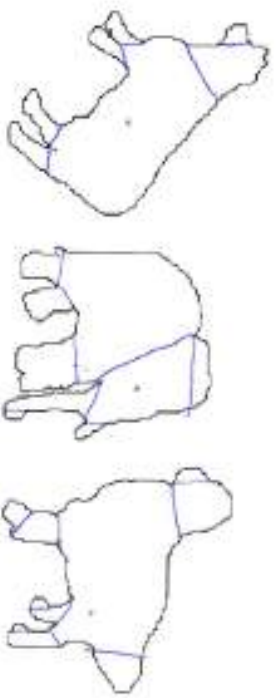
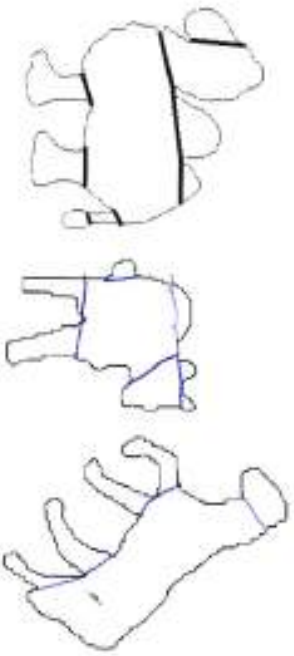
Name: Lepelel Momo

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

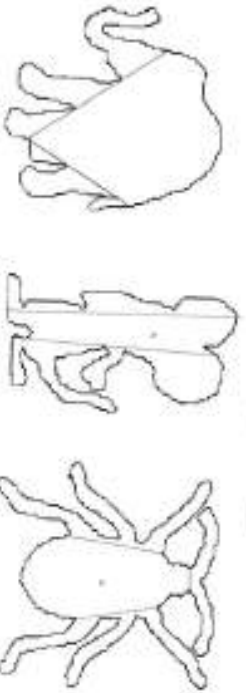
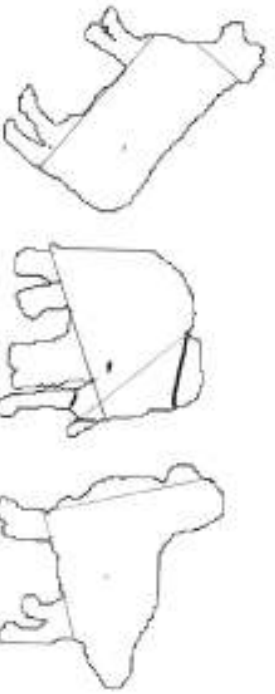
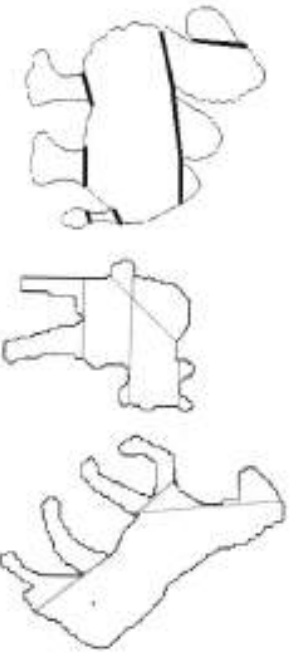
Name: Miriam Foles

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☐ African ☐ Indian ☒ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

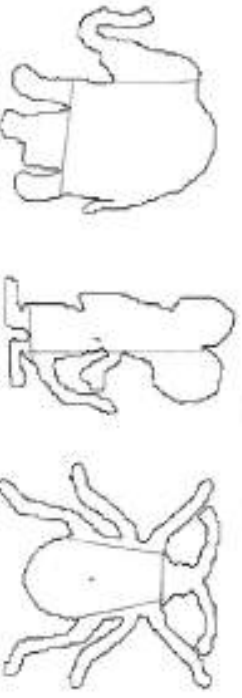
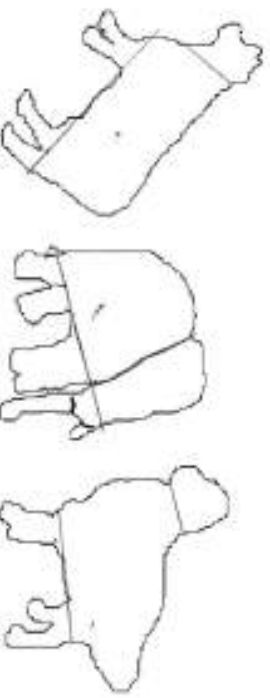
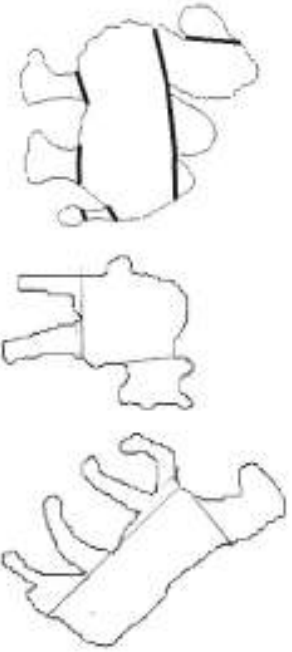
Name: Jashna Krugwagen

Age: ☒ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☒ Male ☐ Female

Race: ☐ African ☐ Indian ☒ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

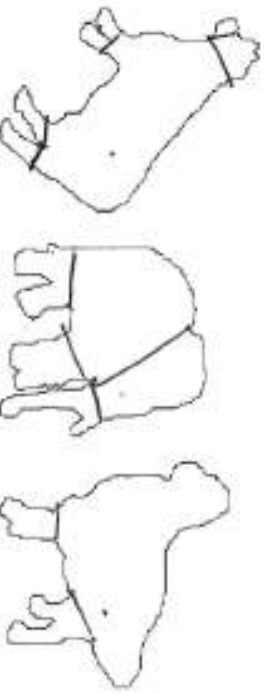
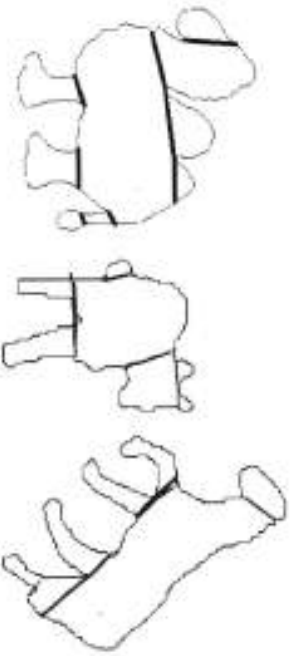
Name: Nikolaos Adaro

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Delisa Nagayuki

Age:

☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

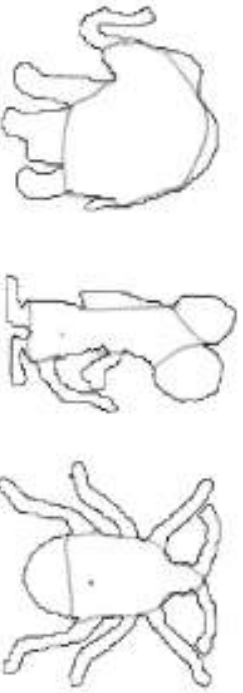
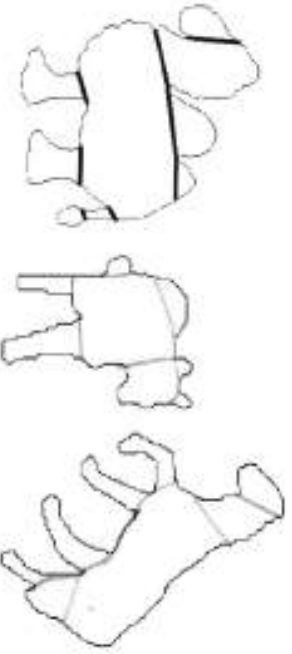
Gender:

☐ Male ☒ Female

Race:

☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

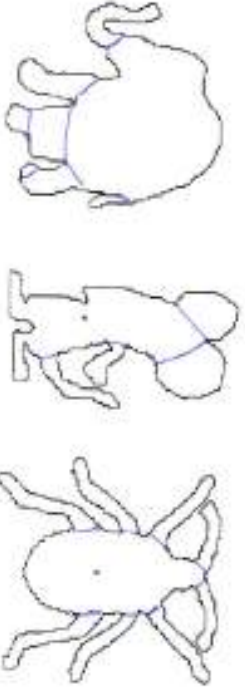
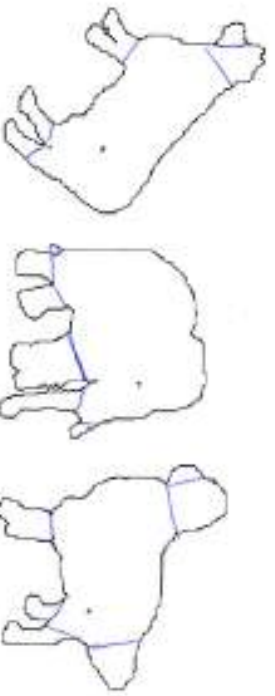
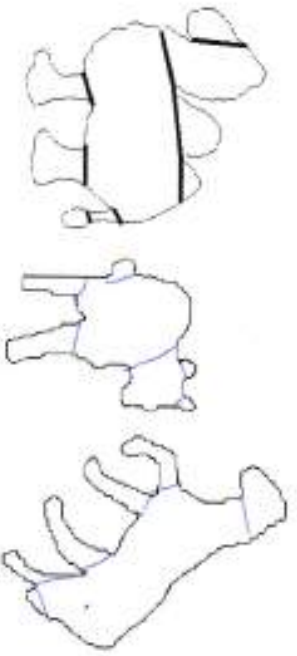
Name: Samira Vaughan Nyongosa

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☒ Male ☐ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

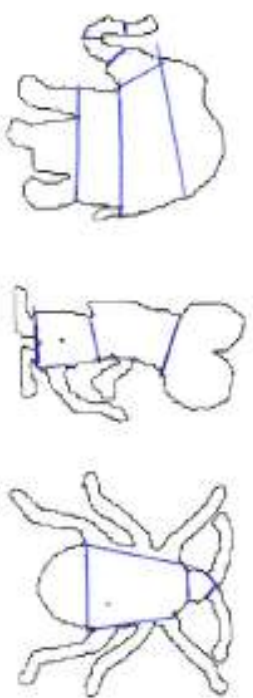
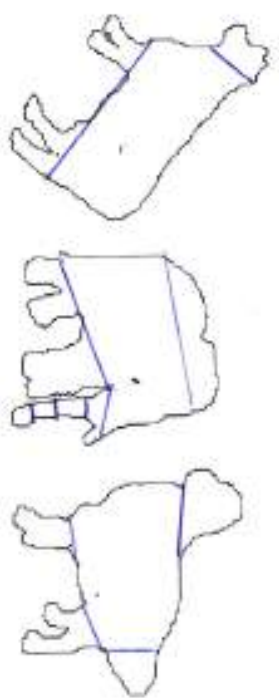
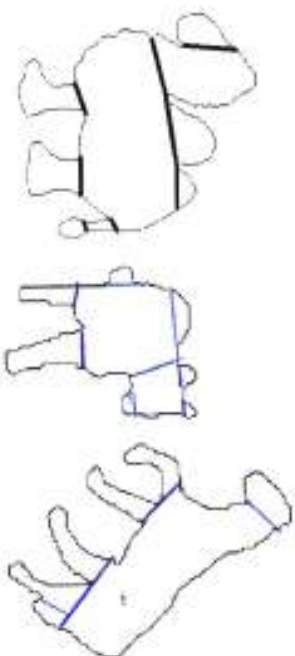
Name: Elika Mashine

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

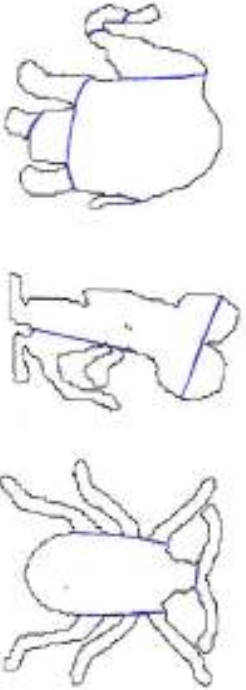
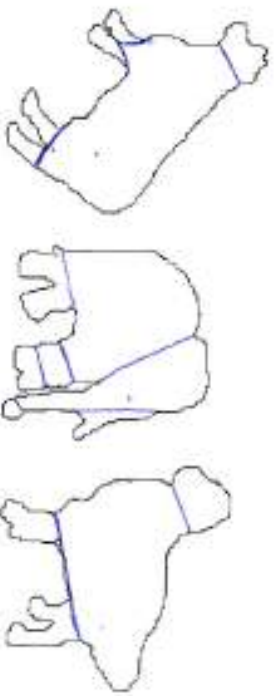
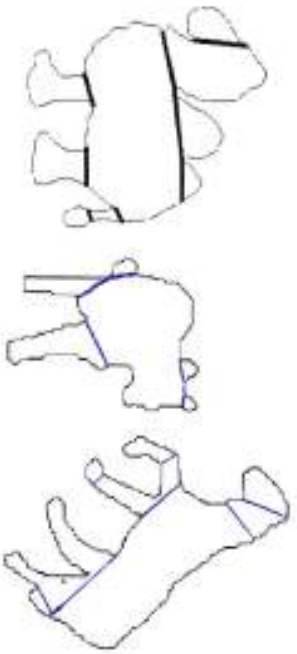
Name: Zeynep

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Thuto Temoiso Wesi

Age:

☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

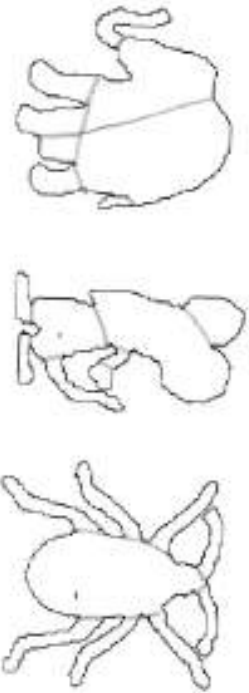
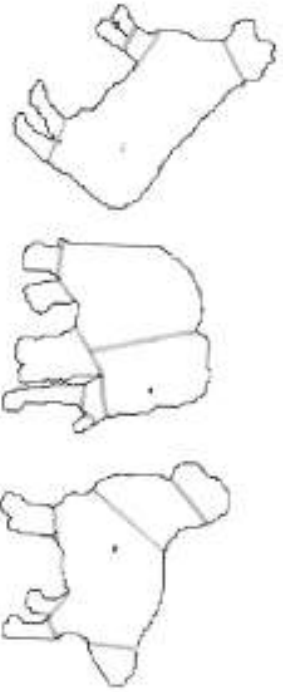
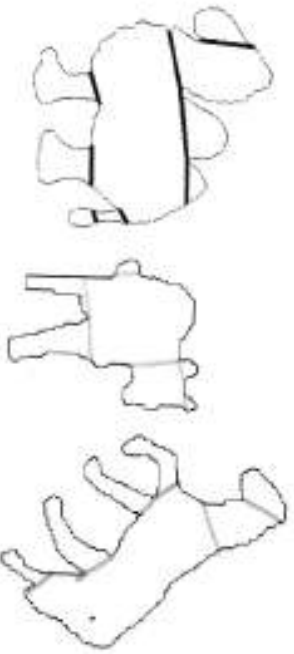
Gender:

☒ Male ☐ Female

Race:

☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: lehhago Malef

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

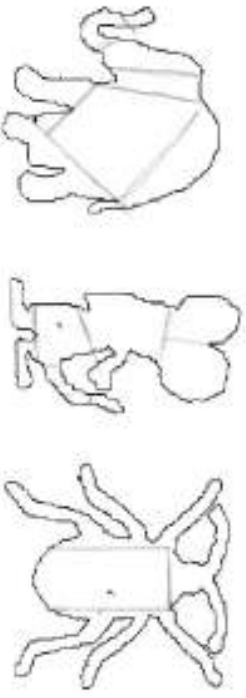
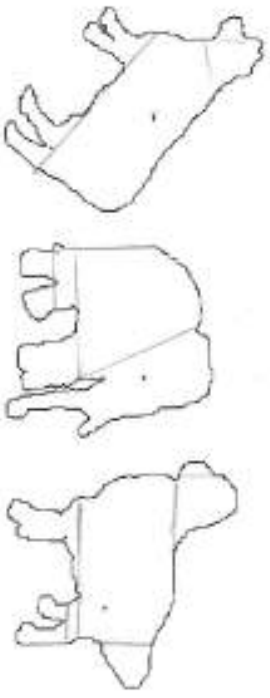
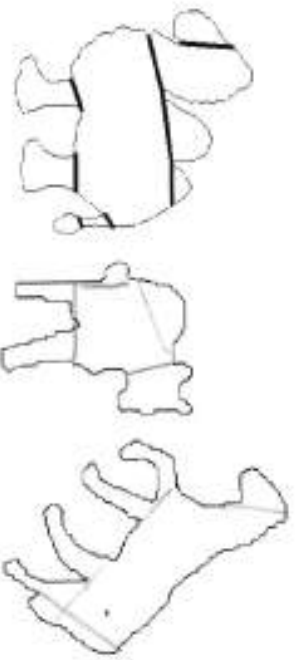
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Sahar K. Alfi

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

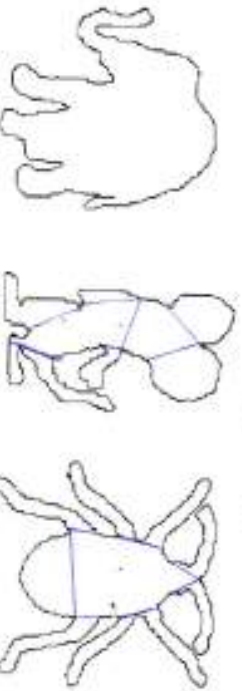
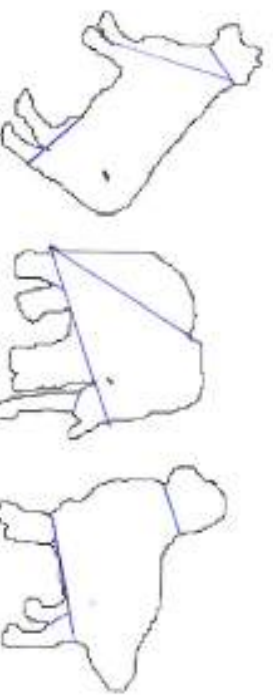
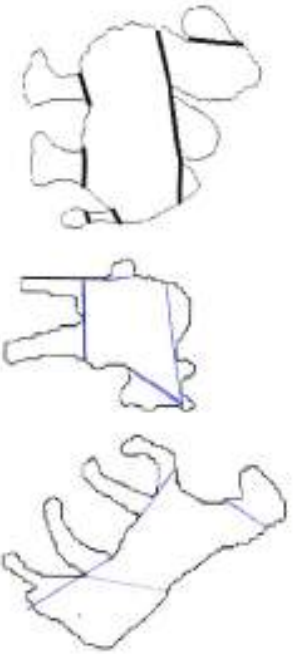
Gender:

☒ Male	☐ Female
---------------------------------------	------------------------------

Race:

☒ African	☐ Indian	☐ Caucasian	☐ Asian	☐ Other
---	---------------------------------	------------------------------------	--------------------------------	--------------------------------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

