

Equine Utrecht University Scale for Automated Recognition in Facial Assessment of Pain – EQUUS-ARFAP

Bram Jonkers – ICA-4057538 – Universiteit Utrecht – b.jonkers@uu.nl – June 20, 2018

Under supervision of:

prof. dr. R.C. Veltkamp – Department of Information and Computing Sciences – Universiteit Utrecht,

dr. J.P.A.M. van Loon – Department of Equine Sciences – Universiteit Utrecht.



Universiteit Utrecht

Master Thesis for:

Universiteit Utrecht – Department of Information and Computing Sciences – Game and Media Technology

Abstract:

The EQUUS-ARFAP, or the Equine Utrecht University Scale for Automated Recognition in Facial Assessment of Pain, is a project striving to lay the foundations for and prove the feasibility of an application that can automatically assess pain-levels in horses through facial photos. So far only one similar project exists that works on sheep [1], which is why this project aims to expand upon and broaden this field of automated pain recognition tools for animals and veterinary use.

The theory behind the application is based on a combination of the EQUUS-FAP [2] and the HGS [3] pain scoring systems, combining their scoring systems into a single-moment facial-data only pain-scoring scale which is applicable on facial photographs of horses.

To create a first version of this application several computer-vision based techniques were used that have been founded on human facial feature recognition. These have been altered to work on the facial features of horses, through the use of medical information available on them. Active shape models, histograms of gradients, colour space conversions, image thinning, support vector machines and cross validation are some of the techniques that were used to realize this application.

Results comparable to those of a similar application have been achieved, so it can be safely concluded that this kind of application is feasible and that a basis for future research to continue from has been provided.

Table of Contents

Abstract:.....	1
1 – Introduction	4
2 – Research Aim.....	5
2.1 – Research Objectives	5
2.2 – Research Questions.....	5
3 – State of Art	6
3.1 – Pain Recognition	7
3.1.1 – Facial Feature Recognition	7
3.1.2 – Equine Pain Recognition	8
3.2 – Computer Vision.....	12
3.2.1 – Feature Detection	12
3.2.2 – Feature Data Conversion.....	15
3.2.3 – Data Comparison.....	17
3.2.4 – Data Validation.....	18
3.3 – Related Work	18
3.3.1 – Subtle Facial Expression Detection	18
3.3.2 – Emotion Detection through Facial Expressions	18
3.3.3 – Pain Recognition through Facial Expressions.....	18
3.3.4 – Automated Sheep Pain Recognition	19
4 – Methodology.....	20
4.1 – Data Acquisition	20
4.2 – The Application	23
4.2.1 – The Algorithms	23
4.3 – Project Validation.....	24
5 – Implementation	25
5.1 – The Application	25
5.2 – The Algorithms	26
5.2.1 – Loading (Image) Data	27
5.2.2 – Loading Data Statistics	27
5.2.3 – Main Tasks.....	28
5.2.4 – Validation	38
5.2.5 – Data Subsets.....	38
5.2.6 – ASM Training/Creation.....	40
5.2.7 – (Intermediate) Result Saving.....	41
6 – Outcome.....	42

6.1 – Results	42
6.2 – Evaluation.....	44
6.2.1 – General Remarks	44
6.2.2 – Strengths and Weaknesses	45
6.2.3 – Result Comparison	46
7 – Conclusion	47
7.1 – Drawn Conclusions.....	48
7.1.1 – Selected Features.....	48
7.1.2 – Single Facing.....	49
7.2 – Future Work	50
7.2.1 – Improvements	50
7.2.2 – Expansions.....	50
Acknowledgements.....	51
References	51

1 – Introduction

EQUUS-ARFAP is a combined project of the Utrecht University branch of Computing Science and Equine Science, striving to lay the foundations for and prove the feasibility of an application that can automatically determine the pain-score of a horse based on facial features read from photographs. No such application exists as of yet, therefore this project explores the possibilities of such an application, as well as how it could be made. The future goal of this project is to eventually reach the same level of results that a (trained) veterinary would achieve when deciding the pain-scores of the horses within the images it receives.

The pain-scores to be calculated are (based on) a combination of the EQUUS-FAP [2] (the Equine Utrecht University Scale for Facial Assessment of Pain) and the HGS [3] (Horse Grimace Scale) pain scores. These are scores made to describe the pain-levels of horses purely based on (dynamic) facial information. To automatically generate these pain scores, many advanced computer-vision based techniques will be discussed, as each pain-scoring feature has to be described in a manner that is computer-vision friendly. There is a lot of work done on facial feature and gesture recognition in human faces for pain recognition [4] and there is at least one similar sheep-based project that shows that this knowledge should be transferable to other animals as well [1]. However, as no application of this specific sort yet exists, the aim of this project is to improve and expand upon this field of knowledge on automated pain recognition in non-human animals by adding techniques for a second animal, the horse.

The two main things this project aims to add to this field are (a foundation for) the automated recognition and annotation of horse faces, and the conversion (and testing thereof) from traditional pain-feature descriptions of horses to a digitally comparable one. The first is done by using techniques like Active Shape Models [5] to describe the general form and variations within the (landmarks of the) face of a horse. The second by turning these features from physical descriptions like “Showing of muscle tension”, “Direction of ears” and “Sclera visible” into digitally comparable ones like Histograms of Gradients [6] and Shape Densities [7] to compare shapes, and Colour Histograms [8] to check for shadows and colour-based features.

Parallel to this project another one was held at the Department of Equine Sciences to gather data on the pain scores of horses after clinically induced injuries, which has been used to create a database of images and accompanying scores to work with. Using this data, the application has been trained so that it can analyse and generalize this data, in the hopes of making it work on untrained/completely new data as well. As this data was made by professional veterinaries the validity and credibility of this data can be assured. Therefore, it can be generalized that if this application can create similar scores to these, it has reached a desired quality of results.

Due to the scope of the project and the timeframe it must be executed in, there are a few possible expansions to be left open for future work/adaptations. For example, if a Morphable Model [9] were to be constructed of the equine face, a 3-Dimensional reconstruction of those represented in the photos could be created, which would allow the application to read certain data in much more efficient ways than a 2-Dimensional Statistical Shape. However, creating such a model (and the database therefor especially) would be a project as big as this in its own right.

As our project is a proof of concept first and foremost, it focuses on implementing and adapting proven and existing techniques, as more experimental and innovative techniques that could further improve the application should only be looked at once a solid basis has been created. Another focal point of this research is the reporting of possible expansions and improvements found along the way, to help steer future research on this subject in the right direction.

2 – Research Aim

The aim of this research is to explore the possibilities of an application that can automatically generate pain-scores based on facial photo data of horses, and to create a basis/first iteration of this application.

The eventual application will cover multiple different tasks like data extraction, conversion and comparison. An important focus of this project is therefore to keep these tasks modular, so that each of them could be iterated on independently and future expansions/improvements can be more easily implemented.

2.1 – Research Objectives

To realize this application, the following set of issues has to be resolved. Therefore, these are the research objectives of this study.

1. The facial photo (data) must be processed, labelled and categorized to make them usable for the creating of training sets and to be able to properly judge and use them later.
2. The actual horse face present within the photos must be found, so irrelevant background information can be filtered out.
3. The face must be split up into different (relevant) features, therefore a clear definition of these and some way of finding them within the face are necessary.
4. The facial features must be converted into readable/comprehensible data, so analyses and comparisons can be performed.
5. This data must be usable to create a pain-score from by comparing it to examples created from the training set.
6. This pain scoring needs to be evaluated, which can once again be done using the training data.

2.2 – Research Questions

To describe the aim of our research the following overarching research question has been created:

“Can an application be created that, when trained using known professionally scored input data, can automatically determine the pain-level of a horse in an image, at the level of a (trained) veterinary?”

Based on the research objectives created from this question, the following set of sub questions have been formulated:

1. *“How can pain-scores be determined on our training data, and which features are most relevant for this objective?”*
 - Based on research objective 1.
2. *“How can these features be extracted from the training images?”*
 - Based on research objectives 2 and 3.
3. *“How can these features be turned into quantifiable data?”*
 - Based on research objective 4.
4. *“How can this data be compared and generalized to create a trained model of these features?”*
 - Based on research objective 5.
5. *“How can the accuracy of this application be validated and measured?”*
 - Based on research objective 6.

Answering these should allow the main research question to be answered, and the relevant information and implemented methods therefor will be discussed within the following sections.

3 – State of Art

Automatic recognition of pain-levels in animals is a very desirable technology as (unlike with humans) gathering data on these pain-levels is a lot harder, due to it relying solely on extrinsic values. This is also a field that is, as of now, relatively unexplored; only a handful of applications and/or studies of the sort exist.

The reason for choosing to perform pain recognition based on facial features is due to problems arising in non-human animals when it comes to pain recognition. This is because they cannot self-report on their (pain-)state and information must be gained through other, objective (and usually non-verbal) measures [10]. One system often used on animals would be a system using grimace scales based on facial features, as there are a lot of physical tells that can be gained from these [11].

Facial feature recognition is, luckily, a domain well worked out in human computer vision techniques, and one that tends to use very adaptable techniques. Applying techniques therein on similar data accrued on other animals could allow for the recreation of these, which could in turn allow for the creation of an automated pain recognition application for other non-human animals.

For starters, let's take a look at the basis of (facial) pain recognition for humans. Large studies have been undertaken to explore the type of feature detections that can be performed on the facial expressions of humans [12] [13] [14], and what types of information can be gained from them. Combining this with existing Computer Vision techniques has already resulted in automated software being made to recognize and evaluate these features [15] [16], up to the point where automated pain recognition has now become possible for humans [17] [4].

The field of animal (facial) pain recognition has seen a lot less application development and documentation over the years, but there definitely is no lack of available data and research to use. There are several pain-scales created for horses [18] [19] [20], and especially the EQUUS-FAP and HGS, two pain-scoring systems that focus only on facial features, could be very compatible with an automated application [2] [3] [21] [22]. There have also been studies to mapping the unique facial features of horses in much the same way as they have been done for humans [23], which would make a facial feature detection application something very realistically achievable. Combining these should give a solid basis on which an automated pain recognition application could be build.

To build such an application many advanced computer-vision techniques need to be used, as the application needs to perform object [24] [25] [26] and feature [8] [27] [28] detection [5] [29] [30], data extraction/conversion [31] [32], comparison [33] [34] and validation of all its processes, techniques and algorithms [35].

One major source of inspiration and achievability of this project is the automated pain estimating software based on facial action units of sheep [1]. This project shows a fitting example of how the techniques developed for humans can be combined with the information available on other animals like sheep. As nearly all the same types of information that are available for sheep are available for horses as well, it makes sense that some of their ideas should be generalizable to other animals too, which is what we tried to confirm through this research.