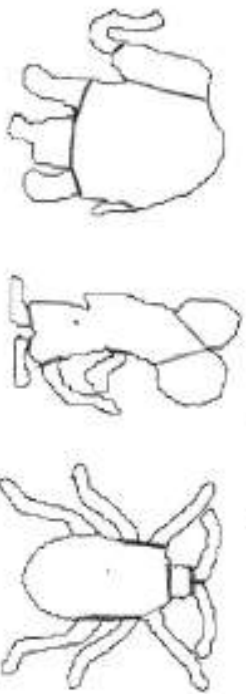
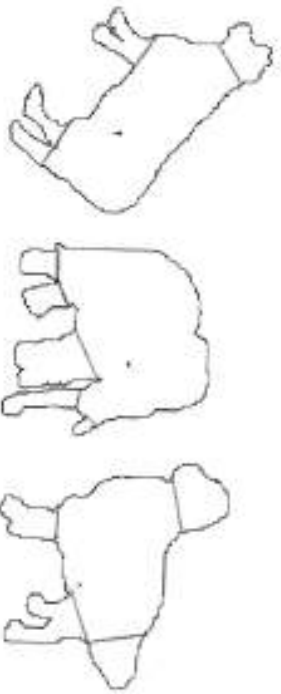
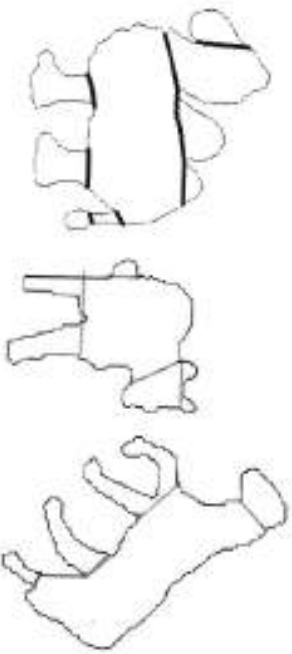
Name: Neerabeking Practices 1cgy4

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Siddhant Gennu

Age:

<20 ✓	20-30	30-40	40-50	50-60	>60
-------	-------	-------	-------	-------	-----

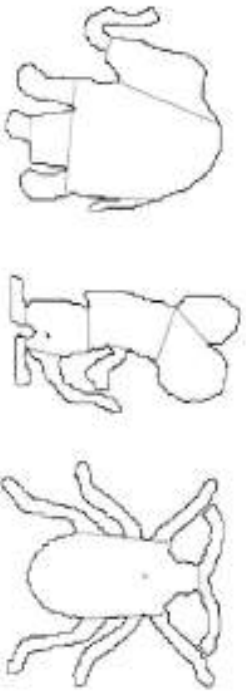
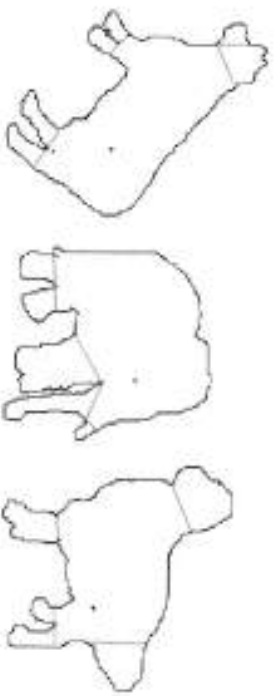
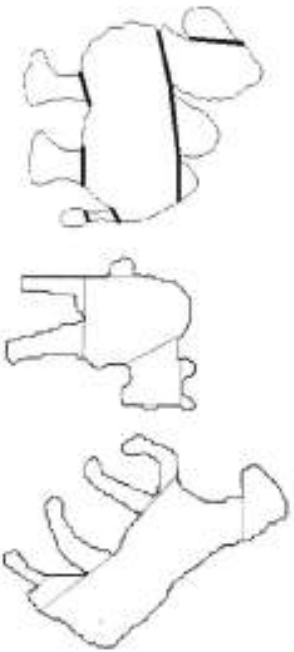
Gender:

Male	Female ✓
------	----------

Race:

African ✓	Indian	Caucasian	Asian	Other
-----------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Ilisa Benoit

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

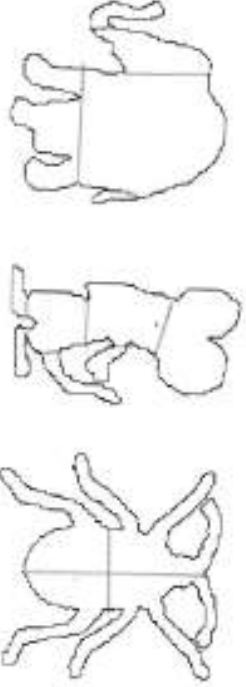
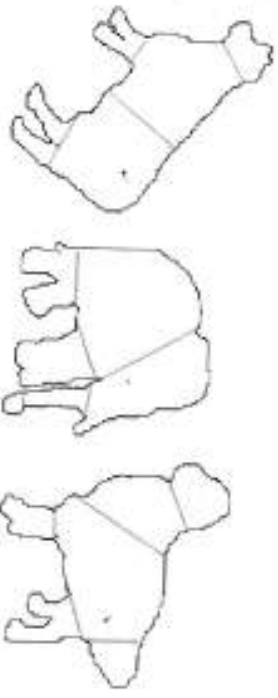
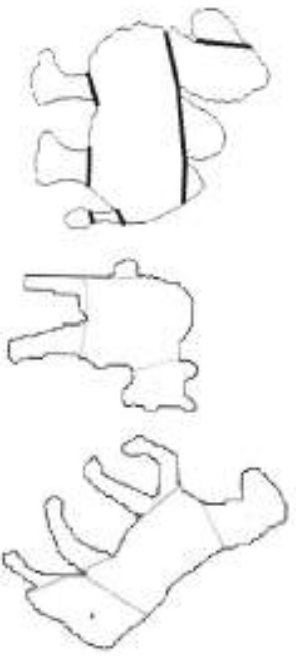
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Pongumelo Mkhuyane

Age:

☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

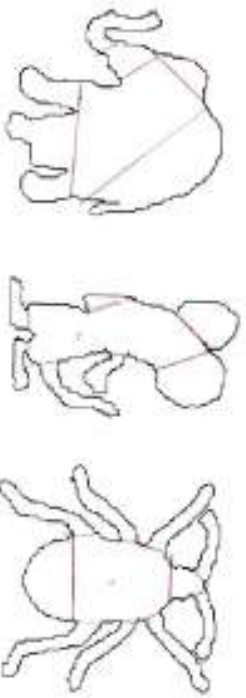
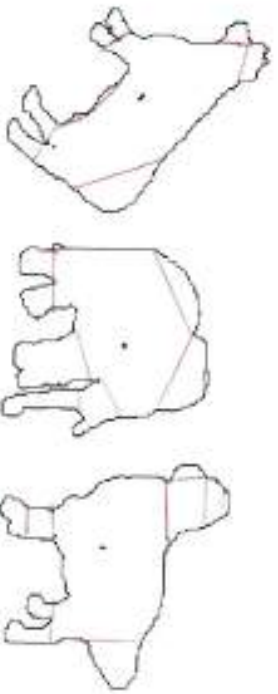
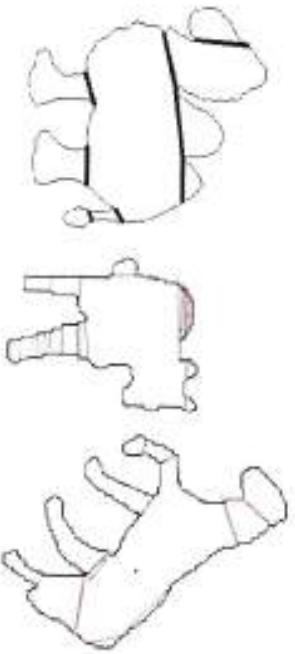
Gender:

☐ Male ☒ Female

Race:

☐ African ☒ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

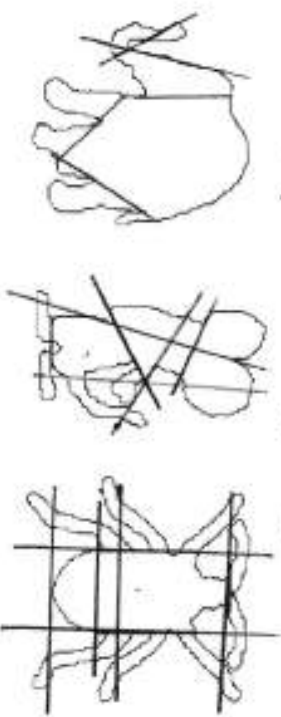
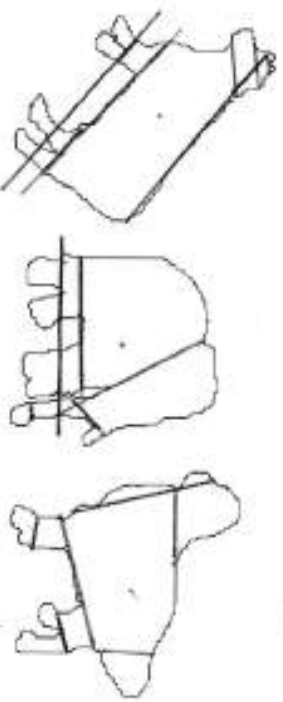
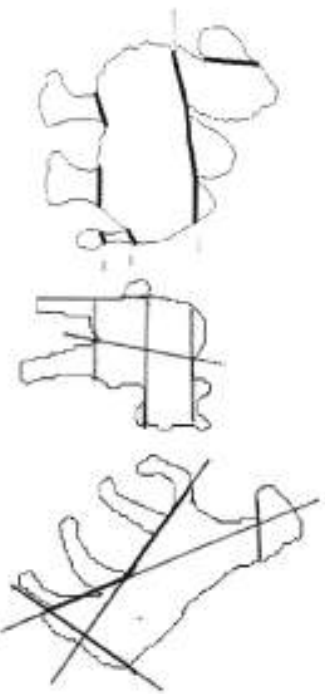
Name: Therique Nell

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☐ African ☐ Indian ☒ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

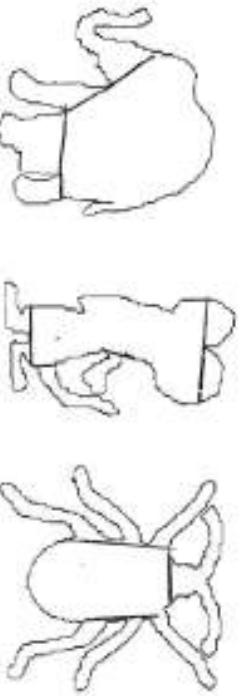
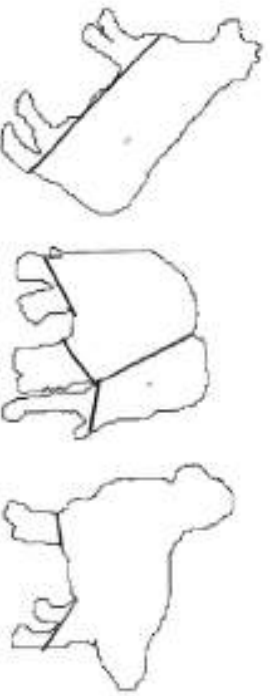
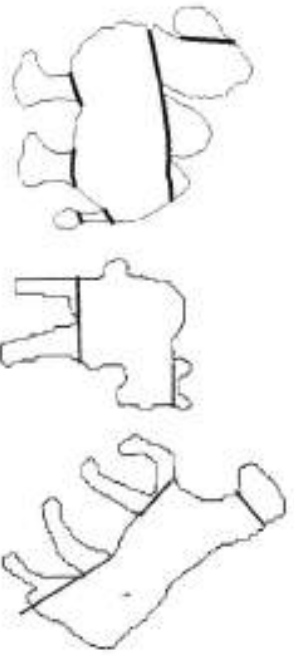
Name: Kanabata Magesi

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

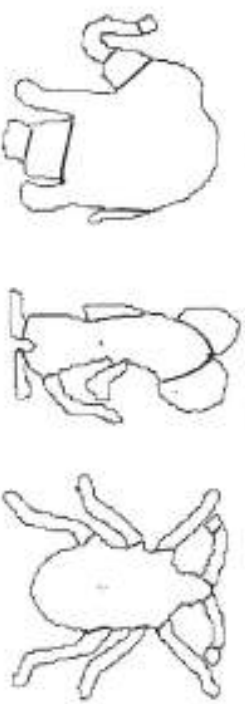
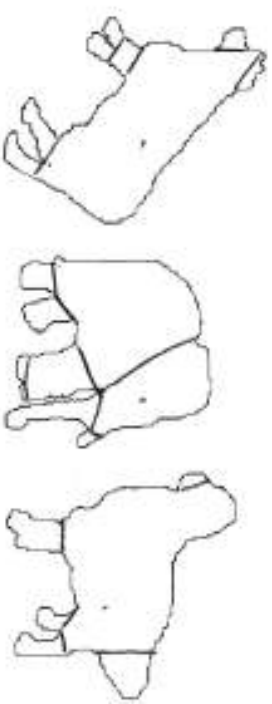
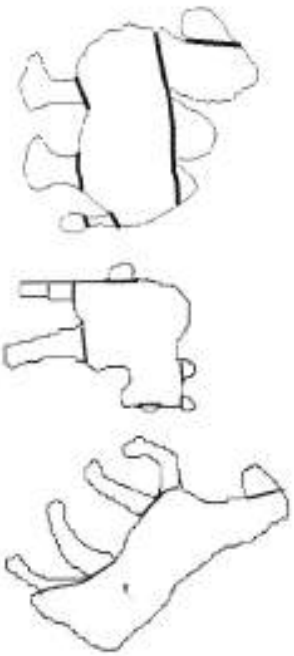
Name: Tebogo Morake

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

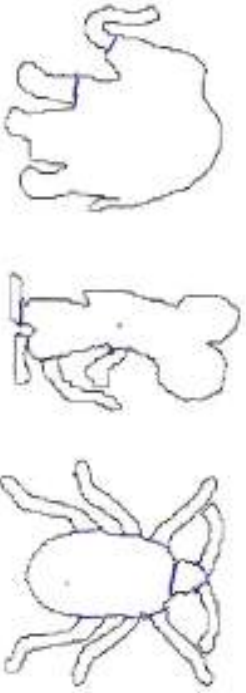
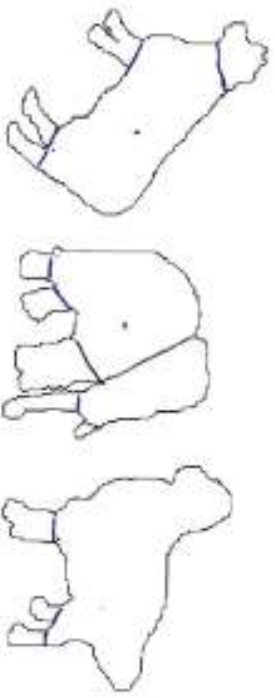
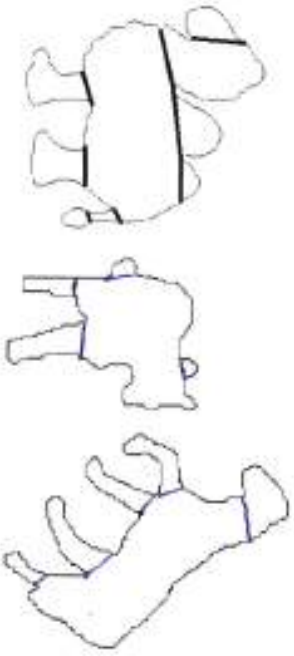
Name: Uw Jinhua

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☒ Male ☐ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: _____

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

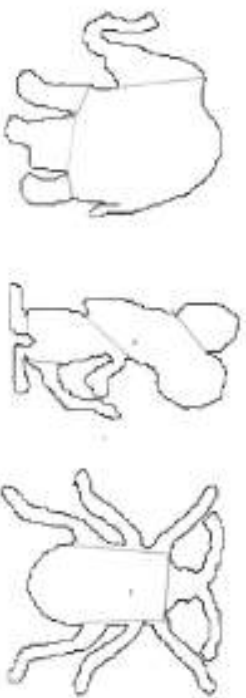
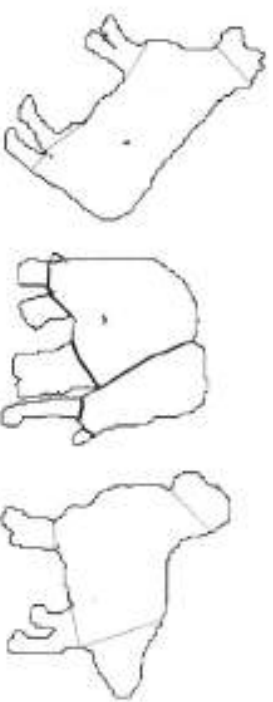
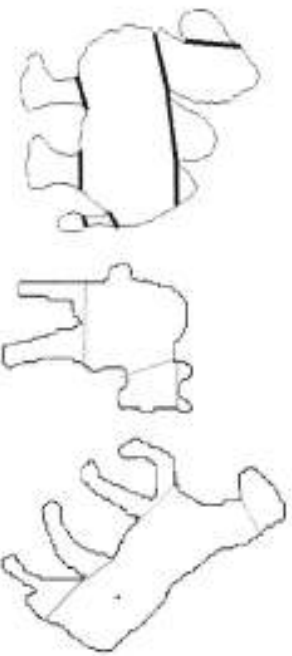
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

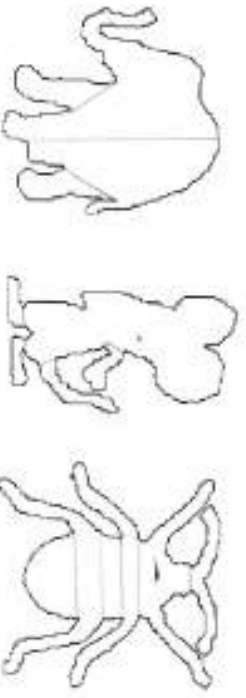
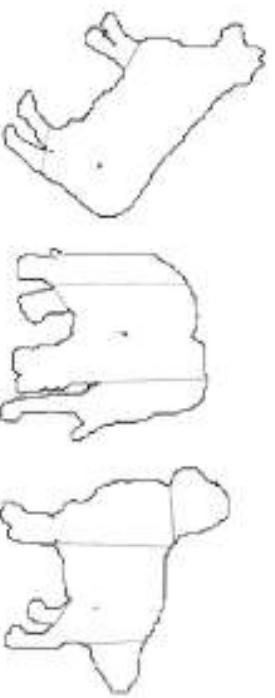
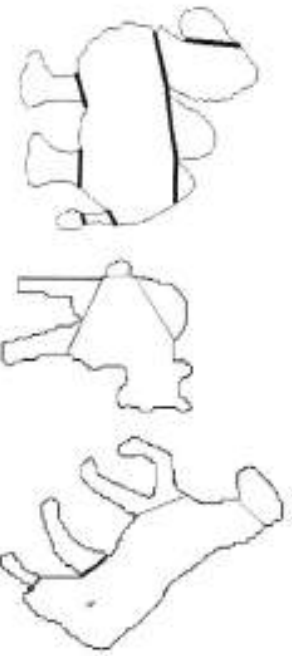
Name: Thomase Muthiga

Age: ☐ <20 ☒ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☐ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Eugene