In the following sections more in-depth information concerning the important discoveries, techniques and features prevalent in these fields of pain recognition and computer vision are given, as well as some examples of projects combining these fields, giving a brief history on the automatization of pain recognition.

3.1 – Pain Recognition

To answer the first sub question, “*How can pain-scores be determined on our training data, and which features are most relevant for this objective?*”, an investigation into the field of pain recognition should be performed, to see how and based on what features pain recognition is usually done. As these ideas need to be applicable to horses, ergo non-human animals, it has already been mentioned that an effective way to do so is by using facial features, as this is a good objective and reliable way to do so, especially in non-self-reported analysis [10]. Facial features that can be used for pain scoring have been well documented both for humans and horses, which means that it should be possible to transfer generalizable ideas between the two, which helped a lot with the incorporation of certain ideas into our application.

3.1.1 – Facial Feature Recognition

Facial feature detection (on humans) is generally done using facial Action Units (AU). Facial AU are measurable muscular activities within the face that produce a change in the facial appearance of a person [12], as can be seen in Figure 1. Based on AU two main techniques have been created to read more information about the facial features and expressions of a person; the Facial Action Coding System and Facial Landmarks.



Figure 1. Example of Action Units (the orange dots) on a human face. (Image source: [12]).

3.1.1.1 – Facial Action Coding System

The Facial Action Coding System (FACS) is a system that scans images of the human face for slight differences in AUs over time [13]. This allows for the comparison in AU changes in different expressions and it has been proven that this is a valid method of measuring emotional and social reactions/states from faces.

3.1.1.2 – Facial Landmarks

Facial Landmark features are more so an extension to Action Units than that they are a different concept [14]. Facial landmarks were developed to improve the variance of AU detection on images with different lighting conditions, view positioning and size. Facial landmarks are constructed by combining stable sets of AU in the face to sets, which are much more robust to change than the single features of which they are constructed, see Figure 2. Facial landmark techniques have even been improved to accommodate images of high expressivity, making them a very strong candidate for emotion/expression detection.

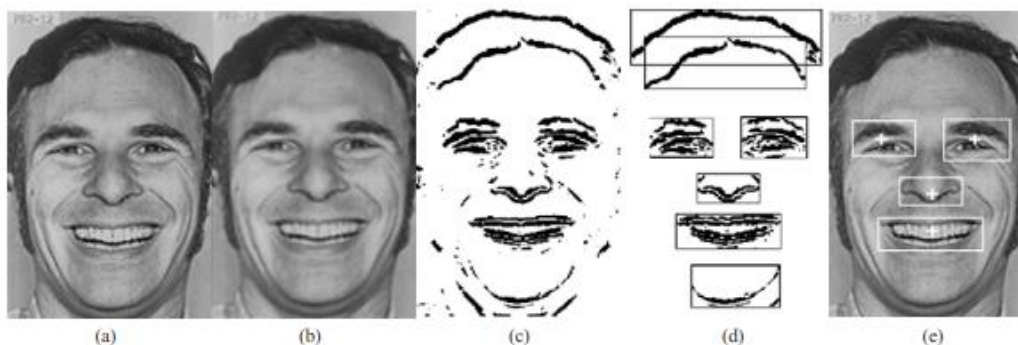


Figure 2. Example of the construction of a set of facial landmarks from a human face. (Image source: [14]).

3.1.1.3 – Equine Facial Features

A similar set of facial landmarks has been defined and described for horses as well. Thorough research has been done into deriving a proper set of AUs to create a FACS that has been used to create landmark sets describing the features of an equine face in detail [23]. The AUs used are based on anatomical definitions and analysis of the domestic horse. Using high detail cameras very subtle (underlying) muscular movements have been identified and described, creating a highly complete set of equine facial feature data. For an example of the key features defined for horses throughout this research see Figure 3.

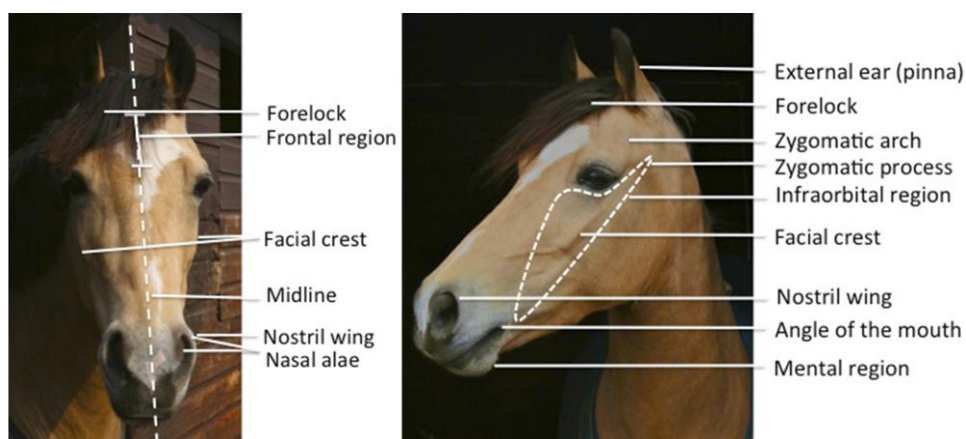


Figure 3. The facial landmarks of a horse. (Image source: [23]).

3.1.2 – Equine Pain Recognition

Due to pain recognition on non-human animals having to rely on objective (feature) data, pain scoring systems for these kinds of animals have been made to allow for a structured and regulated way of doing so. For horses there are a lot of ways to assess pain [19], as features like a pained face, the posture of certain body parts, the types of activities that are displayed and even how reactive a horse is to specific things can all be factors used to determine pain-levels of different types of pain. For a summary of these features see Figure 4.

Using these (and more) features, several pain assessment scales/tools have been made over the years, each focusing on different parts of the horse or on specific symptoms directly related to specific types of pain [20]. Some notable tools used for the creation of equine pain assessment scales are the visual analogue scale (VAS), numerical rating scale (NRS), simple descriptive scale (SDS), time-budget analysis, composite pain scale (CPS) and facial expression-based scales.

VAS scales calculate a continuous variable that represents pain-levels based on a marker placed on a ruler-like scale, where one side represents no pain at all and the other represents the strongest

imaginable level of pain. The accuracy of VAS scales has however been questioned, as (self-)reporting tends to lead to biased/more discrete results and be less reliable on lower and middle based scores.

NRS is a discontinuous/discrete variable scale where value markers are placed at equally distanced points between 0 and 10 to represent different stages from no to maximum levels of pain. NRS scores are more accurate under repetition than VAS scales but do tend to be less sensitive to small changes.

SDS is an ordinal scoring system where scores like 0, 1, 2 ... can be assigned to descriptive values like “no pain”, “mild pain”, “moderate pain” ... to describe pain levels. SDS tends to be at a similar level of usefulness as the VAS and NRS systems.

Time-budget analysis is a log of activities as done by an animal and the time spent (as a percentage of total time) on these activities over a certain analysed period of time (usually observed through filming). This method is in general a very good one and can register very subtle tells of pain, however its large setup and equipment requirements and inability to be done quickly/in real-time make it unsuitable for direct clinical applications.

CPS combines multiple SDS’s (based on different features), assigns weights to these and then combines them into one Composite Pain Scale. CPS is used in a large amount of studies and appears to have good inter-observer reliability. One downside to CPS is that it requires large amounts of repeated evaluation to properly balance the weight of all included features, and that this balancing requires large amounts of practical knowledge to perform.

Facial expression-based scales are scales of the above-mentioned types (usually CPS) that are based solely on facial features. These scales have been a much-researched topic in recent years, as they are a reliable method of testing for more subtle and sensitive changes in pain.

Behaviour category	Score				
	0	1	2	3	4
Pain face	No pain face		Pain face present	Intense pain face	
Gross pain behaviour*	None		Occasional		Continuous
Activity	Exploring, attention towards surroundings or resting	No movement		Restless	Depressed
Location in the stall	At the door watching the environment	Standing in the middle, facing the door	Standing in the middle facing the sides	Standing in the middle facing back or standing in the back	
Posture/weightbearing	Normal posture and normal weight bearing	Foot intermittent of the ground/ occasional weight shift	Pinched (groove between abdominal muscles visible)	Continuously taking foot off the ground and trying to replace it	No weightbearing. Abnormal weight distribution
Head position	Foraging, below withers or high	Level of withers	Below withers		
Attention towards the painful area	Does not pay attention to painful area		Brief attention to painful area (e.g. flank watching)		Biting, nudging or looking at painful area (e.g. flank watching)
Interactive behaviour	Looks at observer or moves to observer when approached	Looks at observer does not move	Does not look at observer or moves away avoids contact	Does not move, not reacting/ introverted	
Response to food	Takes food with no hesitation	Looks at food		No response to food	

* Gross pain behaviour includes all readily visible behaviours such as excessive head movements (vertical/lateral), flehmen, kicking, pawing, rolling, tail swishing, mouth playing, stretching, etc.

Figure 4. A collection of features used for equine pain measuring collected from several combined studies. (Image source: [19]).

3.1.2.1 – Equine Facial Pain Scoring

One of the main manners of testing for pain in recent years is through the use of facial expressions/features. This is because they can give very subtle data, allowing for the recognition of mild levels of pain as well as high levels. A resulting pain scale thereof is called a Grimace scale, which is a pain scoring system based solely on facial data [11].

These scales have also been made for horses and many studies have been done into identifying their grimace pain-scales/the equine pain face [18], of which an example can be seen in Figure 5. These scales are strongly based on the human versions of them, and therefore allow for them to be used in much the same way. The equine pain-face can be deconstructed into five primary features; Asymmetrical/Low-hanging ears, angled appearance of the eyes, withdrawn/tense stare, mediolaterally dilated nostrils and tension of specific facial muscle groups (those around the lips and chin for example).

From these features described in the equine pain-face, two main pain-scoring systems have been derived; the HGS and the EQUUS-FAP.

The HGS [3], Horse Grimace Scale, is the first attempt at creating a standardized facial-expression based pain scoring system based on the human equivalent thereof. The HGS is based on six main facial landmarks and has three recognizable states for each of them. All six of these landmarks can either be in a state directly linkable to pain, a state that is not linkable to pain, or somewhere in between. These three states are then scored with a value from 0 to 2 based on how well the state shows pain. A horse can then be scored by adding up the pain scores related to each of the states in which their six landmarks are in, meaning the total score can range from 0 to 12. For an in-depth example on the features that are scored on see Figure 6.