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

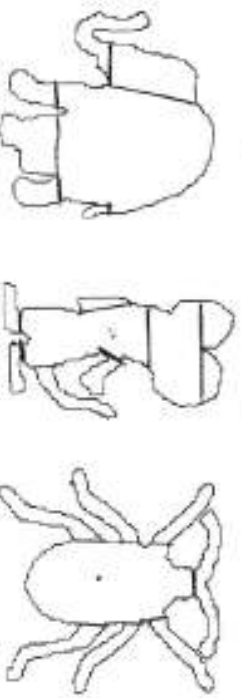
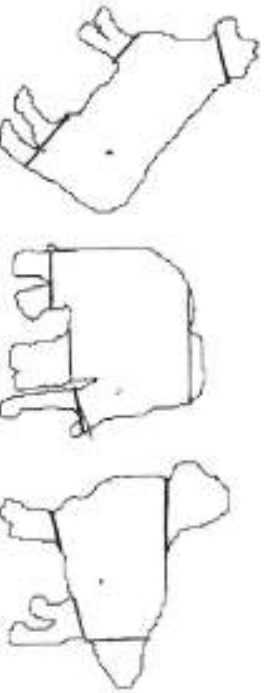
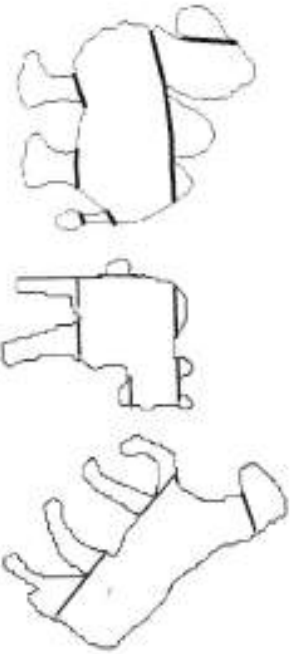
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: _____

Age: _____

Gender: _____

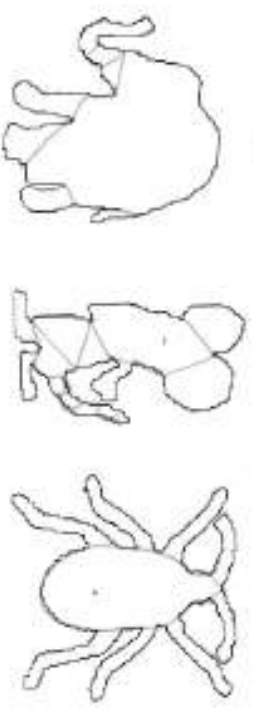
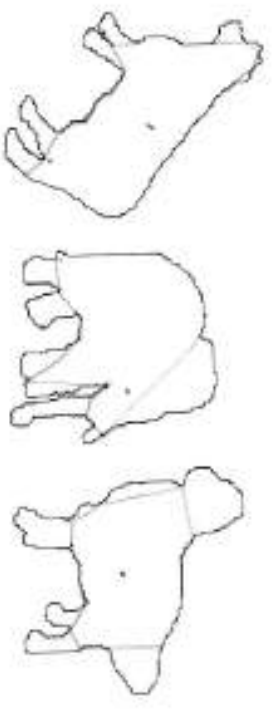
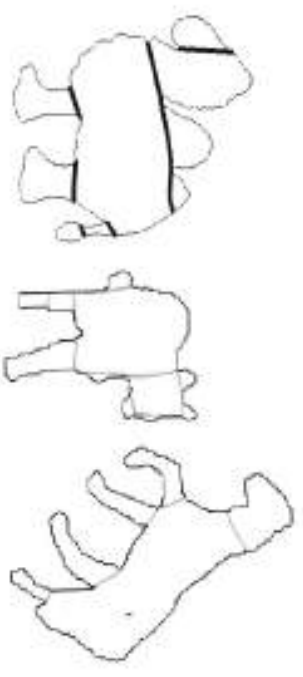
Race: _____

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

Male	Female
------	--------

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Kelipone Maleyone

Age:

<20 20-30 30-40 40-50 50-60 >60

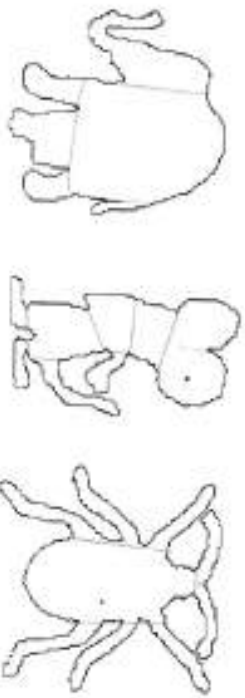
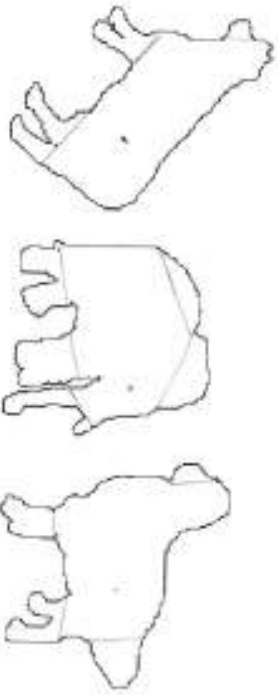
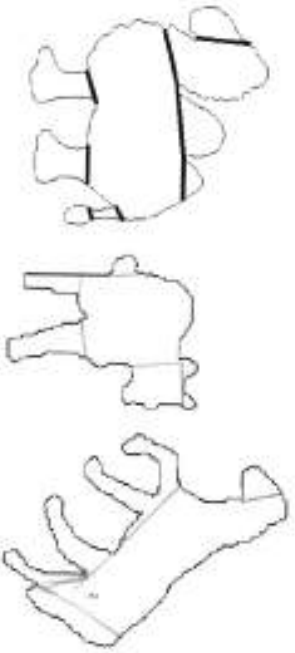
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Nonlani a Ngwenya

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

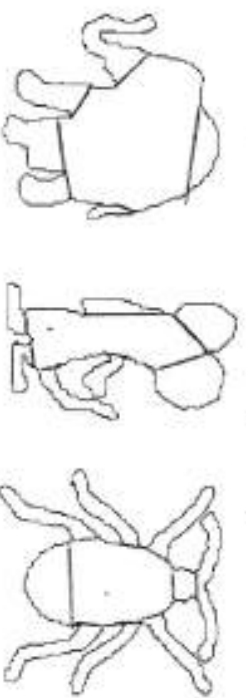
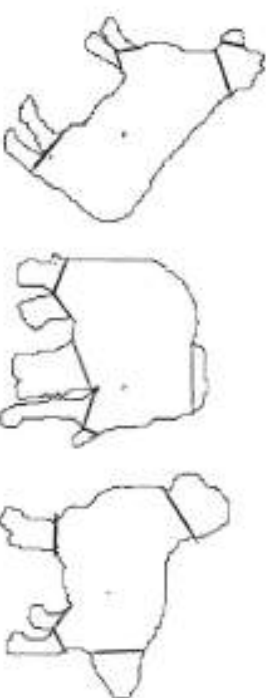
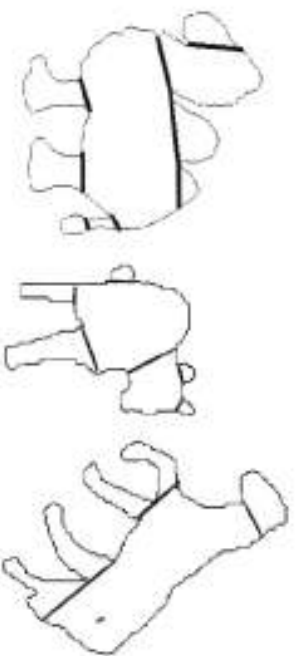
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Desegga Makedi

Age:

☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

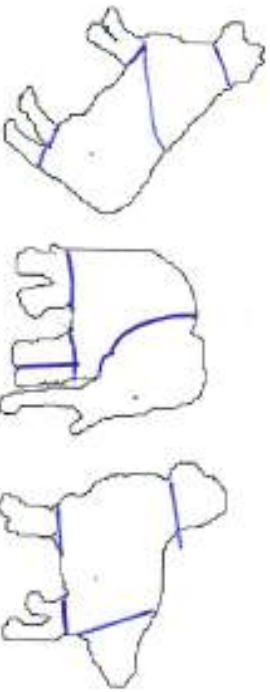
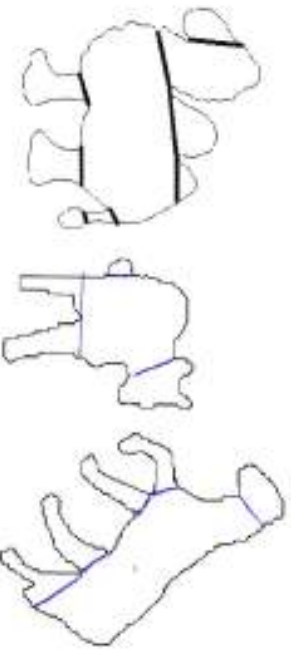
Gender:

☐ Male ☒ Female

Race:

☐ African ☐ Indian ☐ Caucasian ☐ Asian ☒ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Baturino Jetshetu

Age:

<20 20-30 30-40 40-50 50-60 >60

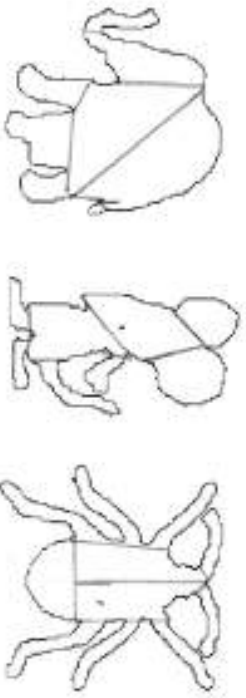
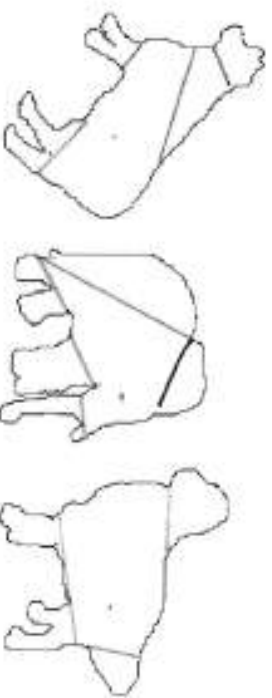
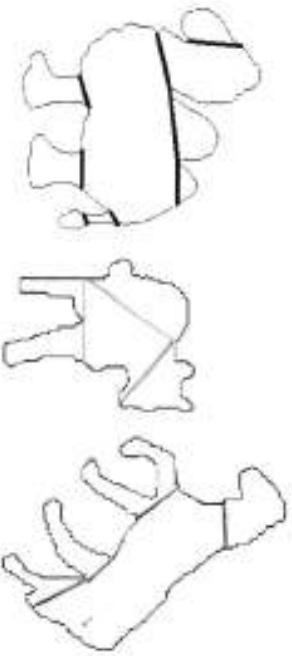
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Enzo Kuitje Barends

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

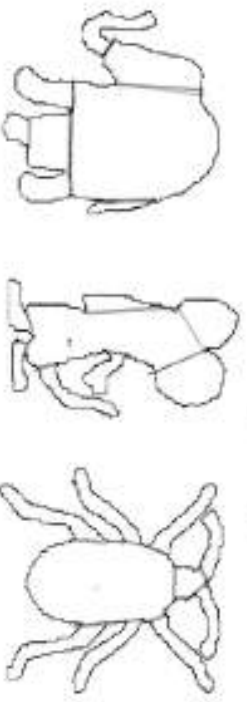
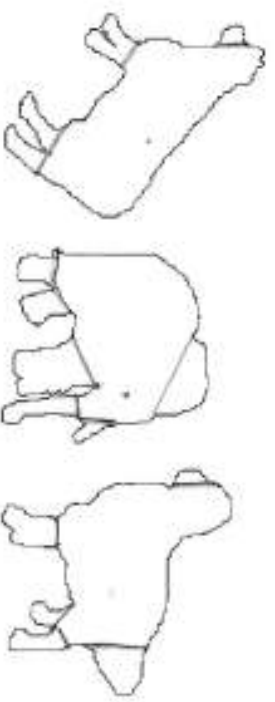
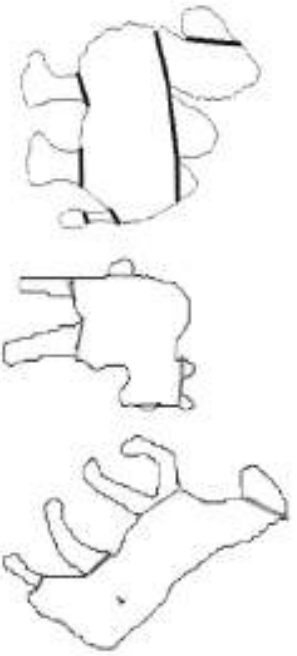
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: _____

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

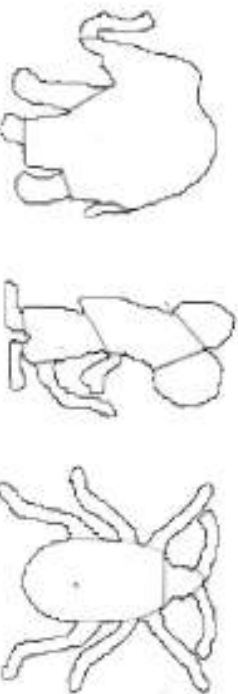
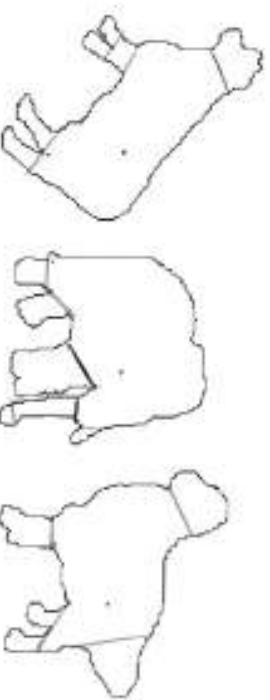
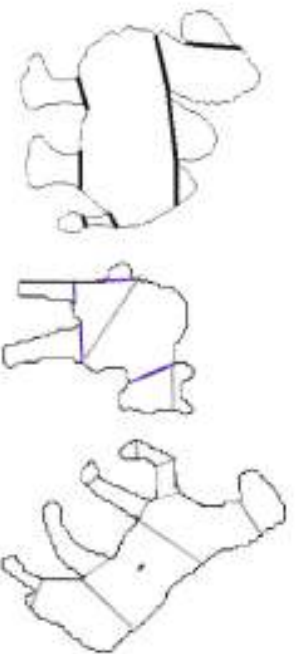
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Karabo Mhelez

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

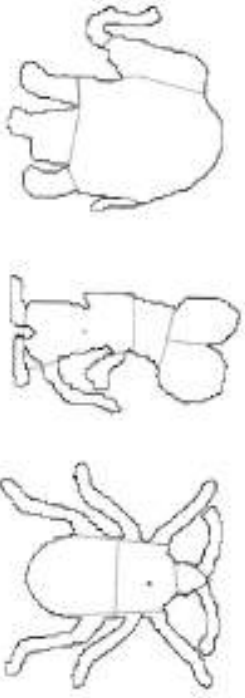
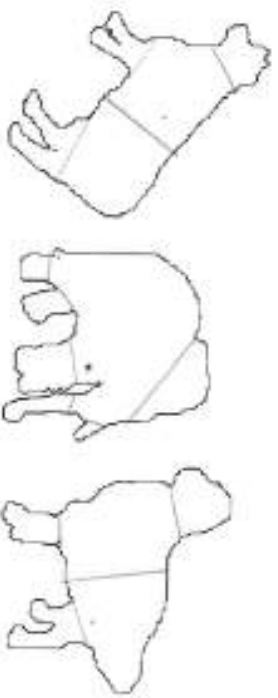
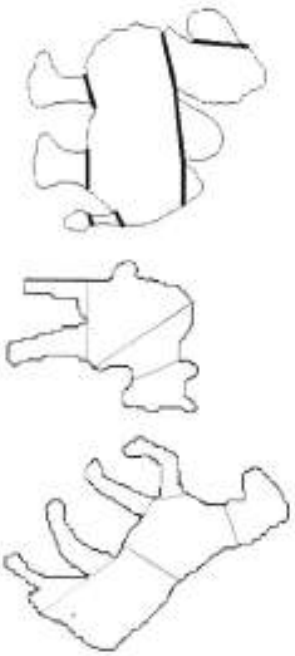
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Natasha DiSanto

Age:

<20 20-30 30-40 40-50 50-60 >60

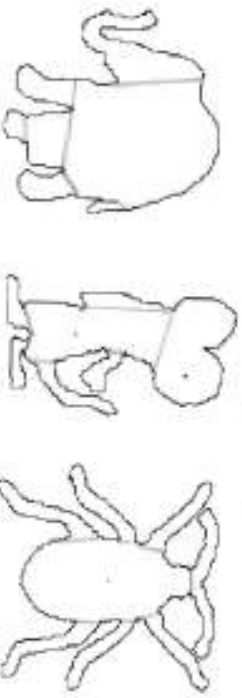
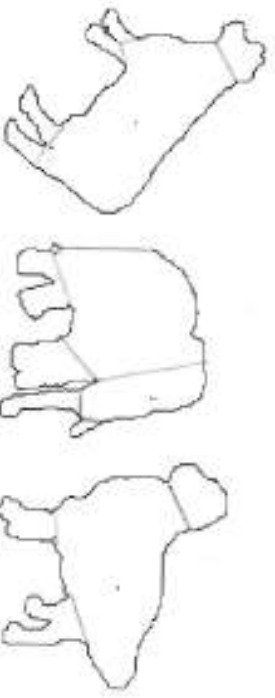
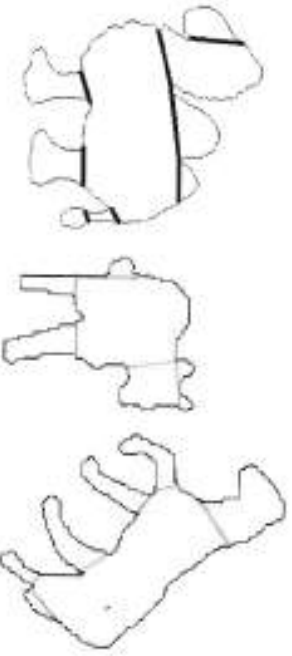
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Heena Mungroo

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

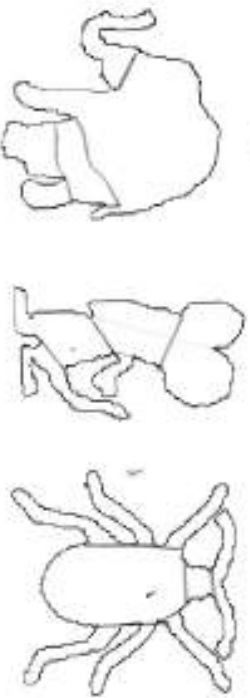
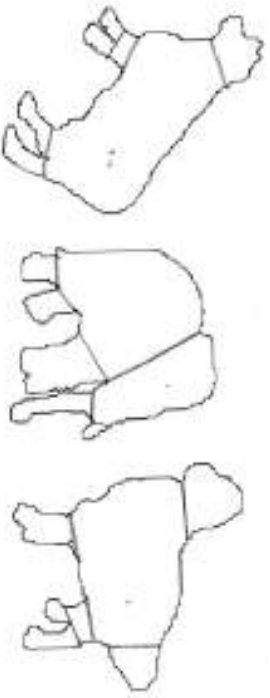
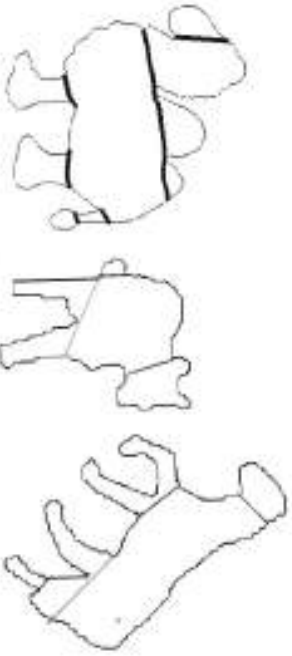
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Nontuthuzelo Duayi