The EQUUS-FAP [2], Equine Utrecht University Scale for Facial Assessment of Pain, is based on nine main landmarks/scoring points instead of six, covering slightly more nuances and tells in the equine faces. These nine points also include a few dynamic features that can only be registered over time, like the inclusion of head motion and flehming/yawning. This does make the system less applicable to single still images however (as would be more desirable in automation). Apart from these features the system works in much the same way, scoring every feature from 0 to 2 and adding the total score to create a pain score ranging from 0 to 18. For an example of these features, see Figure 7.

As these two pain-scoring systems were constructed and investigated around the same time, there are lots of tests, comparisons and validations done on both these systems [21] [22]. And as they are two of the best and most used systems as of now they have also been the basis for the one used and created within this project.

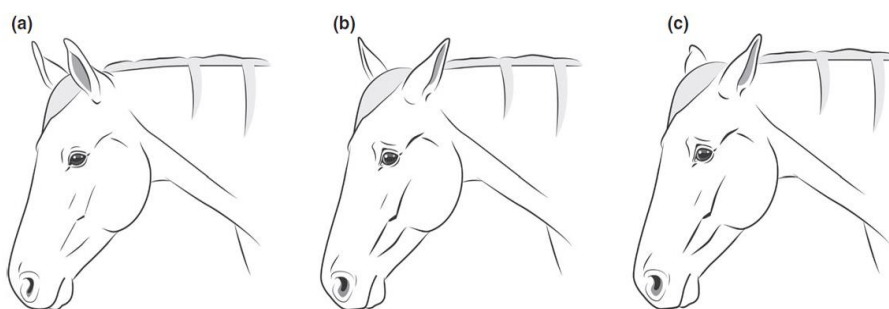


Figure 5. Example of a pain-free (a) horse face and faces of acute pain (b) and (c). (Image source: [18]).



Figure 6. Example of the facial landmarks and states thereof used in the Horse Grimace Scale (HGS). (Image source: [3]).

Data	Categories	Score
Head	Normal head movement/interested in environment	0
	Less movement	1
	No movement	2
Eyelids	Opened, sclera can be seen in case of eye/head movement	0
	More opened eyes or tightening of eyelids. An edge of the sclera can be seen 50% of the time	1
	Obviously more opened eyes or obvious tightening of eyelids. Sclera can be seen >50% of the time	2
Focus	Focussed on environment	0
	Less focussed on environment	1
	Not focussed on environment	2
Nostrils	Relaxed	0
	A bit more opened	1
	Obviously more opened, nostril flaring and possibly audible breathing	2
Corners mouth/lips	Relaxed	0
	Lifted slightly	1
	Obviously lifted	2
Muscle tone head	No fasciculations	0
	Mild fasciculations	1
	Obvious fasciculations	2
Flehming and/or yawning	Not seen	0
	Seen	2
Teeth grinding and/or moaning	Not heard	0
	Heard	2
Ears	Position: Orientation towards sound/clear response with both ears or ear closest to source	0
	Delayed/reduced response to sounds	1
	Position: backwards/no response to sounds	2
Total		.../18

Figure 7. Example of the scoring sheet used in the EQUUS-FAP pain scoring system. (Image source: [21]).

3.2 – Computer Vision

To cover sub question two to five, several techniques within the field of computer vision need to be discussed; For the questions, “*How can these features be extracted from the training images?*”, “*How can these features be turned into quantifiable data?*”, “*How can this data be compared and generalized to create a trained model of these features?*” and “*How can the accuracy of this application be validated and measured?*”; the focal points are feature detection, feature data conversion, data comparison and data validation respectively.

3.2.1 – Feature Detection

First of all, the relevant data needs to be filtered from the image. For this end, a large amount of options has been researched over the past and are openly available to use [8].

For the feature extraction it is important to keep in mind that apart from the actual extraction of relevant features, it is also important to assure that irrelevant data is discarded to remove as many false positive matches as possible.

3.2.1.1 – Background Subtraction & Object Recognition

As only a specific section of the image is used and, as mentioned above, it is important to avoid errors in later processes due to things like false positive matches in our data extraction, it is necessary to perform some form of background subtraction [24] and/or object recognition to filter out irrelevant data in our images.

As only the equine faces within our images are of importance, some form of object recognition like those described in the references [25] and [26] would be a fantastic way of distinguishing between foreground and background within the images. Once such a process has been completed, everything that is not included in the recognized object could be blacked out to perform a proper background subtraction on the image.

If the program is still working with given input data, this can also be somewhat cheated by creating pre-defined background masks/ground-truth images that contain a definition of what the foreground and background are. This would allow for the focus to remain on the rest of the algorithm early on and to keep the option open of returning to this kind of pre-processing at a later stage.

3.2.1.2 – Landmark Detection

As mentioned many times above, one of the main methods of assessing pain in animals is through facial feature detection, and more specifically so, through facial landmarks using Action Unit detection.

Facial AU recognition stands at the core of landmark detection and allows for the construction thereof. To recognize facial action units, techniques like dense-flow, feature-point tracking and edge extraction are combined to acquire information on sections of an image, which can be compared to templates describing the action units to be tracked [27].

Facial landmark detection is one of the core activities of this application, as all other analysis are based upon information read from the facial landmarks defined in horses. As it is desirable for the application to be able to run at a consumer-level eventually, preparations should be made to accommodate for input data that may not be in perfect conditions and subject to changes in pose/lighting and even under (minor) occlusions.

Taking these reasons into account and with the combined knowledge that there is a clearly determined set of facial AU/landmarks to look at, studies like Robust Cascade Pose Regression, a landmark position estimation and recognition tool that is made specifically to deal with images that include substantial amounts of occlusion, should be discussed [28].

RCPR works by adding a layer of occlusion data to its images. Using these during landmark estimation allows it to work around occlusions much better than other similar techniques. See Figure 8 for an example.

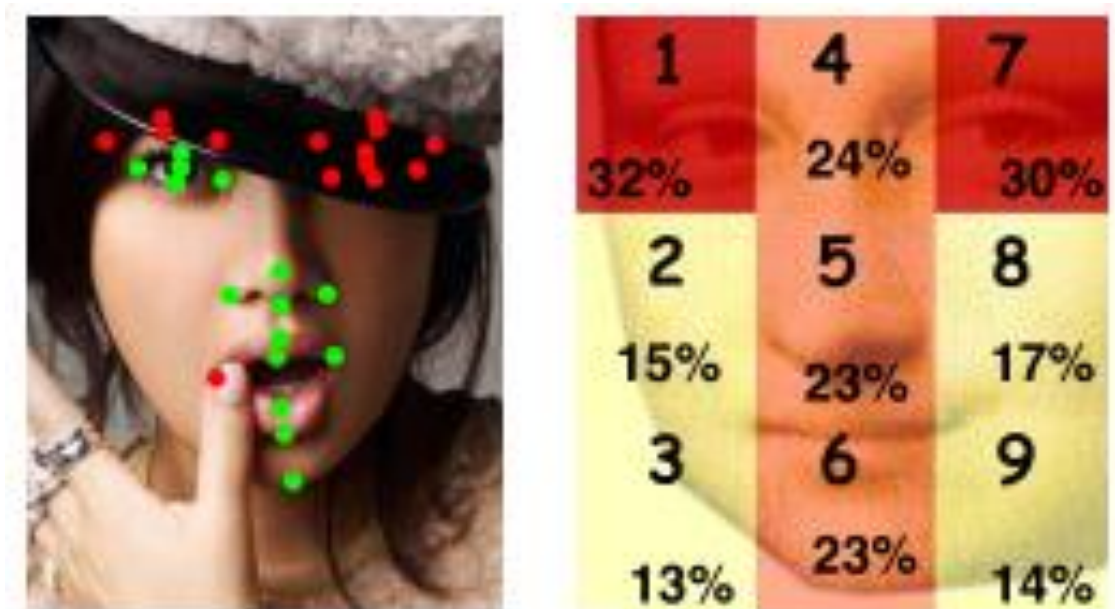


Figure 8. Left: annotated facial action units, including occluded ones. Right: example of the occlusion data of an image. (Image source: [28]).

3.2.1.3 – Active Shape Model

Expanding upon Landmark Detection even further leads to a technique called Active Shape Models (ASM) [5] [29] [30]. ASM is a technique aimed at automatically finding a set of landmark points that make up a general shape.

ASM works by defining a statistical model of the shape and variations therein of the object you are trying to recognise, for an example of such a set see Figure 9. By using a large set of manually annotated training images, you can calculate a baseline of the position of all features relevant within your set, accompanied by a variation range these features can be in (based on relative orientations and distances).

Once this ASM has been defined it can be run over (unknown) images to find and fit over the defined object. See Figure 10 for an example. The system first tries to calculate a rough starting position of the model using Genetic Algorithms [36], where it tries to place the template at random points within the image and then combines and reiterates using the best positions of matching image object and template boundaries whilst not violating the template constraints, to create a quick optimal initial guess. Once this initial guess is made the model is further iterated by comparing the model boundaries with boundaries/edges found within the image and trying to move each model point to close matching image points whilst not violating the model constraints, see Figure 11.

Due to an ASM being a full set of landmarks, with defined relative positions to each other, it is much more invariant to background objects containing possible “fake” landmark features. This is because it is trying to match multiple landmarks at once, using a large set of constraints.

As ASM includes both a baseline and variations, it is a technique that is very robust to occlusion, (minor) pose/angle alterations, rotation and variance within the objects themselves. These are all very desirable properties for this program, and as this technique also immediately yields the desired landmark regions, it is a very preferable technique to include in the application.

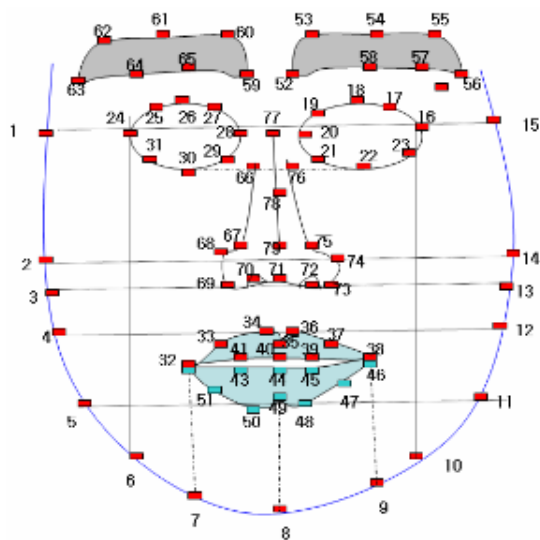


Figure 9. Example of a landmark scheme used to define an ASM. (Image source: [30]).