Age: 13

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

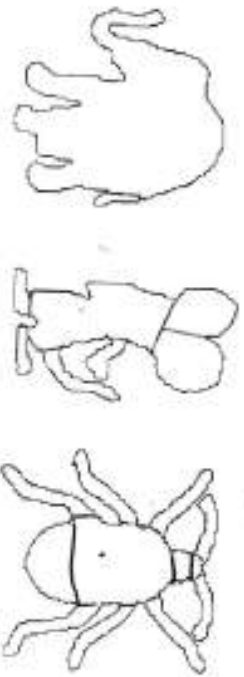
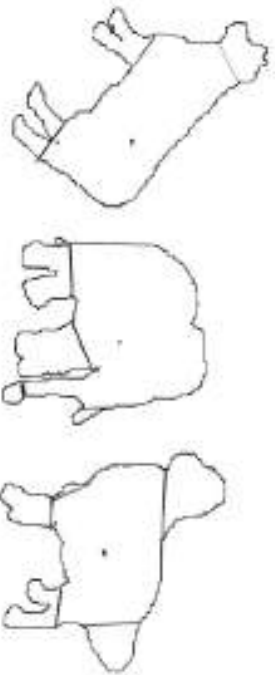
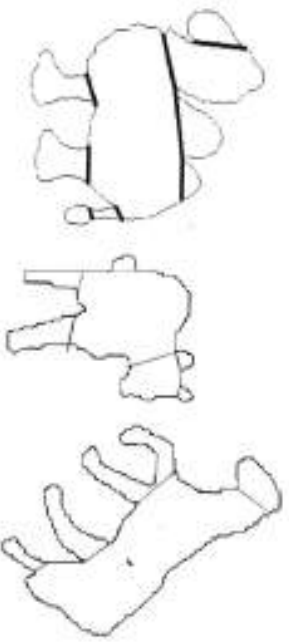
Gender:

Male	<u>Female</u>
------	---------------

Race:

<u>African</u>	Indian	Caucasian	Asian	Other
----------------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: _____

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

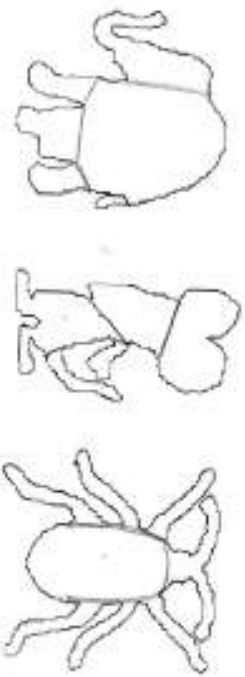
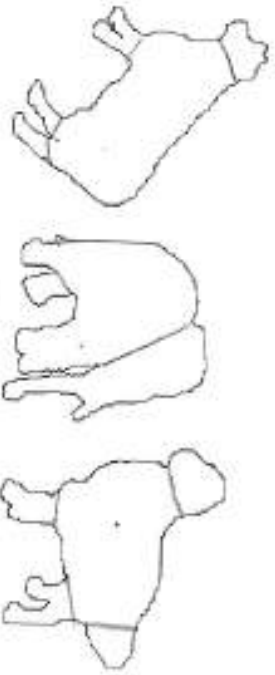
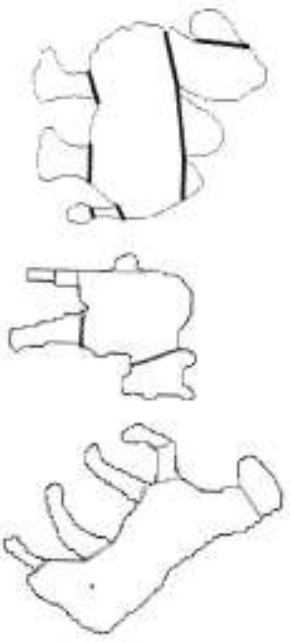
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

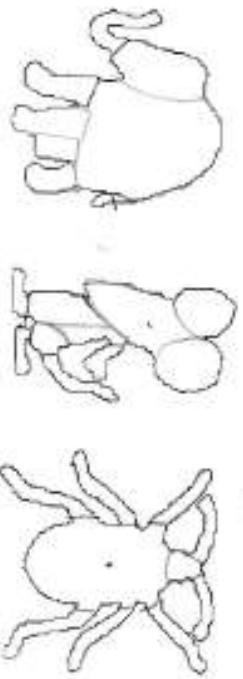
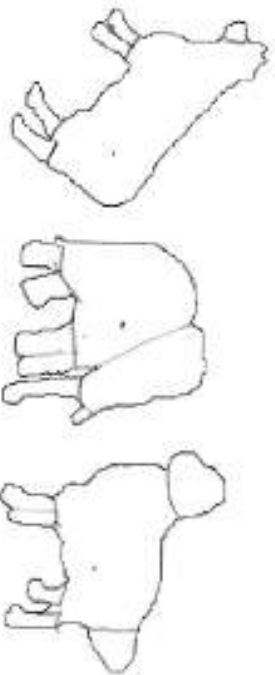
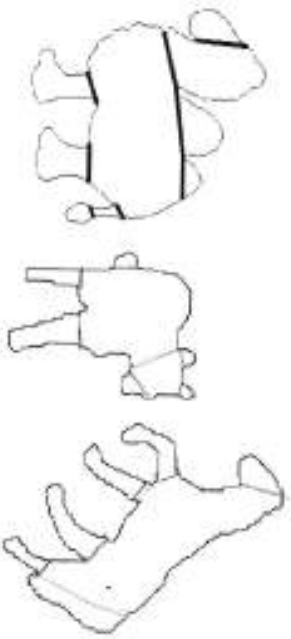
Name: Jessica Khumalo

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: ☒ Male ☐ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

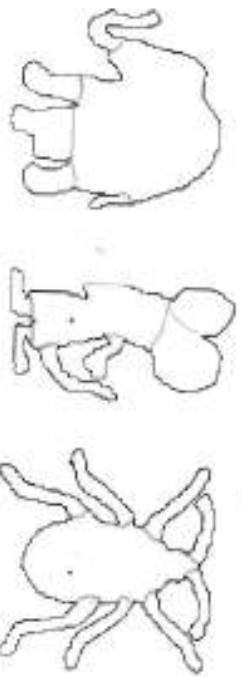
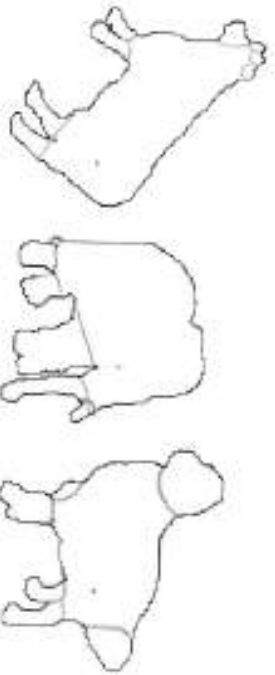
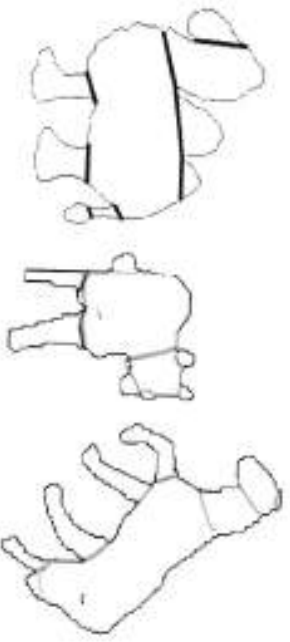
Name: Ketevagile Mwakide

Age: 13 12 <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female ☒

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

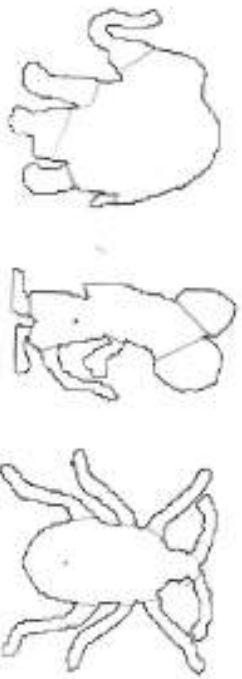
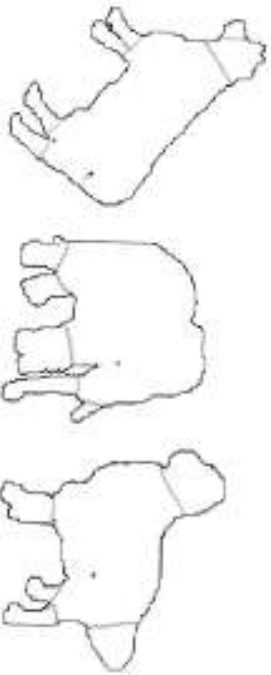
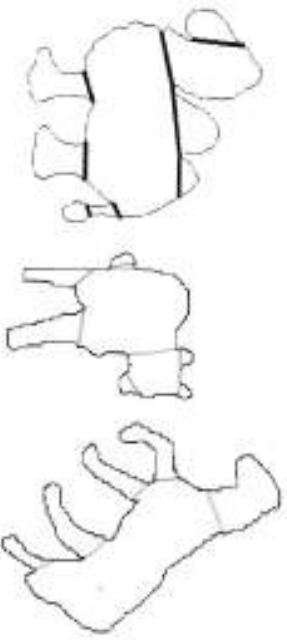
Name: Bonoko Charity Loko

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Arundhata Madosi 28 15 years

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

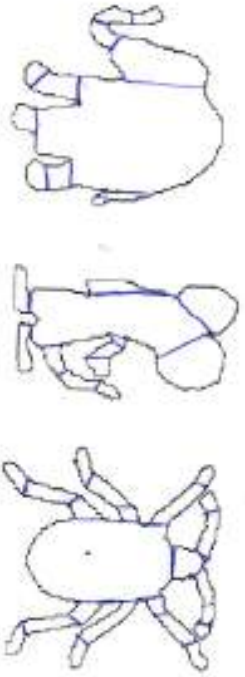
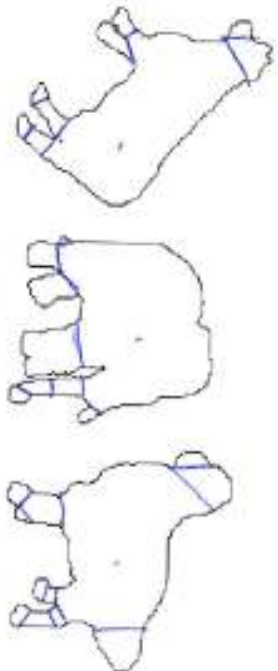
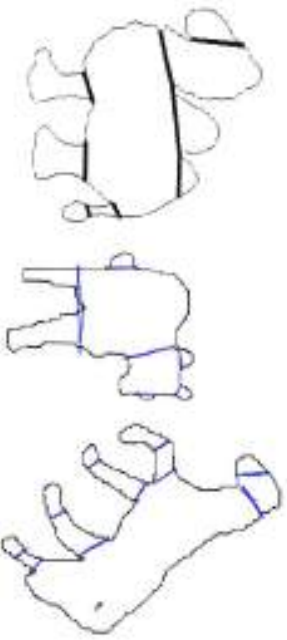
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Quynh Nguyen TB

Age: 15

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

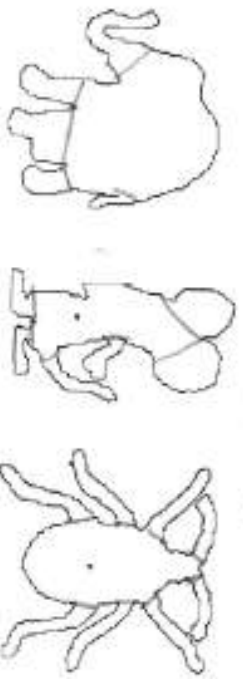
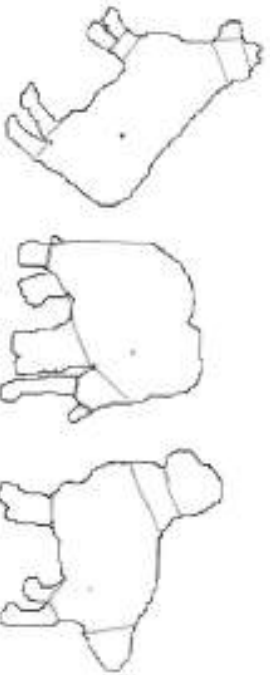
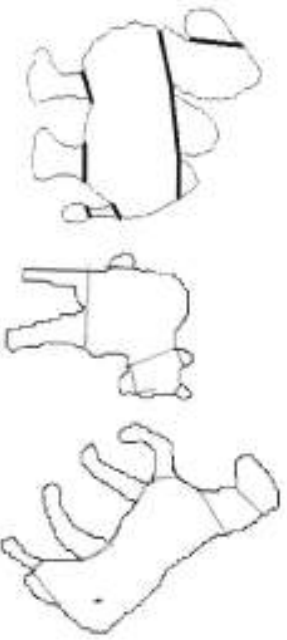
Gender:

Male	☒ Female
------	--

Race:

☒ African	Indian	Caucasian	Asian	Other
---	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Khotso Kobageng

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

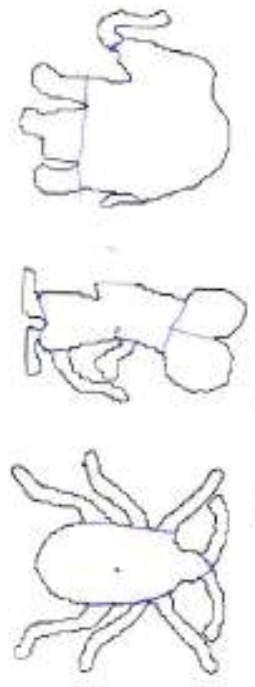
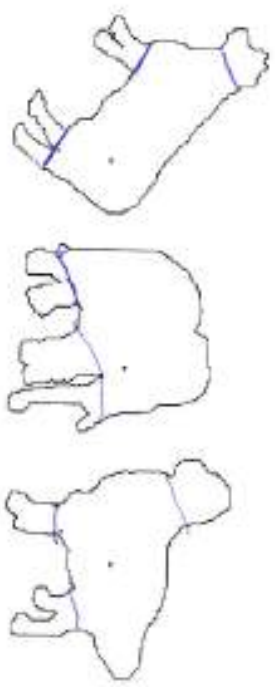
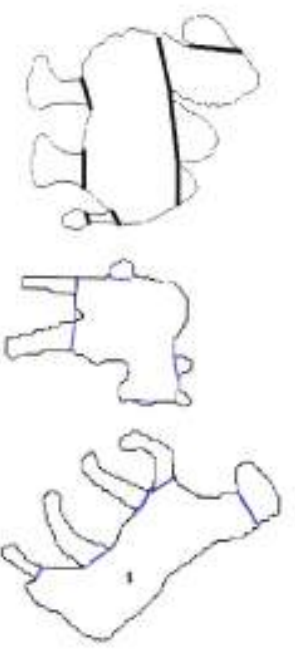
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire CT115

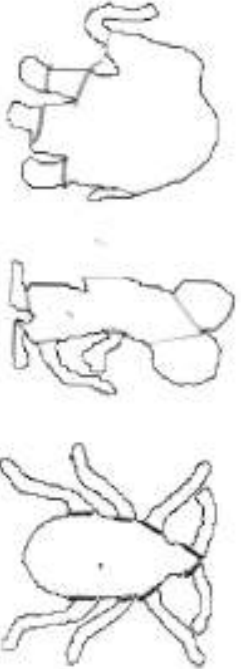
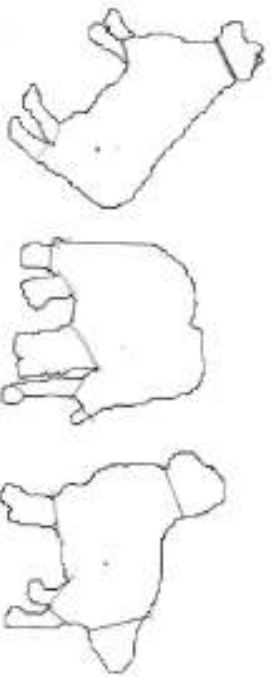
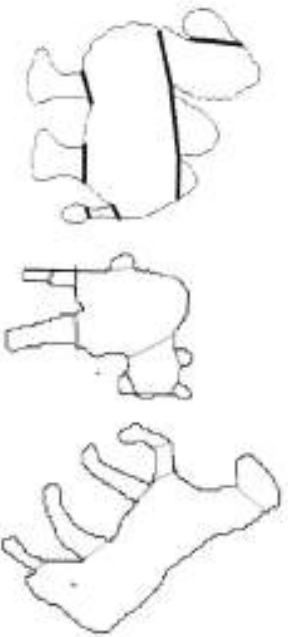
Name: Reganeditse P. Maledzane

Age: 15 ☐ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

Abigail mbiga

Age:

<20 20-30 30-40 40-50 50-60 >60

12

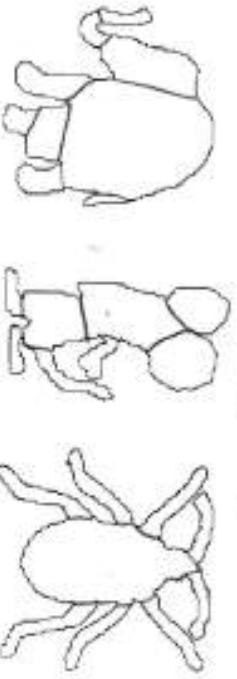
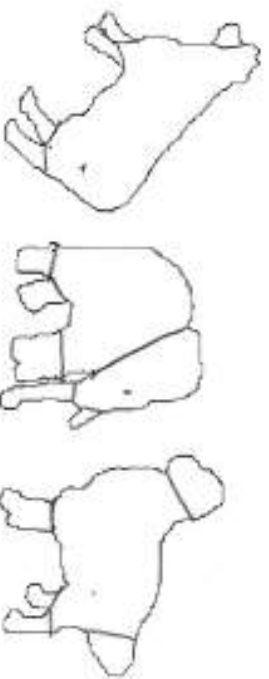
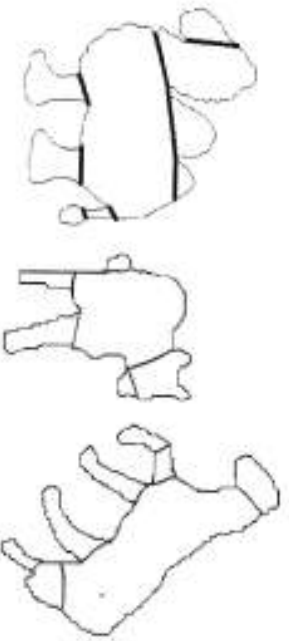
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Sonjiso Mdyane