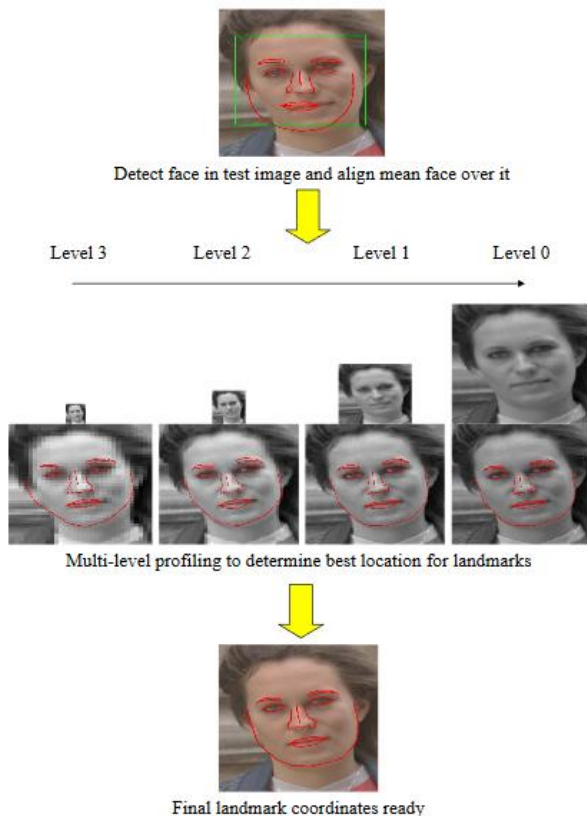


Figure 10. Steps involved during ASMs test phase. (Image source: [30]).

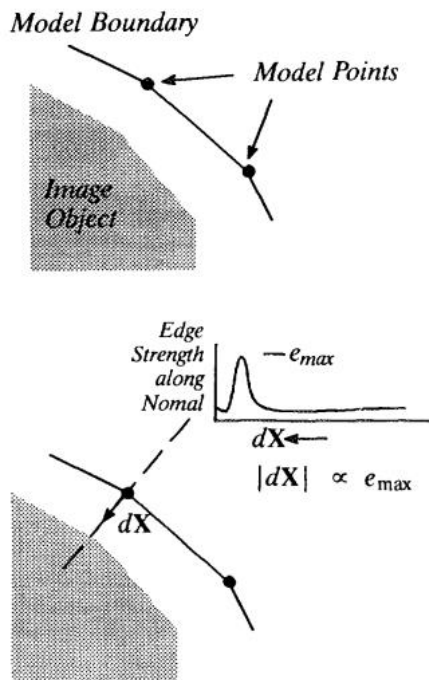


Figure 11. Example of a point on the edge of the model being moved towards a point on an edge (of an object) found within the image. (Image source: [5]).

3.2.2 – Feature Data Conversion

Once the facial AU and landmarks have been extracted, they need to be stored in data in such a way that they can be compared to other copies of the same AU/landmarks, so actual computations and comparisons can be performed. This way of storing data should once again be robust to as many possible changes like size and rotation, so that the data can be compared as easily and correctly as possible.

Some features can be stored and checked by very basic data elements like colour histograms and basic shape (density) description, but others may require much more complex forms of data as their descriptive complexity increases.

3.2.2.1 – Colour Spaces

A lot of information on features can be extracted by looking at the Colour Histograms [37] of images, as shadows, highlights and colour variations can tell a lot about the state of certain features and is very applicable in object recognition. To properly make use of colour histograms/data in general, it is important to make use of the right type of Colour Space [38], as this allows you to focus on or ignore specific parts of the colour data (see Figure 12).



Figure 12. Example of different Colour Models and Invariants. From top-left to bottom-right: RGB (original), Greyworld, RGB-Rank, $L^*a^*b^*$, HSL, Opposite Colourspace, $l1l2l3$, $c1c2c3$, YIQ, YCbCr. (Image source: [39]).

3.2.2.2 – Histogram of Gradients

Another robust way of analysing features is through Histograms of Oriented Gradients (HOGs) [31] [40]. HOGs are created by performing edge/flow detection on images, and then performing a template match on small subsets/boxes within the image, converting them in one of several predefined data units [35]. The groups of data units then store information of the orientation of the gradients within a section of the original image. Collecting this information of the landmark areas would assure that the data is properly comparable. HOGs have the added benefit that they are both invariant to size and lighting/colour, as the HOGs of smaller areas are summed into bigger areas (therefore always ending with same sized HOGs), and as they use edge/flow detection and normalization the original colour/lighting information is filtered out. For examples of this see Figure 13 and Figure 14.

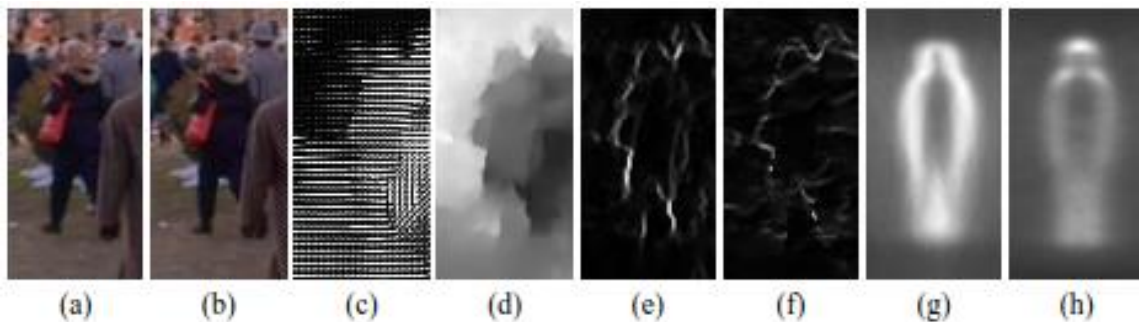


Figure 13. Different state of an image converted into a HOG. Most noticeable are the flow conversion seen in (c) and (d), and the gradient magnitudes seen in (e) and (f). (Image source: [31]).

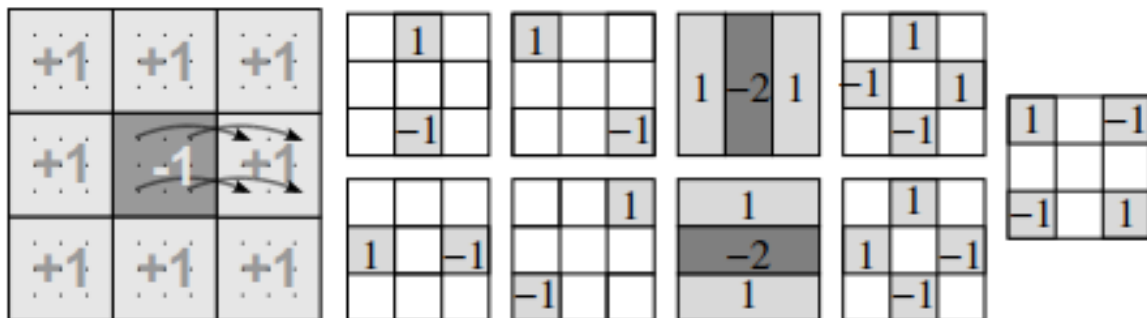


Figure 14. Example of the descriptors used to match and classify (pixel) regions within the image. (Image source: [31]).

3.2.2.3 – Image Thinning

For some features specific parts of a found area are of importance, like for example information about the contour or about the general shape of the area. One important but fairly advanced technique useful here is image thinning [32].

Image thinning can be done by running two parallel iterative algorithms; one that takes away pixels on the bottom-right boundaries and the top-left corners, while another takes away pixels from the top-left boundaries and the bottom-right corners. These two results are then combined to create one “skeleton” of the shape with unitary thickness and preserved end-point connectivity. See Figure 15 for an example.

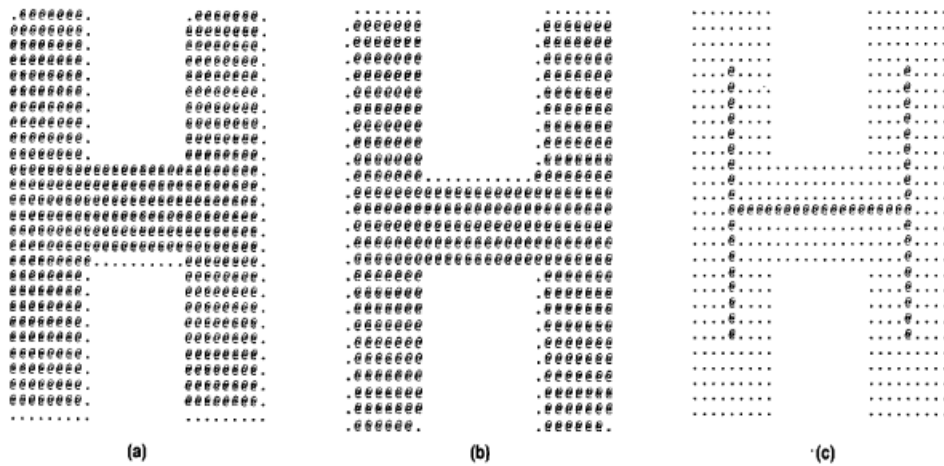


Figure 15. Image thinning algorithm; A: result of one step of the first algorithm. B: results of one step of the second algorithm. C: combined end result on the two algorithms running parallel to each other. (Image source: [32]).

3.2.3 – Data Comparison

Once the facial landmarks have been converted into measurable and comparable data, a way needs to be found to compare and classify this data as belonging to certain values. As there already is a defined and given range of pain-scores, all that really need to be done is to be able to link the found features to their respective (given) pain scores. If this can be done for all the features in all the images, it should be possible to create a properly trained model for each of the pain-scores.

3.2.3.1 – Support Vector Machine – SVM

One of the best methods to do this would be to use Support Vector Machines (SVM) [33] [34]. An SVM works by mapping large input datasets to a very high dimensional function and creating a plane through this highly dimensional field that separates the correct/desired set of input points from the incorrect ones. Using this plane, all new unknown points can then be tested by comparing on what side of the plane they would fall on based on its properties. If it falls on the side of the plane encapsulating a set of “correct” points, this new feature then counts as one of them and will be assigned that value. For an example of such a field see Figure 16.

Combining this with the application and training data, it should be possible to train an SVM to create fields describing, for all six pain-scoring features, the three different values they could have. It should then be possible to match the feature of any new image to these trained sets, and score it based on what plain it either falls in or is closest to.