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

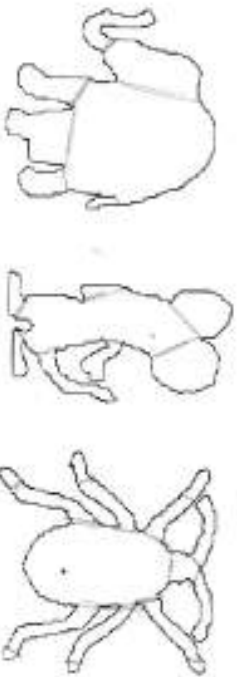
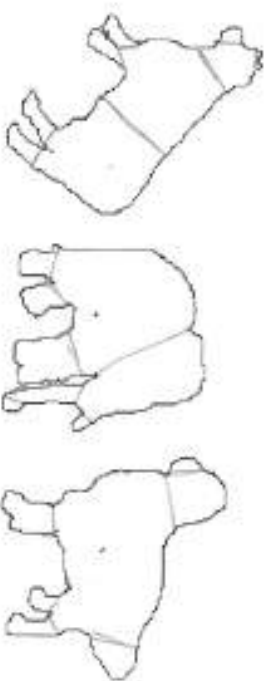
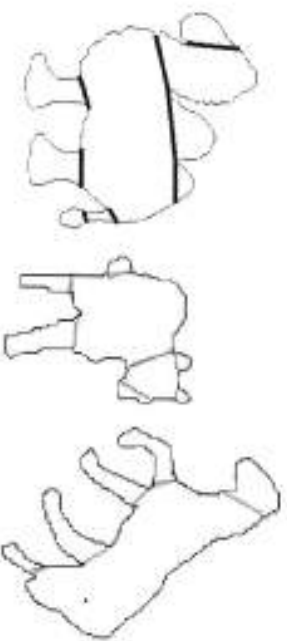
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

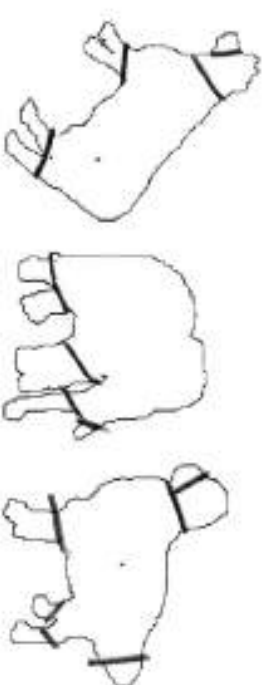
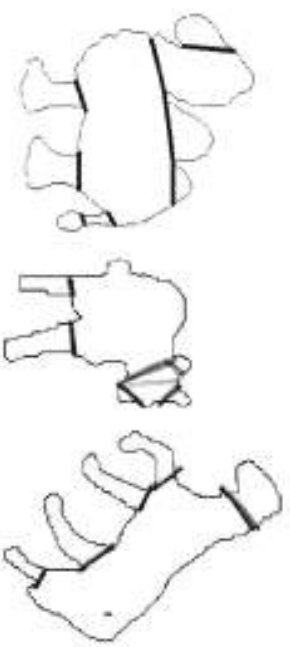
Name: Nico mynd

Age: 13 ☐ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Khuthafzu Mckwebu 8b

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

13

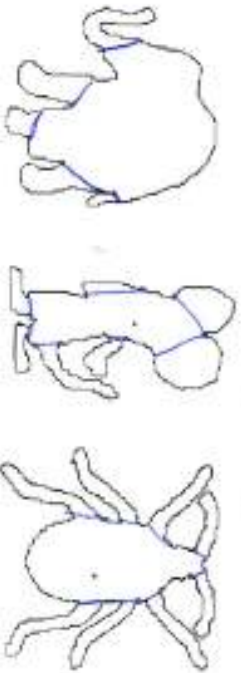
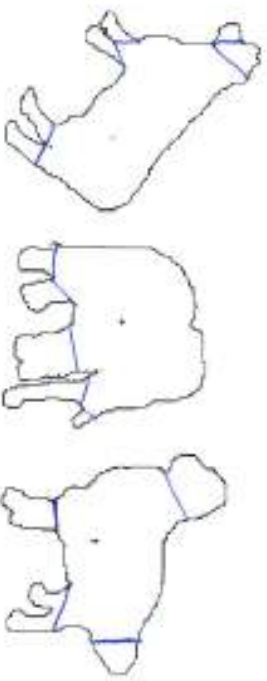
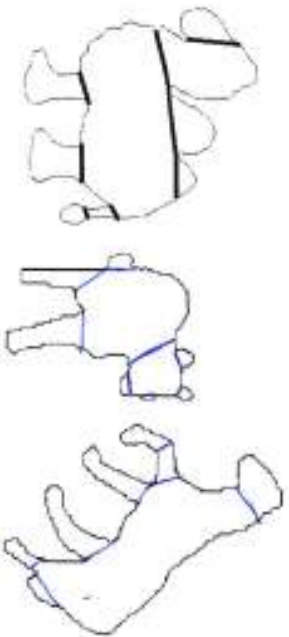
Gender:

Male	<u>Female</u>
------	---------------

Race:

<u>African</u>	Indian	Caucasian	Asian	Other
----------------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

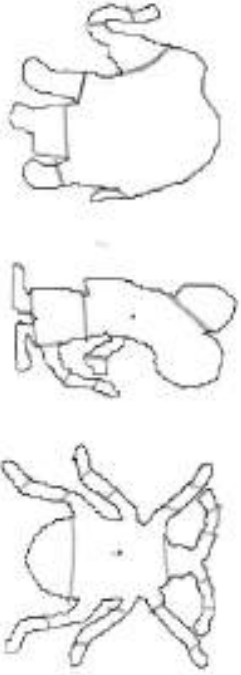
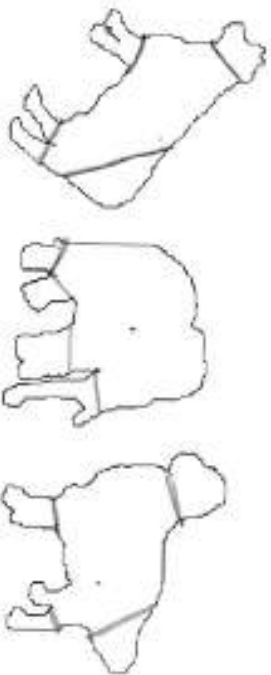
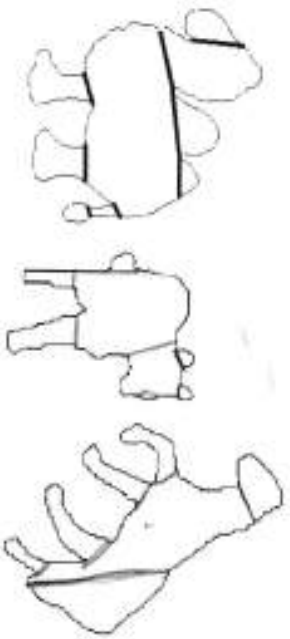
Name: Regina Nozic

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: People Niteone

Age:

<20 ✓	20-30	30-40	40-50	50-60	>60
-------	-------	-------	-------	-------	-----

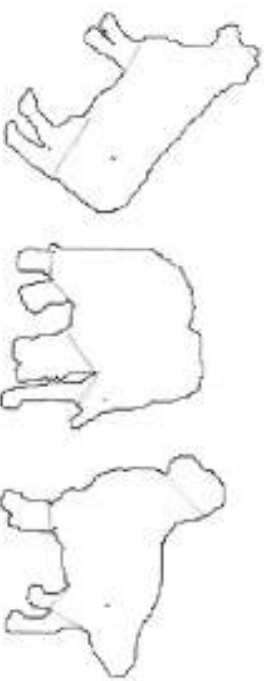
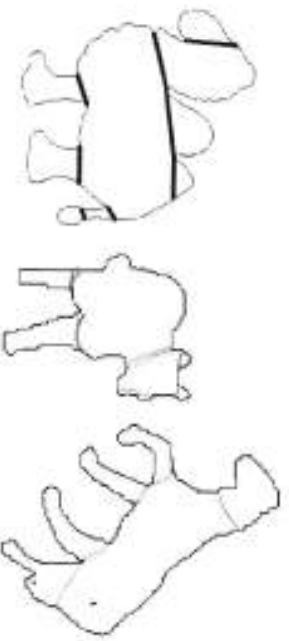
Gender:

Male ✓	Female
--------	--------

Race:

African ✓	Indian	Caucasian	Asian	Other
-----------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

8B

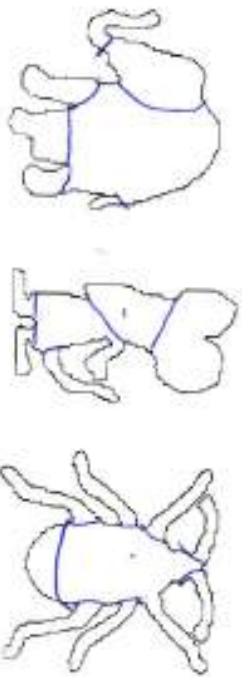
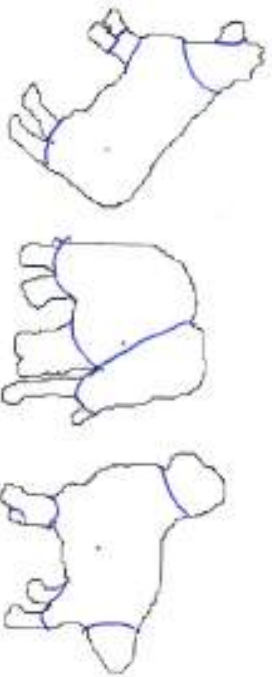
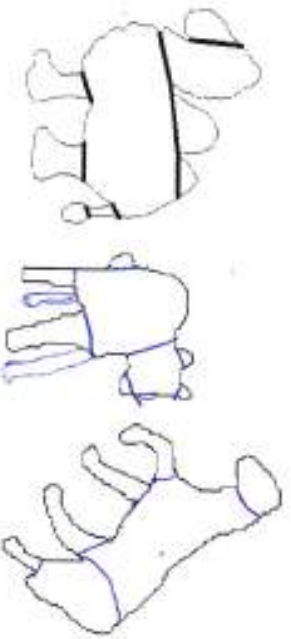
Name: Korabele Ntseki

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: MAREN SCHOLE

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

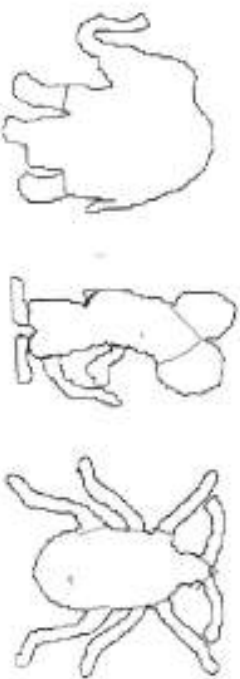
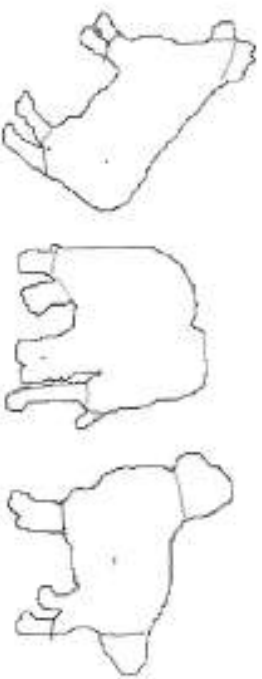
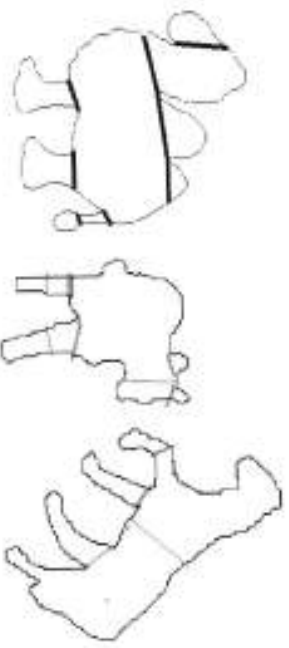
Gender:

Male	☒ Female
------	--

Race:

African	Indian	☒ Caucasian	Asian	Other
---------	--------	---	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Danika Gunn

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

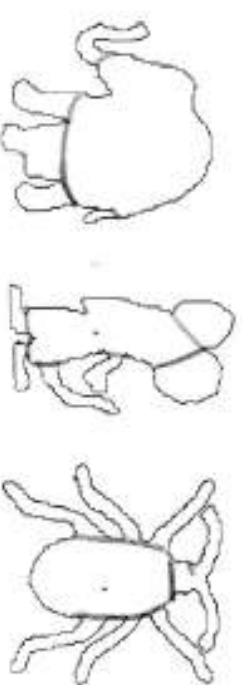
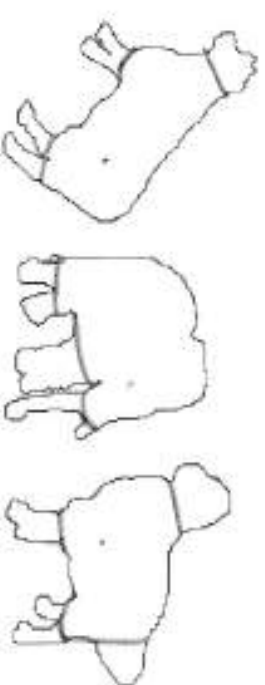
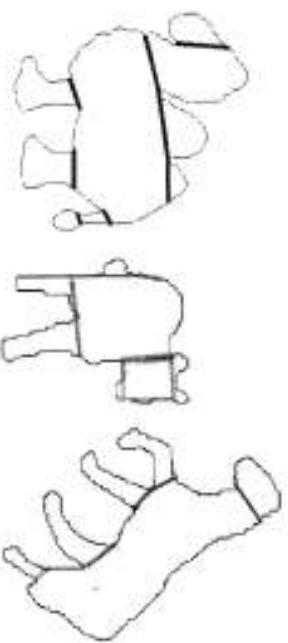
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Alex Nunez

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

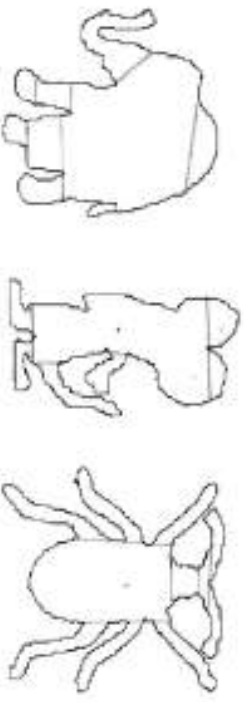
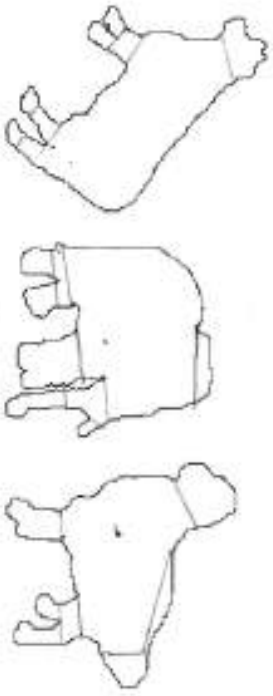
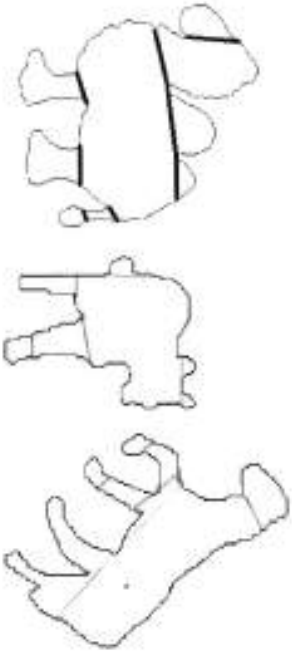
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Ndabha Ntantumbo

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

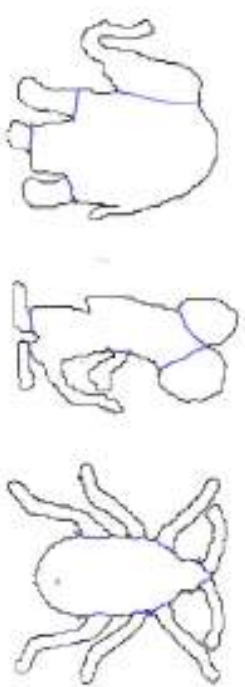
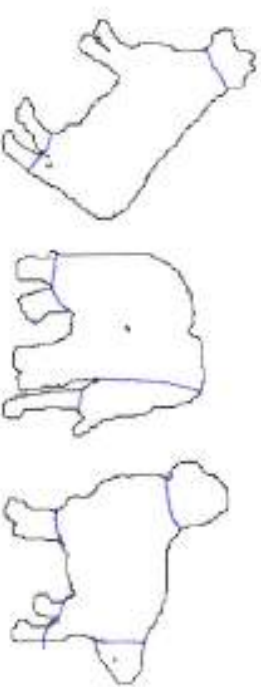
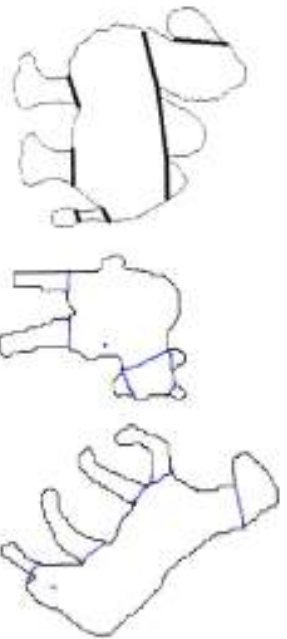
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Whobiscay 'Nau

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

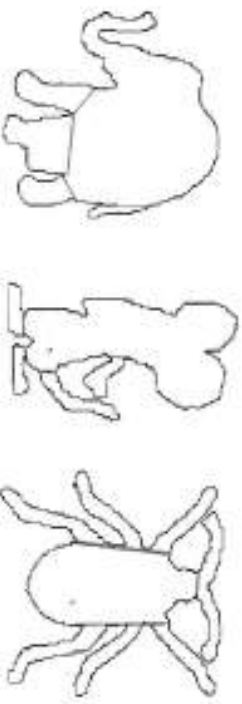
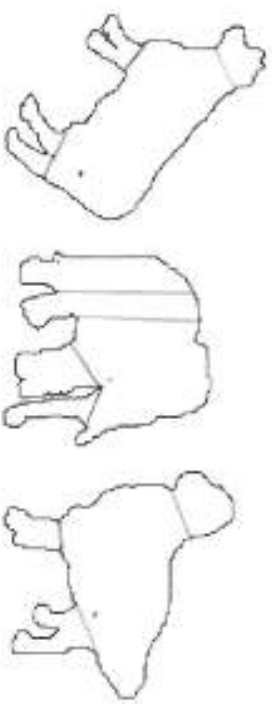
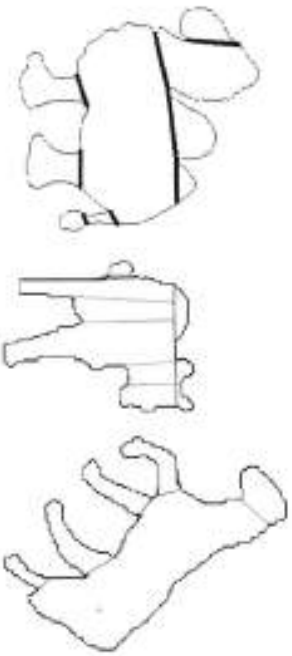
Gender:

Male	Female ☒
------	--

Race:

African ☒	Indian	Caucasian	Asian	Other
---	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

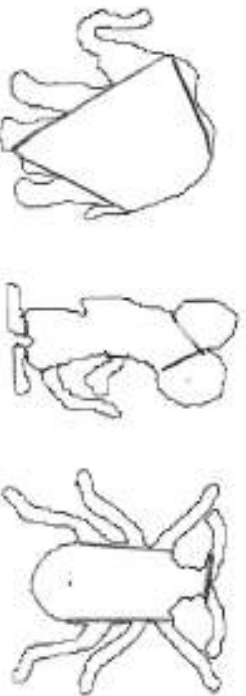
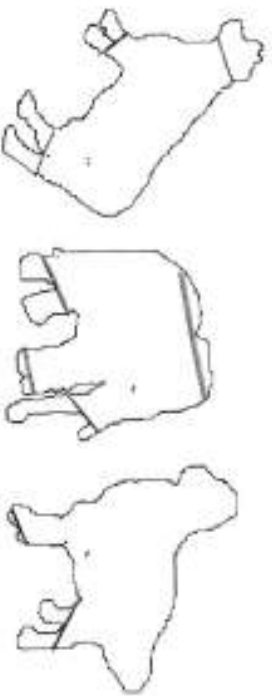
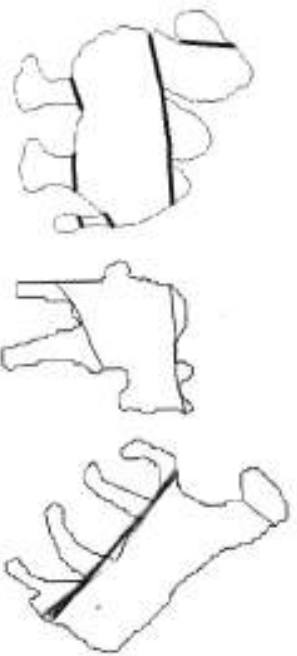
Name: Steven L

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☒ Male ☐ Female

Race: ☐ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name:

Brown-Lugh Sinclair

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

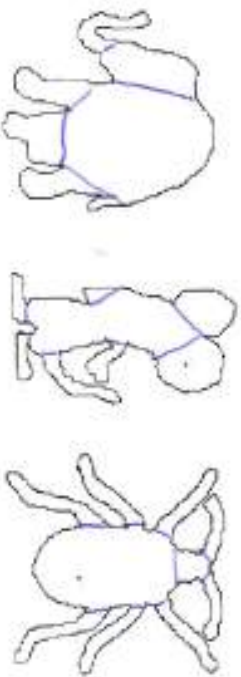
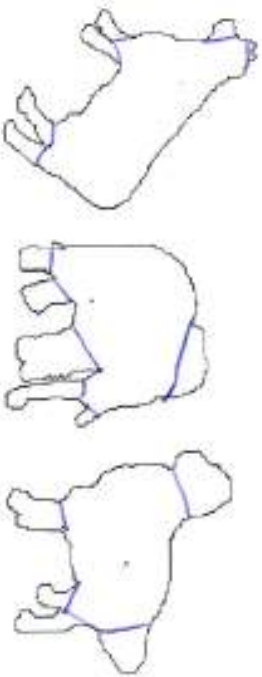
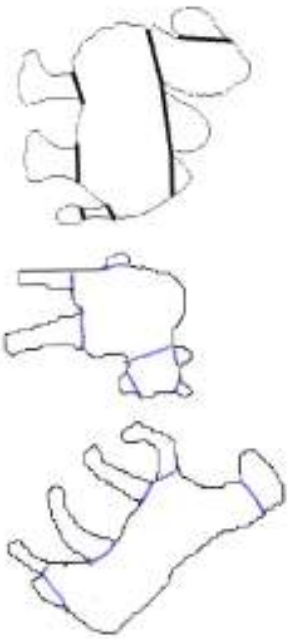
Gender:

Male	<u>Female</u>
------	---------------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Johnna Mhertse

Age: 16

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

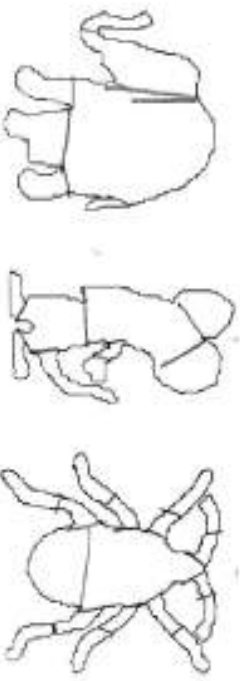
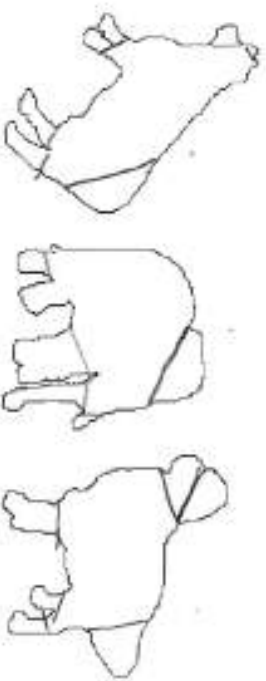
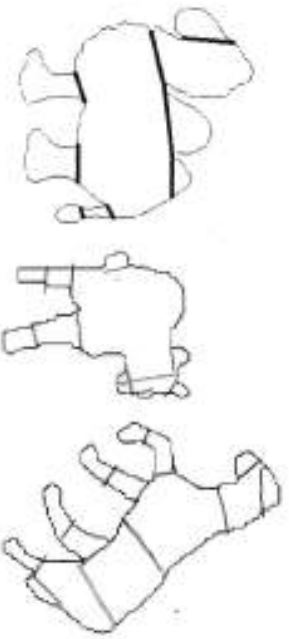
Gender:

Male	Female ☒
------	--

Race:

African ☒	Indian	Caucasian	Asian	Other
---	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

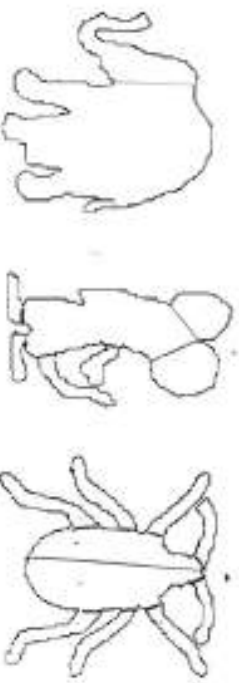
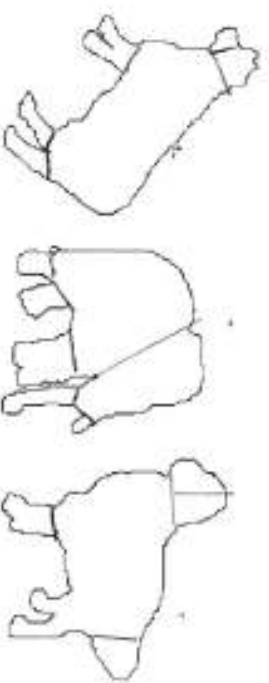
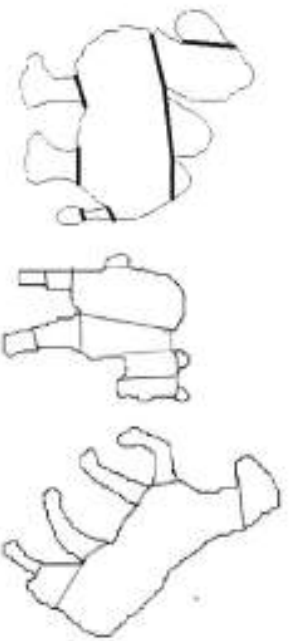
Name: Jennifer Maramba

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☒ Male ☐ Female

Race: ☐ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Peggy Mason

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

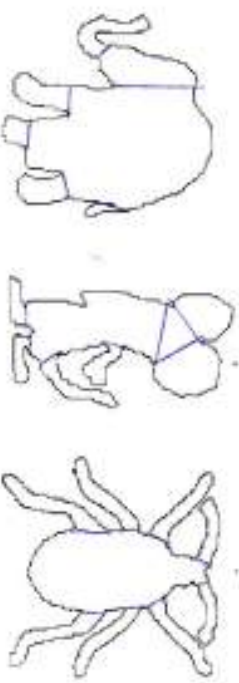
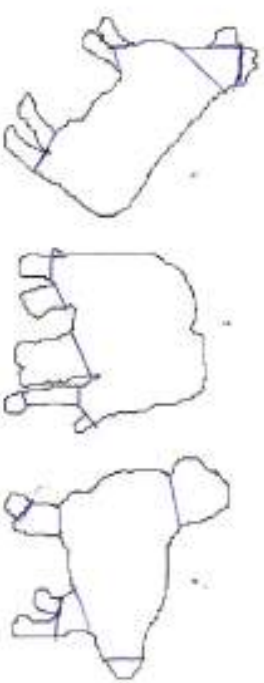
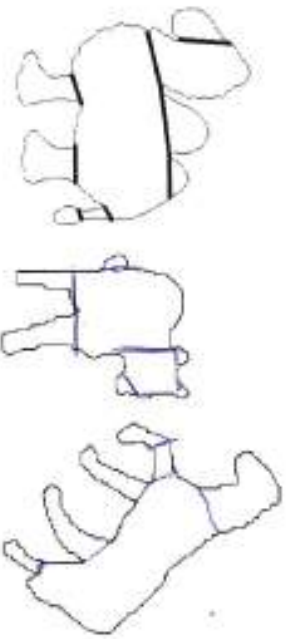
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: _____

Wendy

Age:

<20 ☒	20-30	30-40	40-50	50-60	>60
---	-------	-------	-------	-------	-----

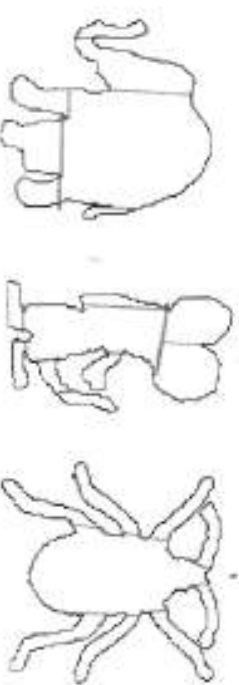
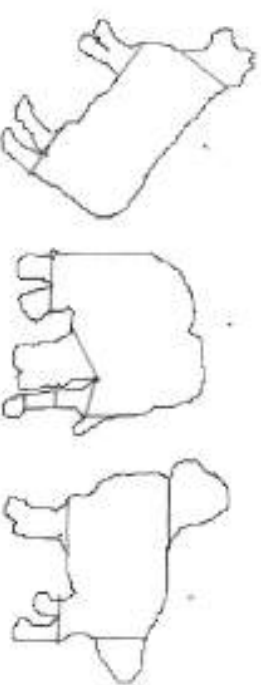
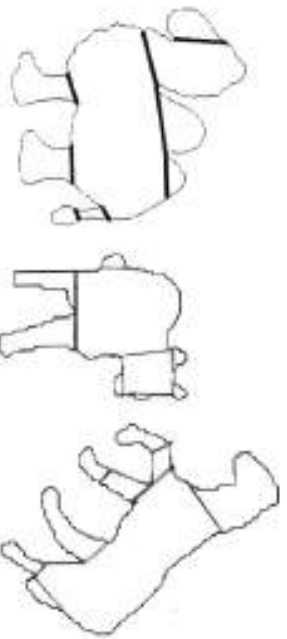
Gender:

Male ☒	Female
--	--------

Race:

African ☒	Indian	Caucasian	Asian	Other
---	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

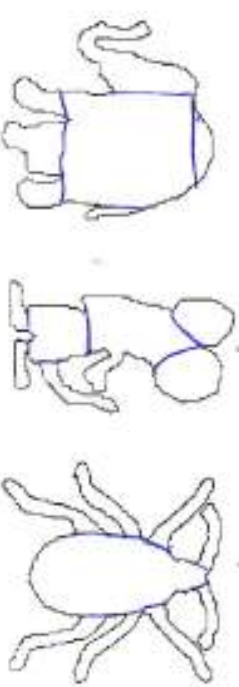
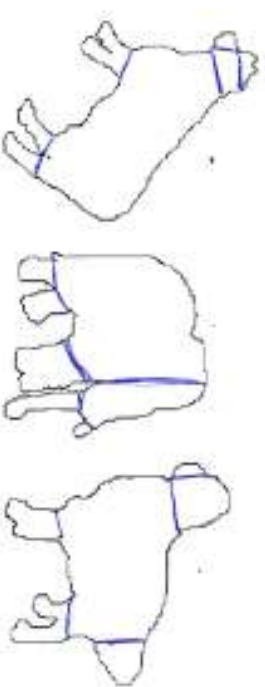
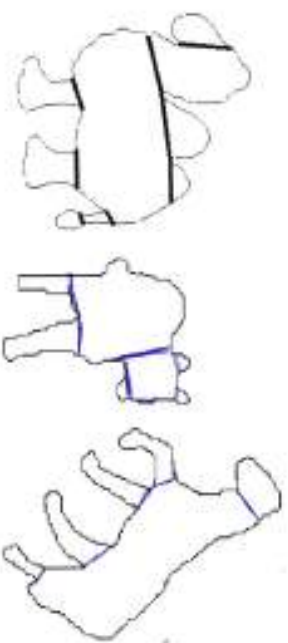
Name: Kenneth Bruce Price

Age: ☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

Gender: ☒ Male ☐ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Amphie Mithwane

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

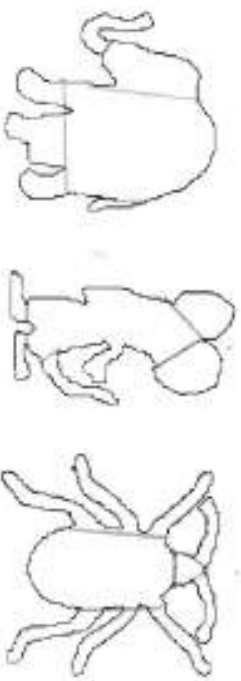
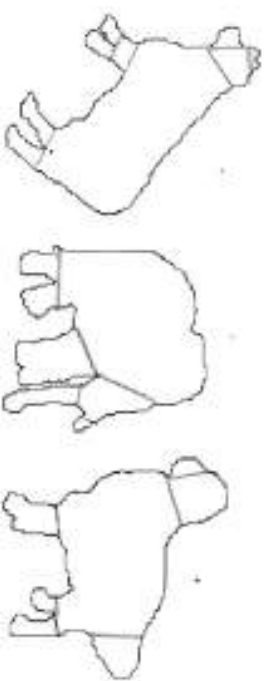
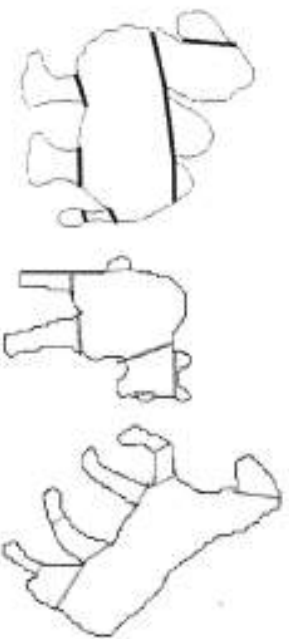
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Likhanya Mchani

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

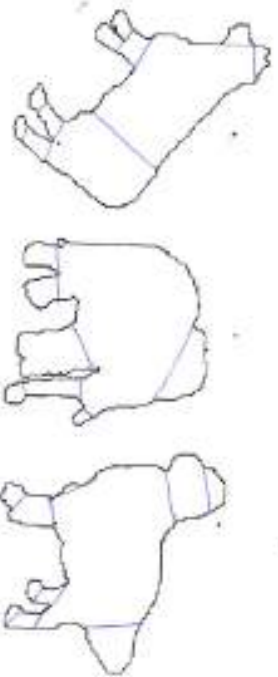
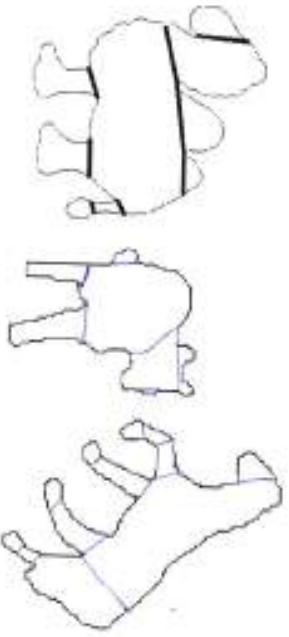
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (a little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

Efrate Selamawit

Age:

☒ 17-20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

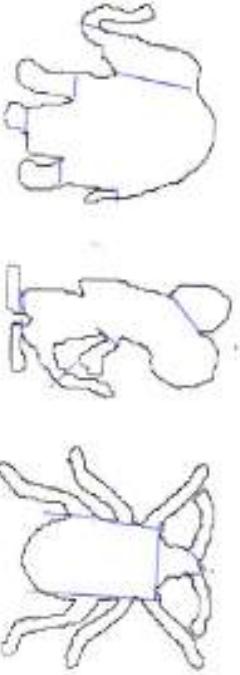
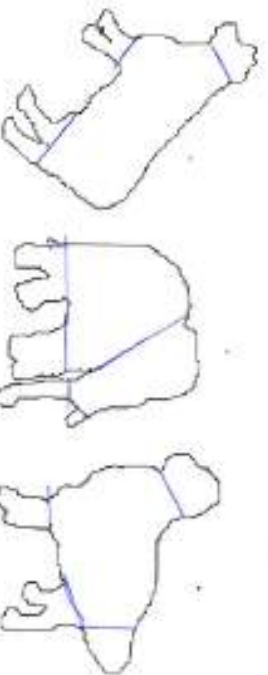
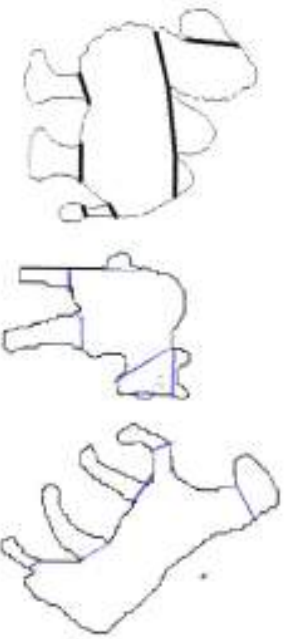
Gender:

☐ Male ☒ Female

Race:

☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

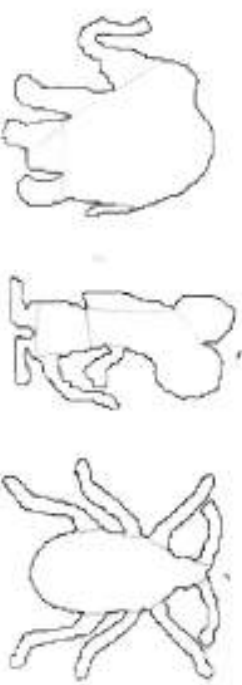
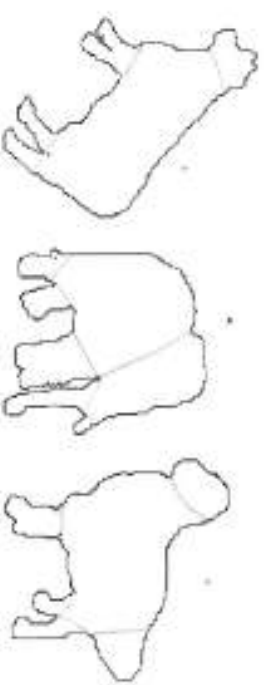
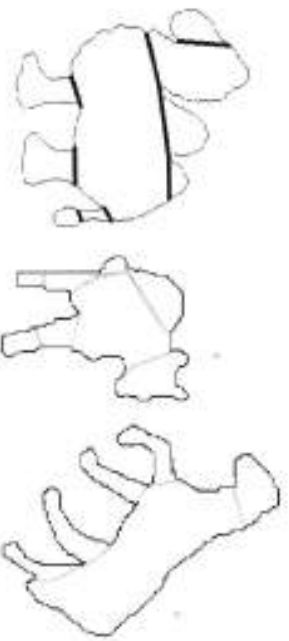
Name: Harper Schubert

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, please break up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

Musonda Fumpe

Age:

<20 20-30 30-40 40-50 50-60 >60

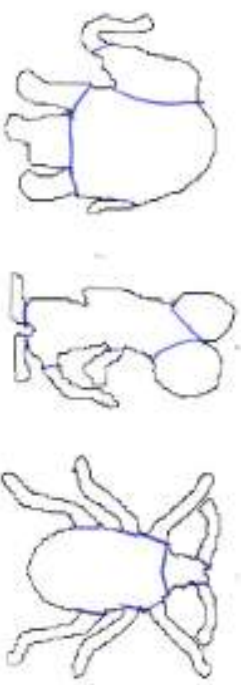
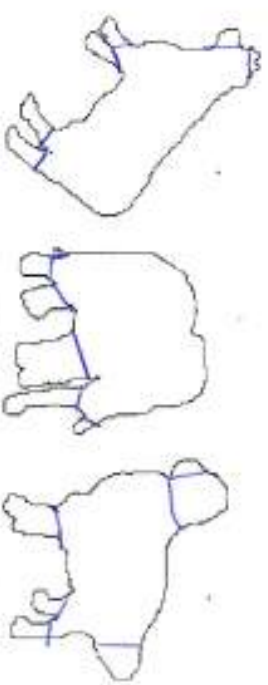
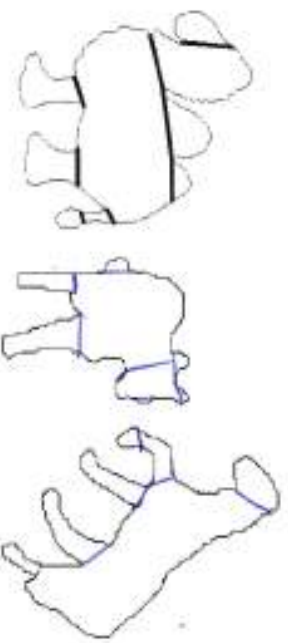
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Rune Jensen

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

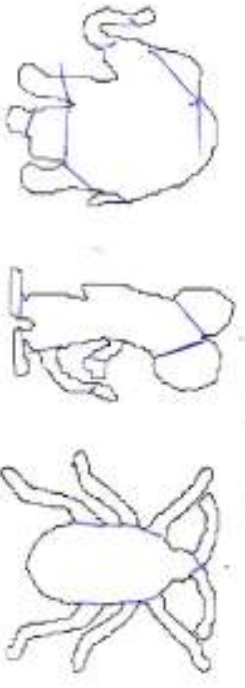
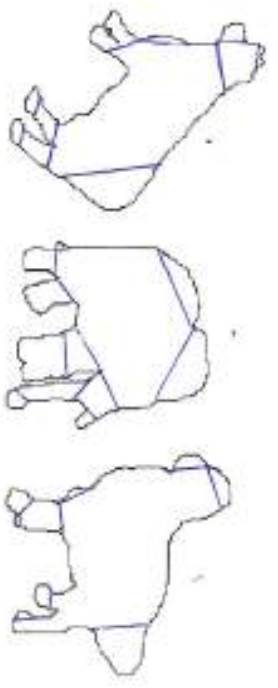
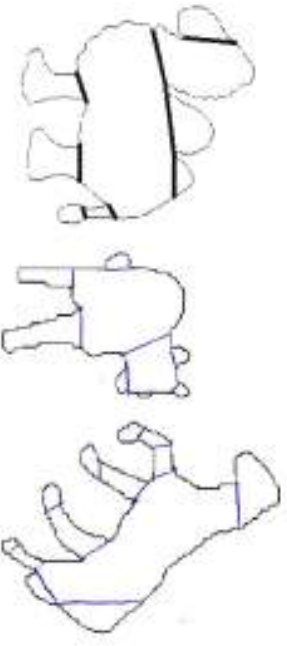
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Nao Hayashi

8/6

Age:

<20 ✓	20-30	30-40	40-50	50-60	>60
-------	-------	-------	-------	-------	-----

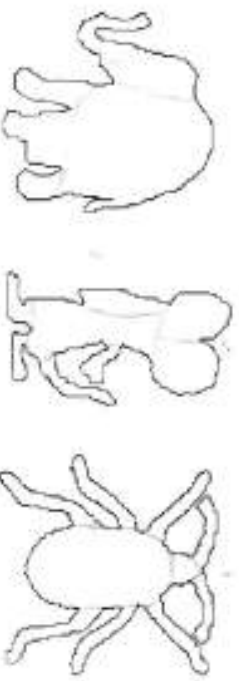
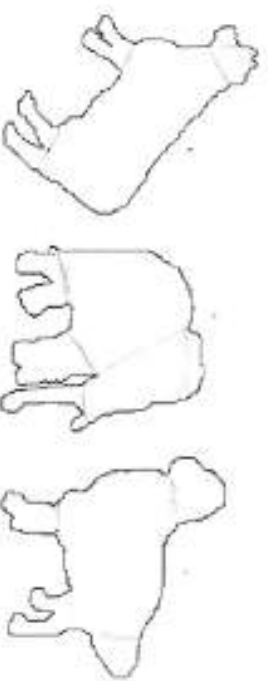
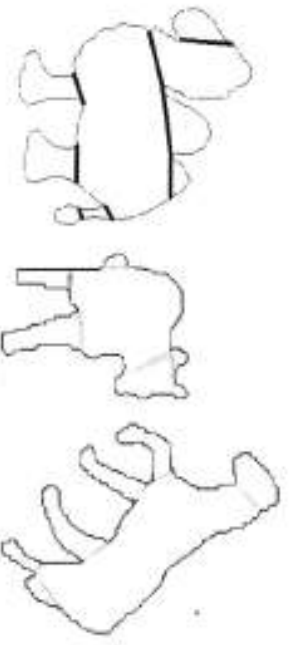
Gender:

Male ✓	Female
--------	--------

Race:

African ✓	Indian	Caucasian	Asian	Other
-----------	--------	-----------	-------	-------

By using straight lines, please break up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Anayik

Age:

☒ ≤20	☐ 20-30	☐ 30-40	☐ 40-50	☐ 50-60	☐ >60
---	--------------------------------	--------------------------------	--------------------------------	--------------------------------	------------------------------

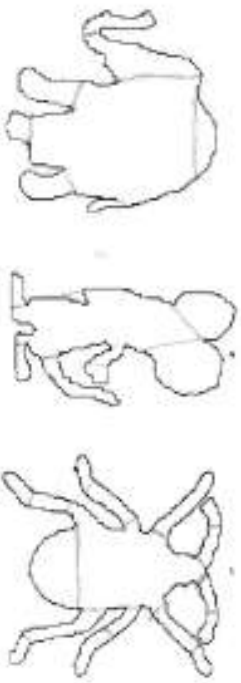
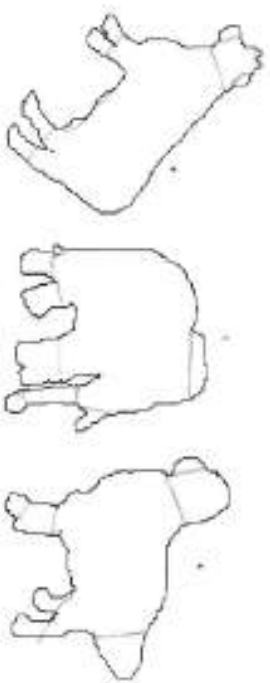
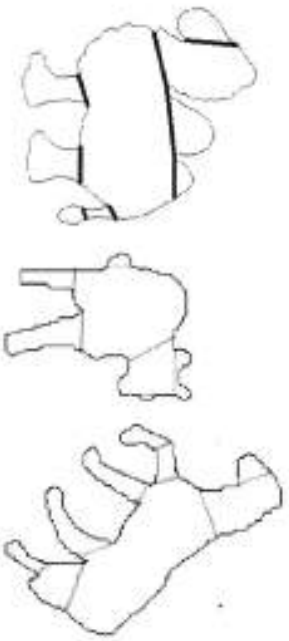
Gender:

☐ Male	☒ Female
-------------------------------	--

Race:

☒ African	☐ Indian	☐ Caucasian	☐ Asian	☐ Other
---	---------------------------------	------------------------------------	--------------------------------	--------------------------------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

Victoria Mphahane

Age:

<20 20-30 30-40 40-50 50-60 >60

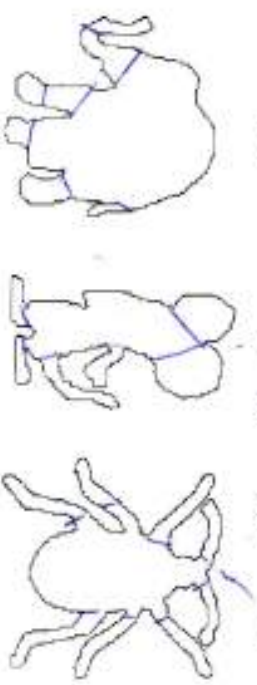
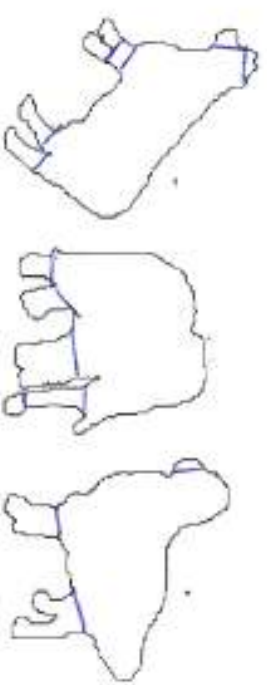
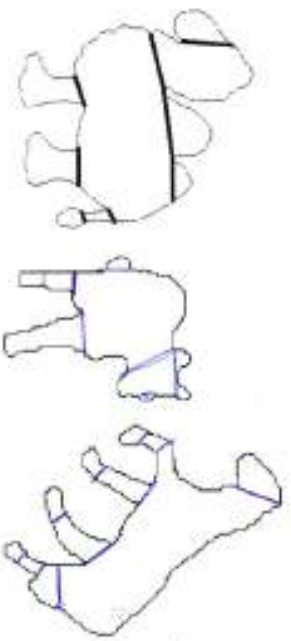
Gender:

Male Female

Race:

African Indian Caucasian Asian Other

By using straight lines, please break up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Neo Melville

Age:

<20 ✓	20-30	30-40	40-50	50-60	>60
-------	-------	-------	-------	-------	-----

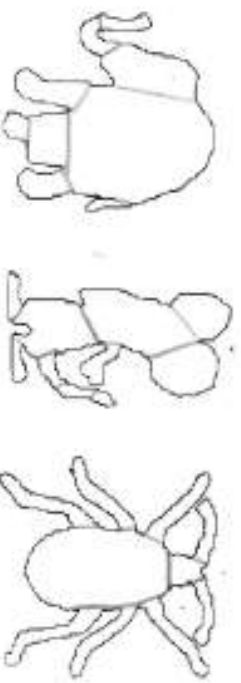
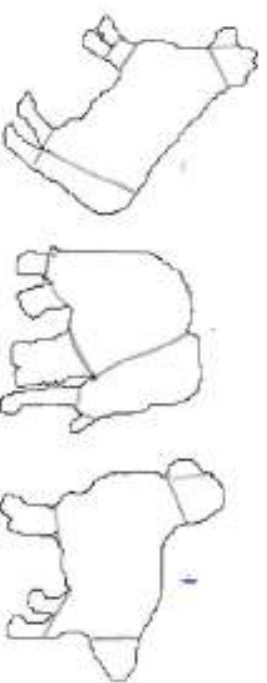
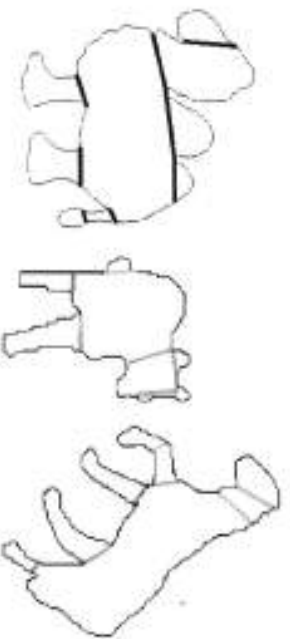
Gender:

Male	Female ✓
------	----------

Race:

African ✓	Indian	Caucasian	Asian	Other
-----------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

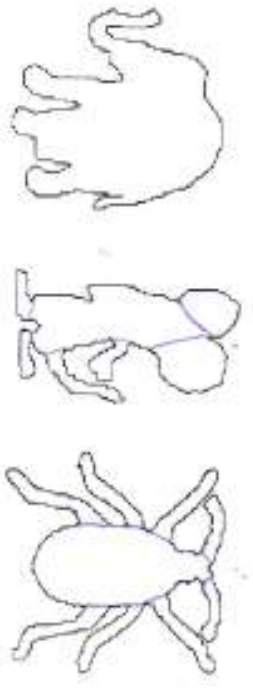
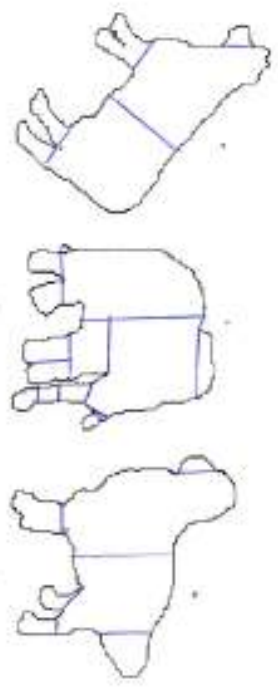
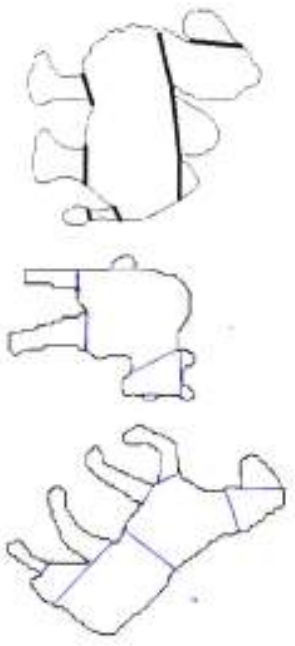
Name: Kyahle Mafane Gr: 8g

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Olivia

Age:

<20	20-30	30-40	40-50 ✓	50-60	>60
-----	-------	-------	---------	-------	-----

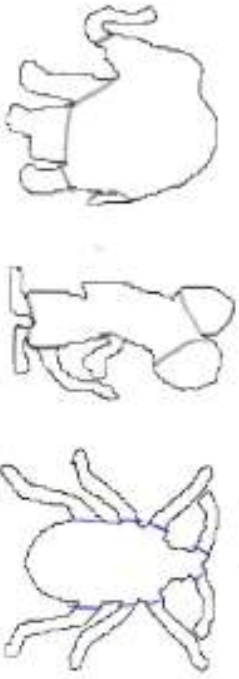
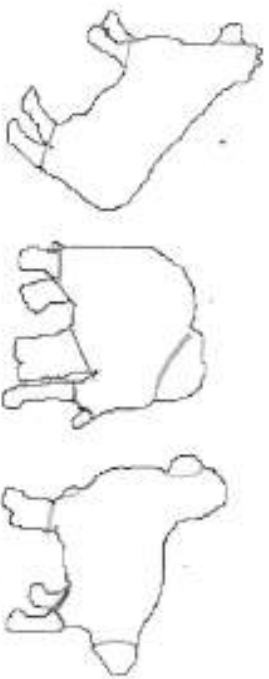
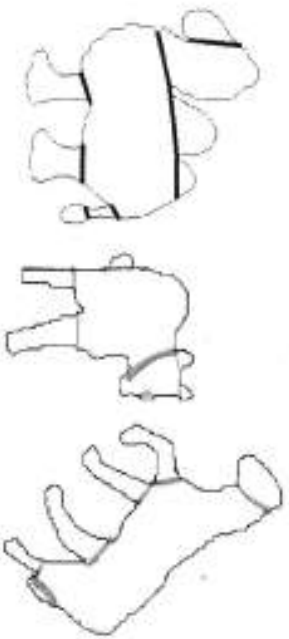
Gender:

Male	Female ✓
------	----------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Uyenda Mathenywa

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

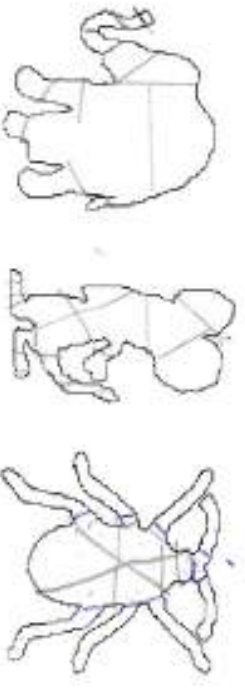
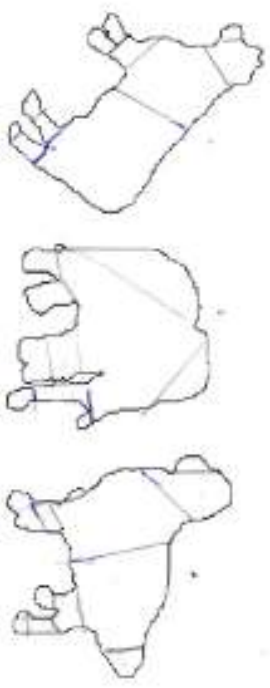
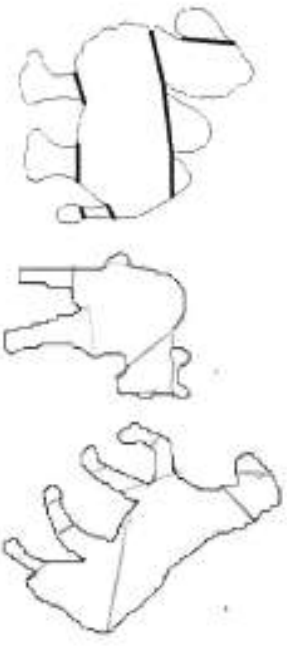
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