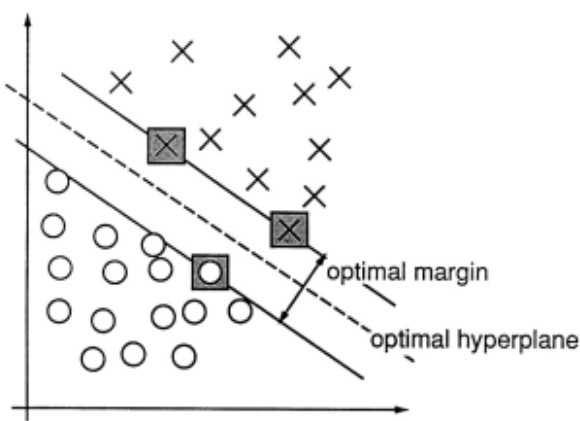


Figure 16. Example of the creation of a separating plane in 2D. (Image source: [33]).

3.2.4 – Data Validation

Once all of this is in place, all that is left to do is to evaluate the application. First of it is necessary to have an actual way of validating the algorithms based on accuracy. As our application results in training a model describing the pain-scores of each feature, an effective way of testing this would be by running some of these images (preferably by omitting them from the model first) against the model and comparing their actual known pain-scores with those returned by the model.

3.2.4.1 – Cross Validation

What has just been described above is basically a shortened version of the explanation of Cross Validation, which is therefore the preferred technique to use here [35].

For cross validation it is required that the network is first trained on a large subset of the data, keeping a small subset out of this training set. The pain scores of this small unknown subset are calculated against the application that has been trained with the larger one. The scores generated by the application can be compared with the actual known scores of this input data, allowing for the calculation of a general accuracy score of the application using the current sets of data. Then the separate small subset is swapped with another one from the larger training set and this is repeated until this has been done sufficient often (or using all possible combinations). Then generalizations and averages about this accuracy can be made across these tests to calculate a realistic accuracy score of the application.

3.3 – Related Work

Using combinations of the above-mentioned techniques facial expression, emotion and even pain-level recognition automation on humans has been researched and successfully done before. This has also been done for a handful of animals, and especially so in a project very similar to this study on the automated recognition of pain in sheep.

3.3.1 – Subtle Facial Expression Detection

As full emotional states are rarely expressed in daily life and most facial queues in humans are relatively small, development firstly started on recognizing subtle facial expressions that occur frequently in humans [15]. One way of doing this is using FACS. Using dense-flow extraction, facial feature tracking and edge and line detection considerable amounts of feature information can be extracted from facial images. Feeding these through a Markov model allows for the creation and categorization of FACS based on very small changes in facial expressions, allowing for the recognition of very subtle changes therein.

3.3.2 – Emotion Detection through Facial Expressions

Creating a more advanced version of the automatic recognitions of facial expressions through facial action coding systems, the ARFAS system was made, allowing for FACS based emotion detection to be improved even more and to be doable in real-time applications [16]. With this a proper application and use of the recognition of subtle facial features had been made.

3.3.3 – Pain Recognition through Facial Expressions

As self-reported pain validation is a difficult subject in pain assessment, the necessity/usefulness of an automated application becomes immediately apparent [17].

Expanding/adapting the techniques used on emotion detection, using a combination of machine learning techniques, like Support Vector Machines, in combination with FACS allows for the training of specific pain-feature recognition modules. Testing these modules on large datasets allows for an

application to expertly estimate pain-scores with an acceptable accuracy, and even allows for the differentiation of fake and real pain in humans [4].

3.3.4 – Automated Sheep Pain Recognition

As mentioned before there is one similar project to what this study attempts to achieve; an automated pain estimation program for sheep based on facial action unit detection [1].

This project combines a lot of the abovementioned ideas and techniques into an application which does exactly that which is required of this project, except this one is made for sheep instead of horses. They state in their work that their application should be generalizable to, amongst others, horses, and it is the aim of this project to be able to prove this statement. This work has been a big inspiration to this project, as they have already shown, tested and proved a lot of things that are relevant to this project as well.

Apart from being a major source of inspiration to this project, the proposed algorithm is also a very usable source for validating the algorithm of this project; as this study attempts to create a similar application (at least) comparable results/accuracy to this other project should be amongst the main goals. Comparing the eventual results of the algorithm this project makes for horses against the one made for sheep should therefore be a good benchmark to start from.

The sheep application works by starting with the basis of automated human emotion recognition, building upon the existing framework of these existing programs. The application takes an image as its input, on which it first runs a facial detection algorithm to find the face of the sheep within the image. Once the face is found it localizes/looks for facial landmarks, which it then uses in combination with a normalized version of the image to do facial feature extraction. These extracted features define the sheep's face and are compared with a pre-learned dataset of pain-scored sheep and their facial landmarks, through Histogram of Oriented Gradients comparisons, using Support Vector Machines. This results in similarity scores to other sheep images in the trained set, and these decide the calculated pain score of the sheep. For an example of this pain recognition pipeline see Figure 17.

The application makes use of many of the above-mentioned techniques as well, using CPR for the facial landmark detection, cross validation to validate their results and HOGS and SVMs to compare and analyse their results. One final thing their application does that has not been explicitly covered yet is feature-based normalization of the image. To help improve their information extraction the image is first normalized in a way that allows for specific feature data to be set, easing the comparison of data extracted thereof. For example, when normalizing the features, the face of the sheep is rotated in such a way that the eyes are always horizontally oriented and the nose vertically, which allows the information that is extracted to be rotation invariant.

Apart from some future work/improvements concerning making the program more robust and generalizable to other animals; they show significant results in the recognition and estimation of pain through facial expressions in sheep. Their application has an accuracy varying around 67% at self-validation and 49% under cross-validation, although this score is speculated to be higher if a larger dataset were available to train the program with, giving an overall acceptable result for the pain estimation both in general and in generalization thereof.

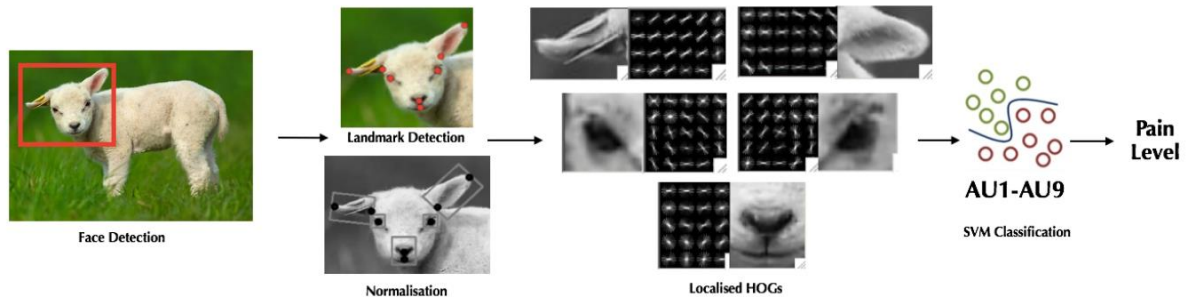


Figure 17. Example of the implemented sheep pain recognition pipeline, showing the landmark/feature detection, normalisation, conversion into HOGs and comparison through SVM. (Image source: [1]).

4 – Methodology

As mentioned earlier the following research question to describe the problems that are being tackled within this project have been set up;

“Can an application be created that, when trained using known professionally scored input data, can automatically determine the pain-level of a horse in an image, at the level of a (trained) veterinary?”

Which can be split up into the following five sub questions, where answering these should allow for the performing and conclusions of this research;

1. *“How can pain-scores be determined on our training data, and which features are most relevant for this objective?”*
2. *“How can these features be extracted from the training images?”*
3. *“How can these features be turned into quantifiable data?”*
4. *“How can this data be compared and generalized to create a trained model of these features?”*
5. *“How can the accuracy of this application be validated and measured?”*

To be able to answer these research questions this research can be split into three main tasks;

- Gathering and preparing training data.
 - Relevant in answering sub question 1.
- Creating an application to process and analyse this data.
 - Relevant in answering sub questions 2, 3, 4 and (partly) 5.
- Gathering ground-truth data and information on similar projects to validate and compare the results and quality of this application.
 - Relevant in answering the other part of sub question 5.

4.1 – Data Acquisition

The data required for this project consists of two parts; a set of facial photos of horses in different pain-states so that analyses can be made concerning the features of a (pained) horse face, and need official pain-scores of the horses as read from the pictures to be able to link features to respective scores.

As the primary goal of this application is to create a first version of the pain-scoring algorithm and not to make a robust generalization of one, stricter requirements can be put on our input data to make sure they are of sufficient quality to train this application on, and so checks for the validity of the method can be made more easily instead of having to take data validity into account as well.

Therefore the following set of requirements have been enforced upon the training data:

- Pictures should contain clear, preferably mono-coloured backgrounds that are easy to filter out.
- Pictures should be taken in two formats/facings, one front facing and one sideways (left) facing.
- The front facing pictures should have both eyes of the horse visible, and the head should be facing directly forwards/towards the camera.
- The sideways facing pictures should still have both ears visible and should also be made with the head looking straight forward.
- There should be pictures in the training set of horses with different skin colours and patterns, and there should be pictures of horses with widely varying pain scores/settings.
- For each picture taken (and thus for each facing), a separate professional/trained party from those that took the pictures should fill in an adapted EQUUS-FAP/HGS pain scoring sheet that is added to the picture.

Furthermore, as it is necessary that the application is usable on unknown data, a large enough training set to draw generalizable conclusions from is a necessity. The training data therefore has to both satisfy certain conditions and be of a large enough quantity if proper statements are to be made concerning the results of the application.

To collect this data and make sure the quantity of this data is enough, a separate parallel project has been run at the Utrecht University faculty of Equine Science. In this project a set of horses received clinically induced injuries causing them to become limb in one or multiple legs for a period of twenty-four hours. During this time, data was collected on several set time-intervals. The data that was collected included a set of pictures and official EQUUS-FAP/HGS pain scoring sheets filled in by trained professional veterinaries, created using the above-mentioned method. This data was made available to use specifically for this project. As the project ran using several horses and different settings, this should help satisfy the condition to have both a varied and vast enough subset of training data of our study.

The pain-scores calculated during this project follow a slightly different format from the regular EQUUS-FAP/HGS pain scoring sheet, as some values therein are based on (dynamic) visual changes over time or on audial clues. To encompass this, an adapted version of the scoring sheet has been created by (some of) the original creators of the EQUUS-FAP scoring system, that combines all the static features of the two systems into a single list of six features, of which an example can be seen in Figure 18.

The fact that still images are used also causes another problem, which is that not all the features of the EQUUS-FAP/HGS scoring system can be seen from any one angle. This is why photos from two different facings are used, frontal and sideways (for clarification on which features can be seen from which angles, see Figure 18).

Facial pain score of a photo

Eyelids – Sideways Facing

- Opened, sclera (ooglid) can be seen in case of eye/head movement. 0
- sclera (oogwit) can be seen 1
- More/obvious opened eyes or tightening of eyelids. An edge of the sclera can be seen by 50% of the time. 2

Nostrils – Front Facing

- Relaxed 0
- A bit more opened 1
- Obviously more opened, nostril flaring and possibly audible breathing. 2

Corners mouth/lips – Sideways Facing

- Relaxed 0
- Lifted slightly 1
- Obviously lifted 2

Stiffly backward ears – Front Facing

The ears are held stiffly and turned backwards. As a result, the space between the ears may appear wider relative to baseline.

- Not present 0 oren tussen naar voren en halverwege gericht
- Moderately present / one ear backwards 1 minstens 1 oor halverwege of verder naar achter
- Obviously present 2 beide oren strak naar achteren

Tension above the eye – Front Facing

The contraction of the muscles in the area above the eye causes the increased visibility of the underlying bone surfaces. If temporal crest bone is clearly visible the score should be coded as 'obviously present'.

- Not present 0
- Moderately present 1 matige frons boven oog
- Obviously present 2 duidelijke frons boven oog

Prominent strained chewing muscles – Sideways Facing

Straining chewing muscles are clearly visible as an increased tension above and behind the mouth. If chewing muscles are clearly prominent and recognizable the score should be coded as 'obviously present'.

- Not present 0
- Moderately present 1
- Obviously present 2

Total ----- .../12

Van Loon, J.P.A.M. and van Dierendonck, M., 2015. *Monitoring acute equine visceral pain with the Equine Utrecht University Scale for Composite Pain Assessment (EQUUS-COMPASS) and the Equine Utrecht University Scale for Facial Assessment of Pain (EQUUS-FAP): A scale-construction study.* Veterinary Journal, 216: 175-177.

Dalla Costa, E., Stucke, D., Canali, E., Minero, M., Leach, M.C. and Lebelt, D., 2014. *Development of the Horse Grimace Scale (HGS) as a pain assessment tool in horses undergoing routine castration.* PLoS One 19;3(3).

Figure 18. Example of the adapted EQUUS-FAP pain-scoring sheet created for this application. Contains underlined Dutch notes on what to look for in certain features.

4.2 – The Application

The main job for this project consisted of creating the actual application that processes, analyses and compares all acquired data.

As this application almost fully consists of computer vision techniques, an immediately (almost mandatory) requirement is to use OpenCV in the application. OpenCV is an open source computer vision library which contains coded examples and libraries of nearly all documented computer vision techniques, or at least contains some form of supporting code for them. As OpenCV is an originally C++ based library, choosing this as the base programming language removes as much unnecessary work and incompatibility issues as possible.

Due to the experimental nature of this application (as it is a research implementation of a new idea), the application does not need to contain much more functionality outside of the computer vision code than the ability to load in the test-data (and display it for debugging/validation) and compare its results. This results in the need for a bare-bones C++ application (as making something like an actual user-friendly or commercial interface is far beyond this projects scope). For this an IDE like Visual Studio would be perfect, as this support the creation of Windows Console Applications, which remove all but the minimal functionality from an application allowing it to run as efficiently and with as little overhead as possible. Also, it is very compatible with the OpenCV library.

If at a later point in time more user-friendly or consumer-level versions of this application would be desirable this is also a very good setup. This is because this setup requires the program to basically be written in general, non-IDE-specific, C++ code only, which can then easily be built into another application/IDE with the only real requirement being that this application needs to support OpenCV.

4.2.1 – The Algorithms

Functionality-wise the application had to be able to perform the following tasks;

- Images and respective pain-scores need to be loaded into the application.
- Images need to be prepared for processing and need to be processed on.
 - Irrelevant information (such as background objects) should be filtered from the images.
 - Relevant information needs to be found within the image (the face and its specific features).
 - These features need to be converted into usable data.
- Extracted data needs to be compared and validated.
 - Actual training must be done to connect specific features to respective values.
 - Actual testing must be done using data sets not trained upon by the application.

For the methodology of specific parts of the application, or at least a first version of it, inspiration was drawn from the method described in “Estimating Sheep Pain Level Using Facial Action Unit Detection” [1], and some of the implemented methods in this application have followed a similar approach/setup.

With all of these as the setup, the following step-by-step implementation would be an effective way to set up the application:

- Training images are loaded into the algorithm, together with their respective pain-scores.
- Front and side images of the same pain-moment are combined into a single pain-moment entry and score.
- Background subtraction is done on the training images to filter out irrelevant data.

- Landmark detection is done on the image to separate it into its relevant features. This is done using Active Shape Models, for which training data has to be made from each image.
- These features are then converted into comparable data using several computer vision techniques, like colour space conversions, histograms of gradients...
- These converted forms of data are, combined with their respective features pain-score, fed into a support vector machine to create a trained model of the features pain-scores.

Due to there being both front and sideways facing images (for the reason as to why this is see 4.1 – Data Acquisition), the application has to define and train two ASM, one for the sideways facing horse face and one for the front facing horse face. Outside of this no other steps should be altered using two different facings of images, as in the end each feature was processed for only one angled version at a time.

To build and use these techniques the easiest and best implementation would be to use the OpenCV library as mentioned above, which comes with lots of prebuild algorithms and computer vision techniques, many of which useful for the current implementation and expansion of the algorithms used in this application.

To compute the ASM the UoMASM library has been used. This because it is a C++ based library containing implementations and tools to easily make and use ASMs. Therefore, it is very compatible with the proposed application so far. On top of that, this library was made by some of the original creators of the ASM technique, which has resulted in this being one of the most up-to-date and well-built implementations of the technique.

4.3 – Project Validation

As the purpose of this project is the exploration of the possibility of the proposed application, it is very important that validation steps are performed on every part of the application, so as to assure that all ideas do work and keep tabs on all eventual weak-spots within the application.

Things that need to be validated in the abovementioned proposed implementation would be;

- The filtering of background information.
 - Is it even necessary in the first place?
 - Does it filter out too much/little?
 - What is the best way of doing this?
- The quality of the ASM.
 - What is its general accuracy and is this good enough (comparing it to similar implementations)?
 - Are there better alternatives (comparing it to other implementations of the same purpose)?
- The quality of the data conversion.
 - Is the application losing any relevant data?
 - Are there better alternatives (comparing it to other implementations of the same purpose)?
- The quality of the SVN/data labelling.
 - What is its general accuracy and is this good enough (comparing it to similar implementations)?
 - Are the features checked in right/best ways (compare it to the methodology of similar projects)?
- How well it deals with unknown data overall.

- Comparing results of untrained data analysis with those of trained data.

For comparing the accuracy of the ASM the accuracy of the same ASM implementation of human faces could be taken into account, like those reported in [5], [29] and [30]. For the accuracy of the scores calculated through the SVN their known (ground-truth) values can be used, as these are very concrete values to test against.

Testing for general accuracy differences on trained and untrained data can be done through Cross Validation; the application is run multiple times, each time omitting a certain subset of data from the training set, allowing for a unique set of untrained but validate-able images to be created. The application can then be trained with the training set of all the other images. Once this is done the application can then be run using the omitted set to see how it would score these pictures, which can then be compared to their actual scores to give a general accuracy value of the program with the current set of training data.

Through repeated tests with different subsets and types of subsets of untrained test data, calculating a proper overall accuracy and validity of the program should be possible.

5 – Implementation

This section will cover the actual implementation of the application, the algorithms therein, how the data was handled (and processed) and how the application (and the steps therein) were validated. The choices made within this implementation will also be discussed, as well as which alternatives have been considered.

5.1 – The Application

After much prototyping and testing the final implementation has been chosen to be made in C++, making use of the Visual Studio IDE's Windows Console Application template, and using the OpenCV (version 2.4) and UoMASM (version 1.1) libraries.

For each of these options some considerable alternative was available, which will all briefly be discussed to explain why this application was made the way it was.

First of the programming language to be used consisted of two main competitors; C++ and C#. The reason for using either of these two is because it was important to use an imperative language that has a large existing support for use in computer vision research. Both C# and C++ contain implementations for these (same) kinds of libraries, however C# tends to be a bit better when creating large functional applications, where C++ tends to be a bit more efficient when creating more bare-bones algorithm-intensive applications like the one that was implemented during this study. Furthermore, as C++ has a bit more use in the research field of computer vision, it is more easily compatible with other existing and potentially future useful libraries which is why C++ was chosen in the end.

The next and biggest consideration for setting up the program was the main library/tool to use to be used to cover the (basic) computer vision techniques needed to make this application work. There are two main competitors that come to mind here when working in the C++ language; [OpenCV](#) and [VXL](#). Although VXL is the more research focused one of the two, consisting of some very advanced functionality and specific kinds of support, OpenCV has a much broader userbase. Furthermore,

OpenCV has a focus on the more often used/simple tasks, which means that it is mostly focused on having those tasks implemented in an as optimized way as possible. As the application itself has to make use of a lot of very basic techniques, having more optimized versions thereof is much more valuable than having support on functionality that is not explicitly used, which is why OpenCV was chosen over VXL.

The IDE that was chosen to work in was Visual Studio, making use of the basic C++ Windows Console Application template. The reason for choosing this template is because it is the most bare-bones one that still allows for some form of user control (console input/output), whilst not wasting any computing on added unrequired functionality (as due to the research-oriented goal of this project much more than basic input-output is not (yet) required of the application). Some alternatives over Visual Studio that were considered and could again be considered for future, expanded versions would be Eclipse or Android Studio, as both also support C++ applications and have existing computer vision (OpenCV) based libraries to import and allow the ability for creating multi-platform applications of this input. However, as once again the focus of this study lies on tests and research, something like a consumer-level application is not yet necessary. Therefore, having more than the minimum required support only takes away from the application, meaning that sticking to the most basic and generally compatible tools is the best option for now.

To be able to work with and find the specific features on the faces of the horses, it was necessary to have a recognition tool that was able to find the horse-faces in the first place. The “easiest” way of doing so would be by doing basic object recognition in combination with feature descriptions of what a horse-face object would look like. However, this is a task that is very difficult to perform due to the complexity of the searched object. Another alternative would be using shape models, either in 2D [5] or 3D [9], to create a statistical average horse-face, and then doing a recognition check on the image. Using a 3D model would most likely give the most realistic, most usable and the best results, however this does require a large dataset of 3D horse faces to train on and requires quite the intensive training to be done on this dataset, basically resulting in a project with a scope as big as this one. Using a 2D model might result in a bit lower quality project with a bit more variety, but it is something that is doable within the scope of this project (especially when making use of existing libraries). On top of that the large data-set of 2D images is already given, which is of sufficient quality and quantity to be used to train such a model on, allowing for the project to require very little extra data.