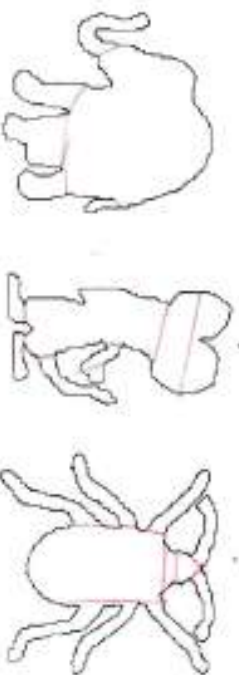
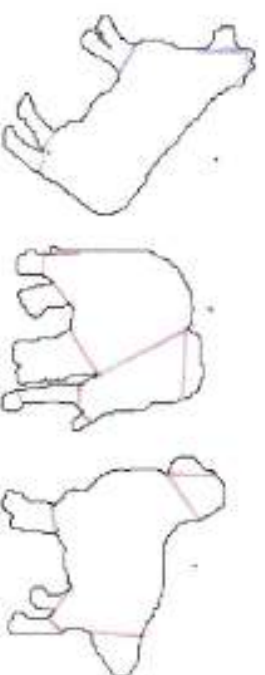
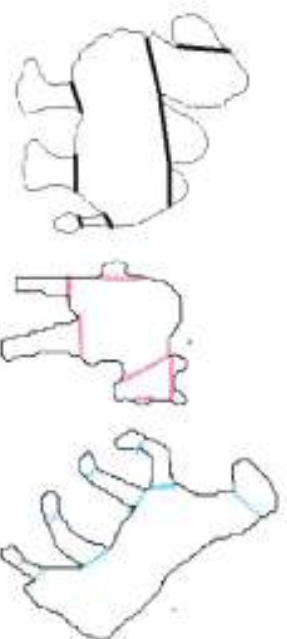
Name: Zenoib Mochengarcia

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: Male Female

Race: African Indian Caucasian Asian Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Keilumetse Morphi

Age:

<20 ☒	20-30	30-40	40-50	50-60	>60
---	-------	-------	-------	-------	-----

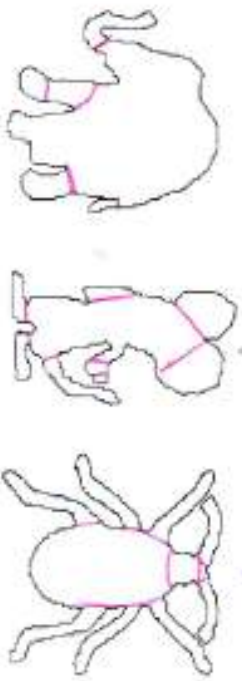
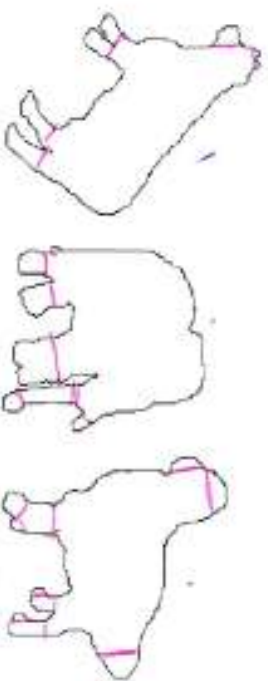
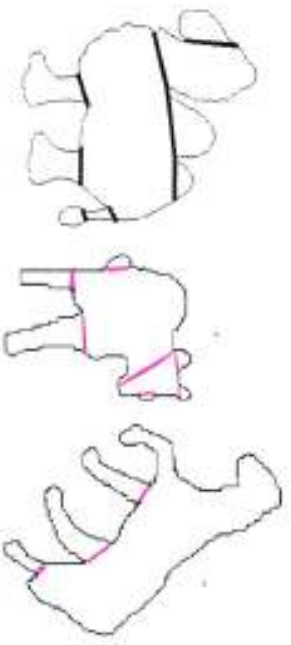
Gender:

Male	Female ☒
------	--

Race:

African ☒	Indian	Caucasian	Asian	Other
---	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

Ortelle Khatso

Age:

☒ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ >60

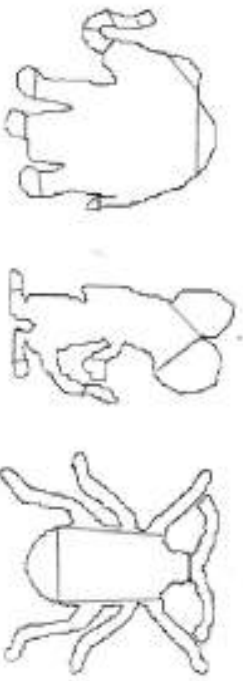
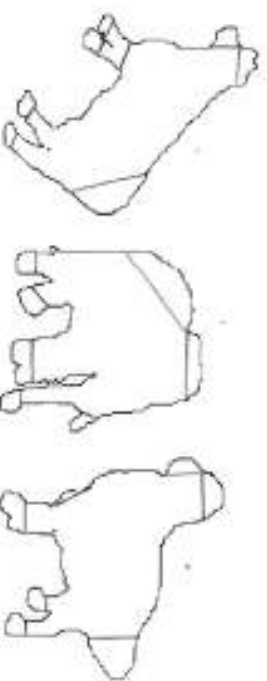
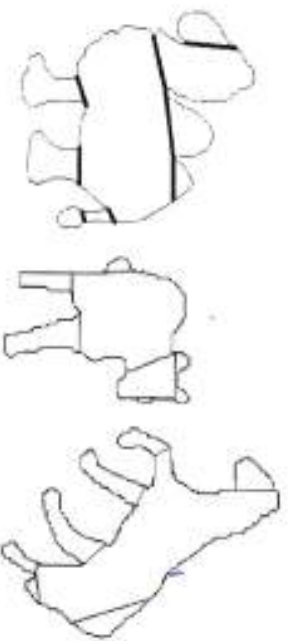
Gender:

☐ Male ☒ Female

Race:

☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

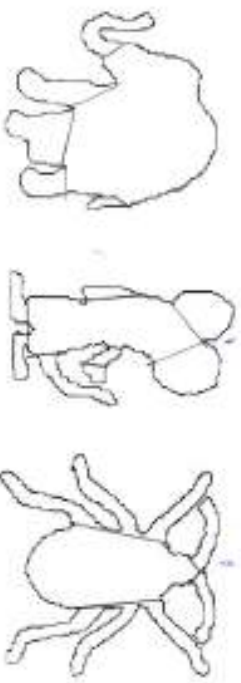
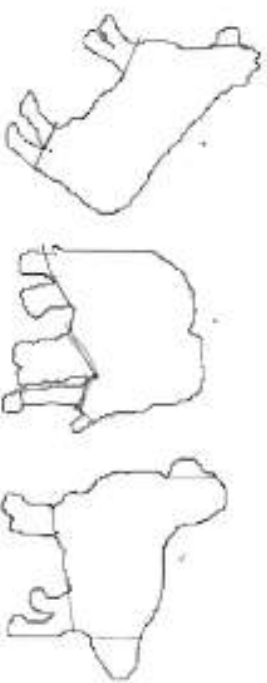
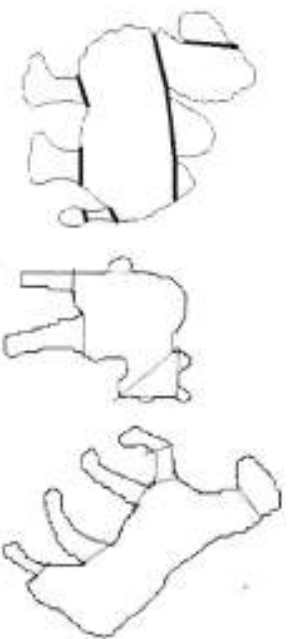
Name: Agnes negak

Age: <20 20-30 30-40 40-50 50-60 >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name:

Jhanda Mayen

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

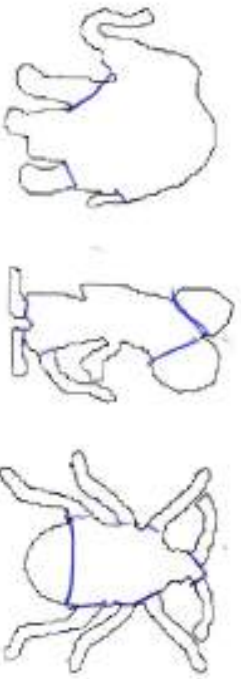
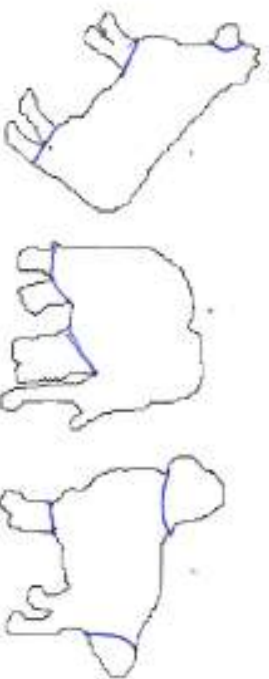
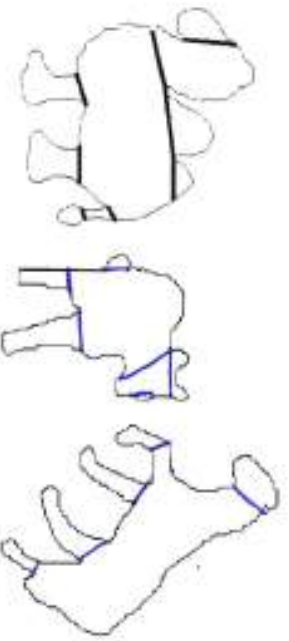
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Zintle Mafika

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

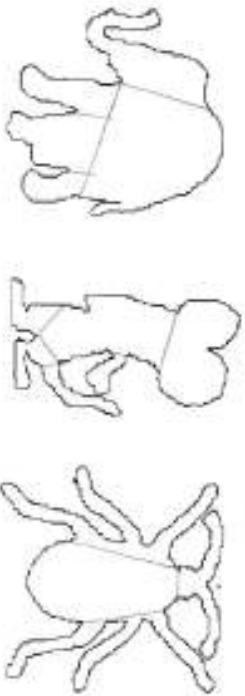
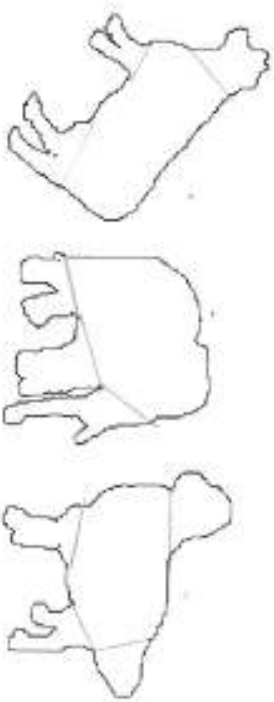
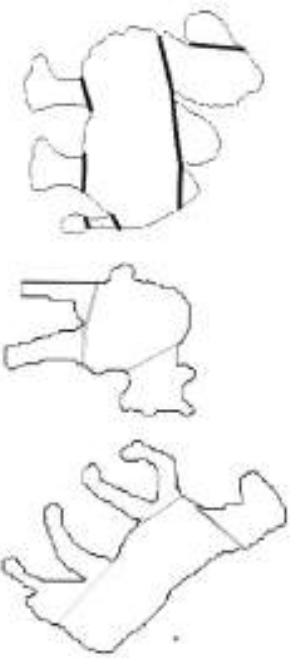
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, can you please decompose (break-up into parts) these shapes into the minimal number (as little parts as possible) of parts:



Shape Decomposition Questionnaire

Name: Nivine Gellera

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

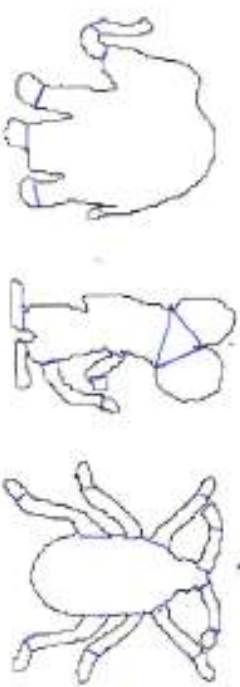
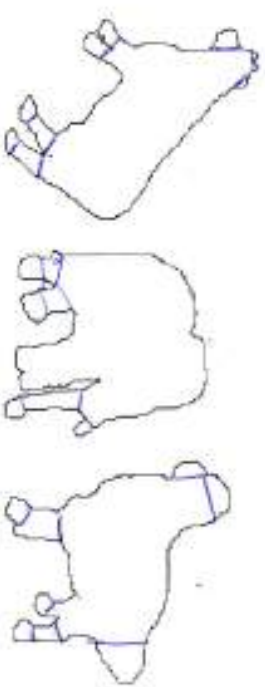
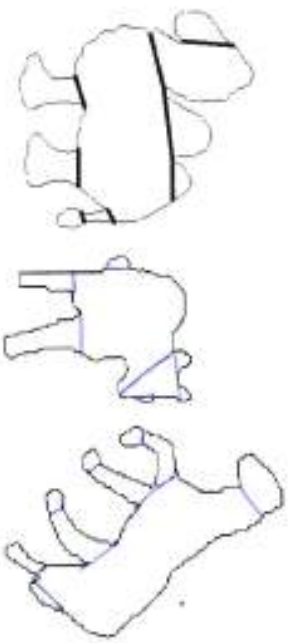
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Chadong Jha

Age:

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

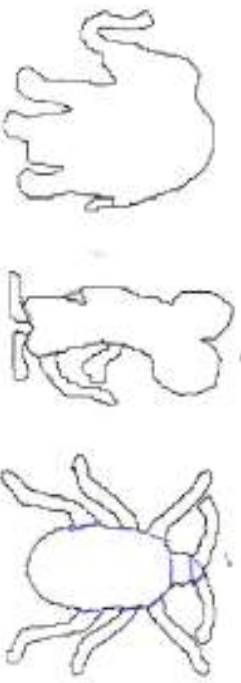
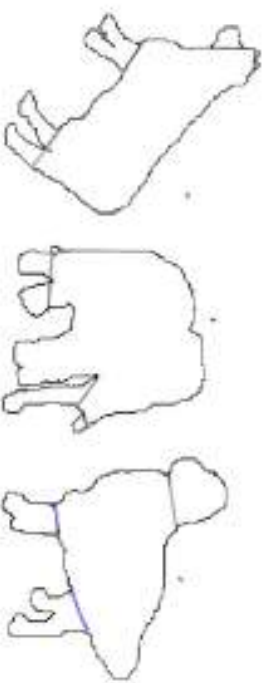
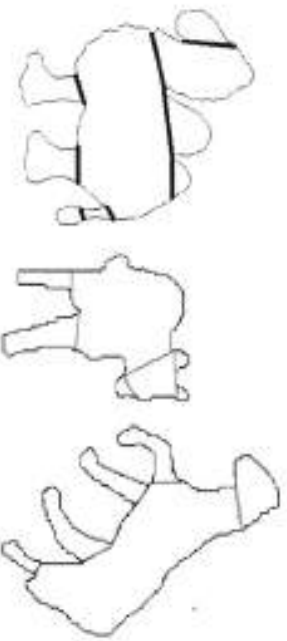
Gender:

Male	Female
------	--------

Race:

African	Indian	Caucasian	Asian	Other
---------	--------	-----------	-------	-------

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

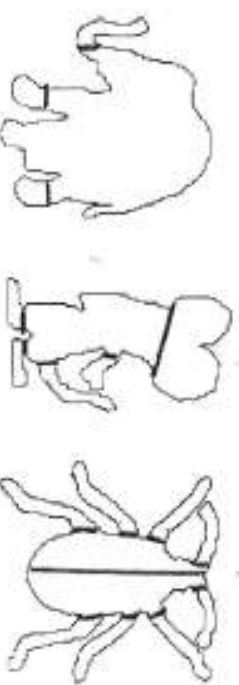
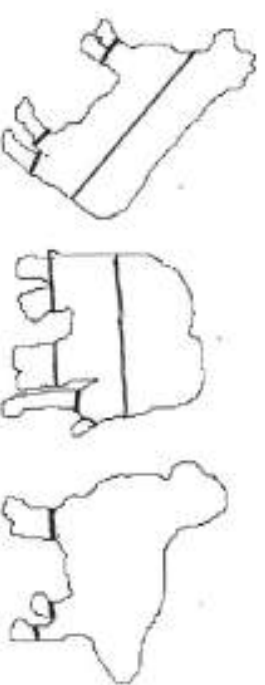
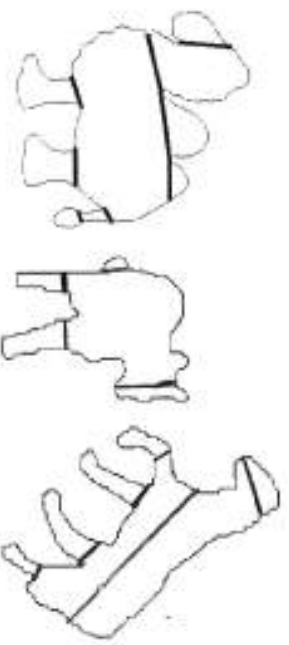
Name: Jaylon Gathelagat

Age: 15 ☐ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☒ 50-60 ☐ >60

Gender: ☐ Male ☒ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

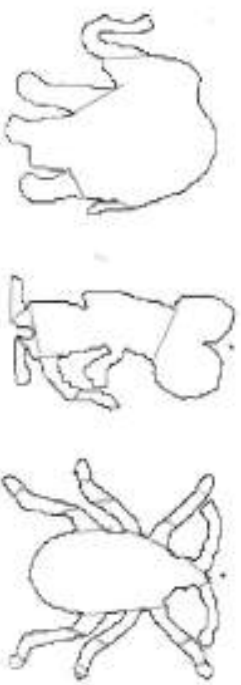
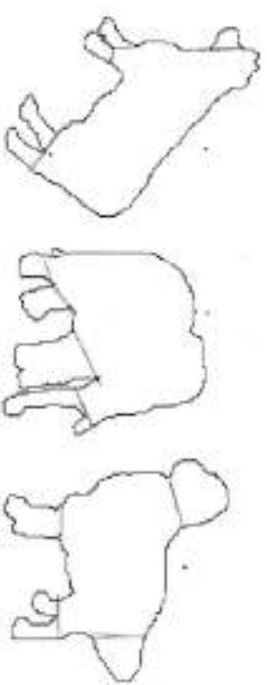
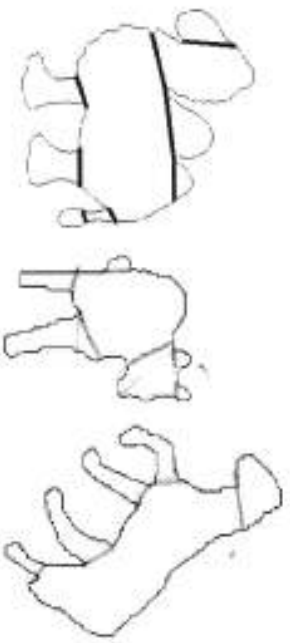
Name: Deall Twa

Age: 4 ☐ <20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☒ >60

Gender: ☒ Male ☐ Female

Race: ☒ African ☐ Indian ☐ Caucasian ☐ Asian ☐ Other

By using straight lines, please break-up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.



Shape Decomposition Questionnaire

Name: Keketsu R

Age: 16

<20	20-30	30-40	40-50	50-60	>60
-----	-------	-------	-------	-------	-----

Gender:

☒ Male	☐ Female
--	---------------------------------

Race:

☒ African	☐ Indian	☐ Caucasian	☐ Asian	☐ Other
---	---------------------------------	------------------------------------	--------------------------------	--------------------------------

By using straight lines, please break up these shapes into the minimal number (as little as possible) parts. Look at the first picture as an example.