This resulted in the project making use of an Active Shape Model for finding the faces of the horses, for which a few existing tools/libraries are available. The two main options would be [UoMASM](#) [30] [41] and [STASM](#) [29], both alterations/improvements expanding upon the base idea of ASM [5], which are set up in much the same way (both C++ based, STASM being built using OpenCV functionality and UoMASM using VXL). Even though STASM is, like this application, built upon the OpenCV library and therefore more compatible with the rest of the code, UoMASM seems to yield more desirable results for this application and it is much more compatible with customized/unique faces. It also has been updated more (UoMASM is still actively being updated while STASMs latest build is from 2013). For these reasons the UoMASM library has been used within the application.

5.2 – The Algorithms

The application itself runs using a main control loop consisting of the following step (these will be clarified in corresponding sub-sections as well);

1. Load in all image data found in specific folders before going into the loop.
2. Allow the user to load in statistics on the pain-scores belonging to these images.

3. Allow the user to run background subtraction, ASM feature selection and feature conversion/comparison to convert images into corresponding feature data.
4. Allow the user to train an SVM on this feature data.
5. Allow the user to check the accuracy and validity of this SVM by testing known and unknown subsets of images against it.
6. Allow the user to perform the aforementioned tasks on specific subsets of images (sorted on their facings).
7. Save intermediate and end results to drive to allow faster processing when rerunning certain settings and to allow proper visualization of results.

5.2.1 – Loading (Image) Data

The application starts of by loading all images presented in its working folder. To process this data, a data convention has been made to assure it is easily accessible and always in a known format. To do this each pain-moment of each horse a single instant has been labelled. For each instant the front- and sideways facing pictures are then saved under this label name as “.JPG” type images, and the accompanying pain-score values as minimalistic “.txt” files. For ease of reading of the training data, all front- and sideways facing pictures have been oriented to be facing towards the left (if applicable), so the application always knows in exactly what format it receives its training data.

Once read, the application saves these image sets/facings so that later steps of the application can easily know what to work with and no further work is required looking for images. For an example of the information is stored in this way see Figure 19.

```

D:\Work\School\ThesisProject\Project\EQUUS-ARFAP\EQUUS-ARFAP\x64\Debug\EQUUS-ARFAP.exe
-----
Reading image data...
-----
Amount of photos processed:
-----
Pain moments labels found: 90
Total Amount of Photos found: 161
-----
Front facing photos: 72
    full: 33
    tilted: 39
Side facing photos: 89
-----
Usable/Complete photo sets: 71
-----
Loading data finished, press any key to continue.
-----

```

Figure 19. Example of data read from the image folder.

5.2.2 – Loading Data Statistics

As mentioned above each image is loaded in with a respective pain-scoring file containing all its facings relevant pain-scores. As these are the scores used to train the application with, the way these scores are spread could have an influence on the eventual results of judging them (for example a features score that never occurs within the training data should also never be able to occur in our results, and one that appears very few times would be less well trained on than one that does appear more often), a generic analysis has been built in to be able to see this score spread (see Figure 20).

```
D:\Work\School\ThesisProject\Project\EQUUS-ARFAP\EQUUS-ARFAP\x64\Debug\EQUUS-ARFAP.exe
-----
Amount of features processed.
-----
Processed photos pain score features:
-----
Eye Shape Side score occurrences:
value 0: 40
value 1: 47
value 2: 2
-----
Nostrils Front score occurrences:
value 0: 21
value 1: 32
value 2: 19
-----
Nostrils Side score occurrences:
value 0: 64
value 1: 18
value 2: 7
-----
Mouth Side score occurrences:
value 0: 49
value 1: 40
value 2: 0
-----
Ears Front score occurrences:
value 0: 33
value 1: 24
value 2: 15
-----
Ears Side score occurrences:
value 0: 42
value 1: 21
value 2: 26
-----
Eye Sclera Side score occurrences:
value 0: 68
value 1: 16
value 2: 5
-----
```

Figure 20. Example of de pain score data read from the images from the image folder.

5.2.3 – Main Tasks

The main tasks of the application are those captured within its control loop body (see Figure 21), which are those performing background subtraction, feature recognition/detection, feature-data conversion and the training and testing of the SVMs.

```
D:\Work\School\ThesisProject\Project\EQUUS-ARFAP\EQUUS-ARFAP\x64\Debug\EQUUS-ARFAP.exe
-----
Main application loop entered.
-----
Type any of the following commands to continue.
-----
painscores:  View (data on) the pain scores found by the application.
photos:      View data on the images found by the application.
analyze:     Process and analyze the input data found by the application.
test:        Train and test the application using the analyzed(!) data.
exit:        Close the application.
-----
analyze
-----
Settings:
-----
Show each image manual or automatic? ('manual' or 'automatic'):
-----
automatic
-----
Which photos should be shown? ('pairs', 'front' or 'side'):
-----
side
-----
Force re-initialize the files? (yes/no):
-----
no
-----
```

Figure 21. Example of the applications main control loop and the options therein.

5.2.3.1 – Background Subtraction

The first step of the main algorithm is to perform background subtraction on the images to make sure there is no interference in later steps due to background objects/data. To do this a set of ground-truth background masks has been created that show where the face of the horse is for each image. For an example of this process see Figure 22.

These masks are created in such a way that firstly a background subtraction implementation that is guaranteed to be correct is created, so it can guarantee that this is of no influence on later steps of the algorithms. And secondly this allows the program to, for now, keep the focus on the feature recognition and conversion steps of the algorithm. Thirdly, when an actual background subtraction algorithm is built in a future version of the application, for example when it becomes necessary for it to work with completely unknown/new images, these can be used to test for the accuracy, as these masks can also be used in ground-truth validation comparison.

Another important note here is that, as the focus of project is more so the feature recognition and conversion, this ground-truth method can be used as a first implementation to test whether background subtraction even has an influence on the accuracy at all, before spending (lots of) time into creating an algorithm that might not even impact the end results.

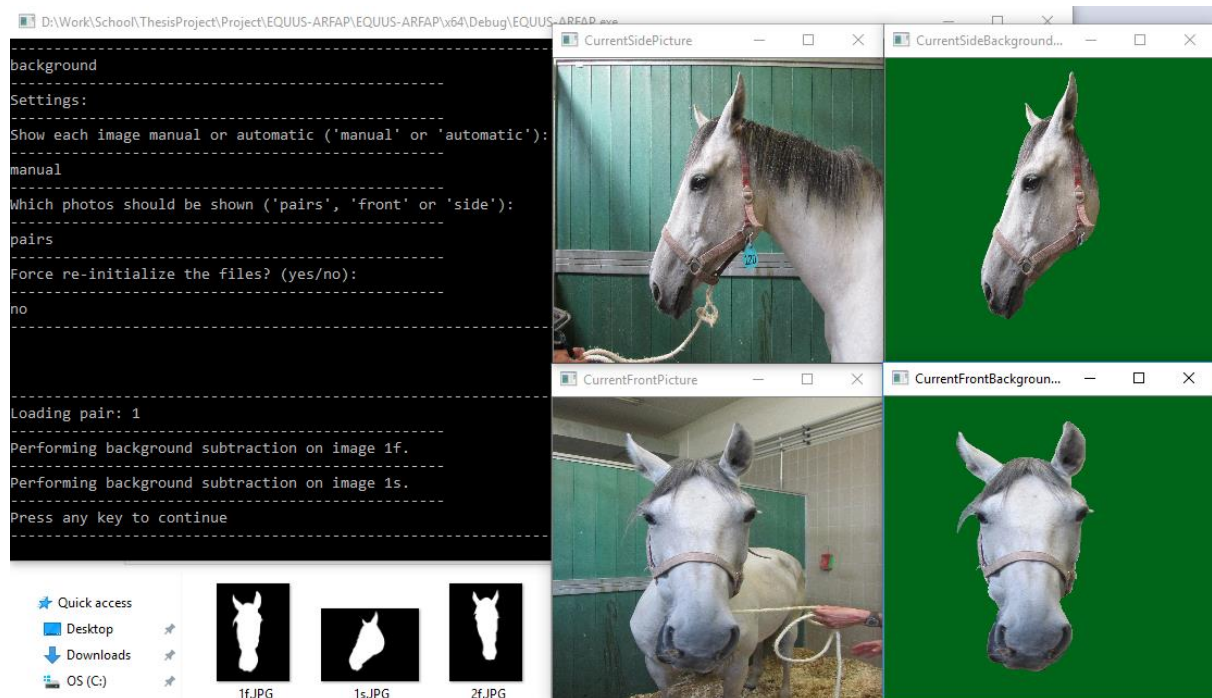


Figure 22. Example of background subtraction using custom made background masks.

5.2.3.2 – Feature Recognition

To perform feature recognition, the UoMASM library was used to implement a set of ASM describing all three facings of the images (for why there are three facings see section 5.2.5 – Data Subsets).

To set up these ASM for each facing a mask has been defined, consisting of points describing their points of interest and the general connecting shape, as well as a set of curve files connecting these points to each other (see Figure 23, Figure 24 and Figure 25). Once the files defining these masks were set up, ground-truth manual implementations were made for all images, filling in the positions of each of their points and saving these in “.txt” files. These files can then be used to both train the

ASMs that the application uses and to validate results given by them, by comparing generated point locations with the ground-truth ones.

Using these calculated masks, these positions can be read into the application and combined with the curve descriptions of which (connected) points are of interest, sub-images can be created with bounding boxes placed around them to segment all of the features of interest (see Figure 26).

Sadly, this project did not have enough time to actually go into training the ASMs, as this turned out to be a much bigger task than expected, and for now they are built in using their ground-truth version as well. A foundation has been built for doing so and all code has been made compatible with the application, but this is for now left open for future expansions and/or implementations.



Figure 23. Frontal facing equine face with marked points of interest, additional boundary points and their connecting curves.

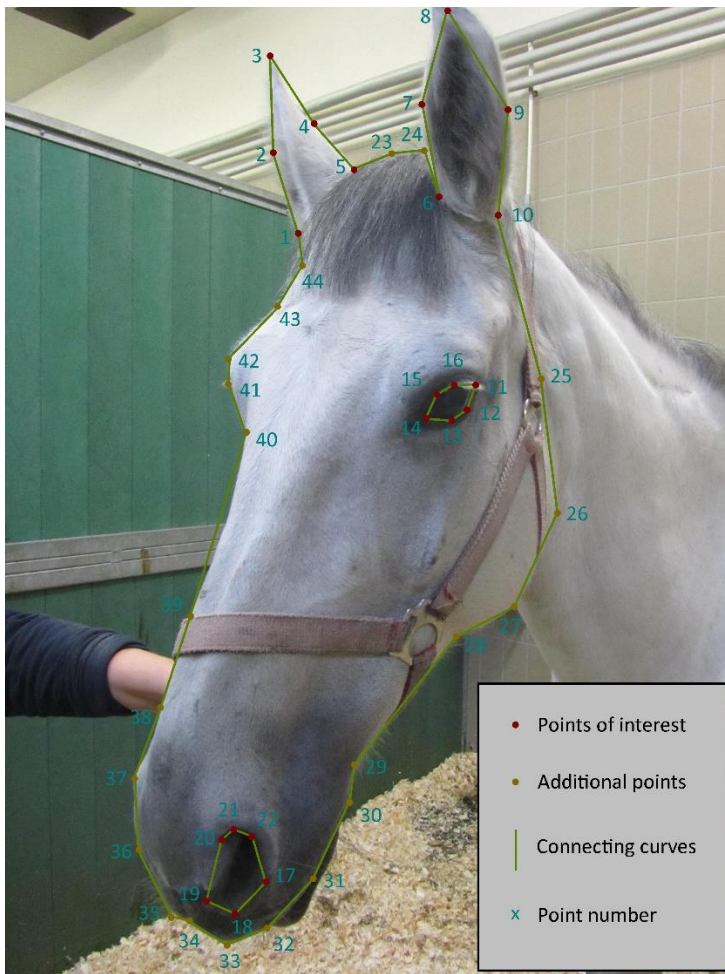


Figure 24. Frontal tilted facing equine face with marked points of interest, additional boundary points and their connecting curves.



Figure 25. Sideways facing equine face with marked points of interest, additional boundary points and their connecting curves.

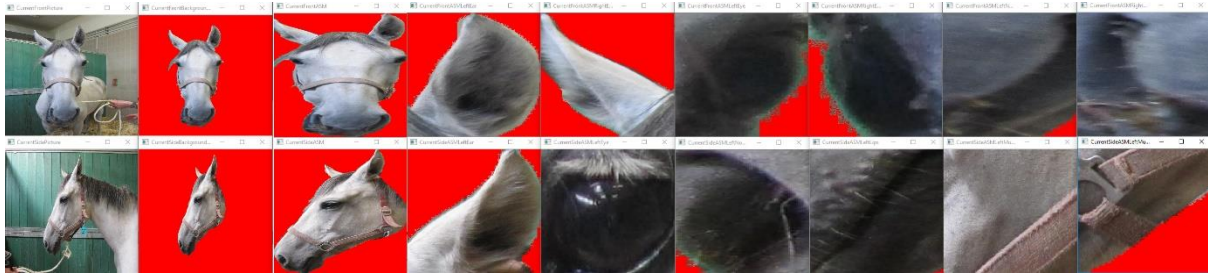


Figure 26. Example of the features of interest read from the ASM masks.

5.2.3.3 – Feature-Data Conversion

To allow for the data to be labelled, methods had to be created to convert the features into concrete (and comparable) forms of data. After initial data extraction and discussions on what points of interest would work for this kind of data conversion, in combination with the pain feature definitions given by the pain-scoring sheet (see Figure 18), the following five features have been chosen and defined for doing so. For each of these features it will be covered how it was decided to convert them into sensible and comparable data;

1 – Sclera (eye white), see Figure 27:

Visible angles	Sideways – Yes. Frontal – No.
Pain-score ranges	0 – No visible sclera, eyes normally opened. 1 – Sclera can be seen, eyes normally opened. 2 – Full edge of sclera visible, eyes more open than normal.
Defining feature	Sclera visibility.
Concrete definition	Presence of white shades/colour within the eyeball area.
Processing steps	1 – Convert image into a colour scale that is invariant to (high)light effects, for example the HSV colour space, where illumination can be detected and filtered out using thresholds. 2 – Use a shape mask to filter out irrelevant areas of the eye where no sclera can be detected. 3 – Turn the image into a colour histogram to know what colours (and therefore the sclera) occur in what percentages of the total space of the eye.

2 – Eye shape (frown), see Figure 28:

Visible angles	Sideways – Yes. Frontal – No.
Pain-score ranges	0 – No visible “frown”, shape of eye and eyebrow area is very rounded. 1 – Somewhat visible “frown”, shape of eye and eyebrow area contains straighter lines with corners more so than being rounded. 2 – Visible “frown”, shape of eye and eyebrow area contains a very hard corner (nearing a 90-degree angle).
Defining feature	Curvature of top half of eye and eyebrow area.
Concrete definition	Angles within area immediately above the eye.
Processing steps	1 – Detect the dark areas within the area found through adaptive mean thresholding to get the significant area of the eye (the area containing the top curve to be looked at). 2 – Smoothen and perform closing on this shape to make sure the top curve is properly connected.

	<p>3 – Use a variation of a thinning algorithm to filter out the most upwards facing boundary of the image and the most leftwards facing boundary of the image, which can then be combined to calculate the top-left boundary pixels/corners.</p> <p>4 – By directly comparing the ratio of the size of the top-left boundary with the top and left ones, the lines level of curvature can be measured.</p>
--	---

3 – Nostrils (shape), see Figure 29:

Visible angles	<p>Sideways – Yes.</p> <p>Frontal – Yes.</p>
Pain-score ranges	<p>0 – Nostril half open, no muscular tension.</p> <p>1 – Nostril more open, some muscular tension.</p> <p>2 – Nostrils fully open/flaring, strong muscular tension.</p>
Defining feature	Openness of nostril.
Concrete definition	Roundness of the nostrils bounding area/curvature.
Processing steps	<p>1 – Detect the dark areas within the area found through adaptive mean thresholding to get the significant area of the nostrils (the darker area describing the nostril).</p> <p>2 – Smoothen and perform closing on this shape to make sure it's properly filled.</p> <p>3 – Calculate the average convexity defect of the shape as a method of calculating in combination with the dimensions of the found area to calculate openness and roundness values.</p>

4 – Ears (position), see Figure 30:

Visible angles	<p>Sideways – Yes.</p> <p>Frontal – Yes.</p>
Pain-score ranges	<p>0 – Ears facing forward, distance between ears normal.</p> <p>1 – At least one ear facing (more than halfway) backwards, distance between ears may appear wider.</p> <p>2 – Both ears pointing (stiffly) backwards, distance between ears may appear wider.</p>
Defining feature	Angle of ears.
Concrete definition	Visible shape of the ears.
Processing steps	<p>1 – Perform edge detection based on the strength of a Histogram of Oriented Gradients on the image of the ears to extract all significant edges and use thresholding to find the strongest curve(s) (the edge/shape of the ear as a whole).</p> <p>2 – Use binning to divide the ear image into a frontwards, middle, and backwards facing bin of pixels.</p> <p>3 – In every bin calculate the area where both the top and bottom line of the ear overlap, giving a sign of direction of the ear (as the direction the ear is pointing in will have high overlap as this is where the tip of the ear will be, as in a neutral pose the 2 bases of the ear will be on the frontwards and backwards facing bins, whilst one of them tends to fall in the middle bin when the ear is facing any direction).</p>

5 – Mouth (lips), see Figure 31:

Visible angles	<p>Sideways – Yes.</p> <p>Frontal – No.</p>
Pain-score ranges	<p>0 – Mouth shape more curved, no muscular tension.</p> <p>1 – Mouth shape straighter, some muscular tension.</p>

	2 – Mouth shape is a straight line, strong muscular tension.
Defining feature	Curvature of the mouth.
Concrete definition	Angles within area where the lips make contact.
Processing steps	<p>1 – Detect the dark areas within the area found through adaptive mean thresholding to get the significant area of the nostrils (the darker area describing the nostril).</p> <p>2 – Thin the curve out to 1-pixel width using image thinning algorithms, effectively converting it into a set of coordinates.</p> <p>3 – Use an M-estimator to find the best-fitting line running through the found curve</p> <p>4 – For each pixel on the thinned curve, calculate its distance to the best-fitted line, and average these values to gain a measure of stiffness/straightness of the found curve.</p>

As noted in the paper introducing HOGs [40], performing Gamma Correction could have some positive influence on the quality of the HOG, which is why this was also implemented as an extra pre-processing step when calculating these.

The sixth original feature, Strained Chewing Muscles, ended up not being used in the final version of the application and was therefore not included in this section. The reason(s) therefore can be found in section 7.1.1 – Selected Features.



Figure 27. Sclera (eye white) detection by converting the eye feature into the HSV colour space, filtering out irrelevant areas with a shape mask and placing specific thresholds on each channel, leaving a selection of recognized eye white. Both an image with high and low (no) percentage of eye white are shown.

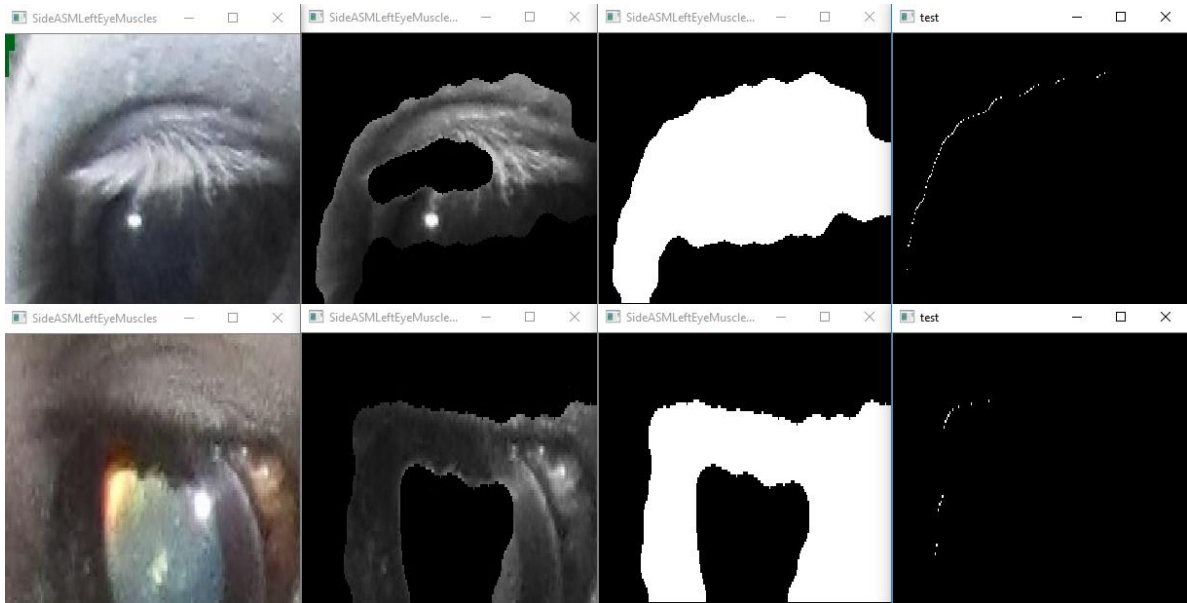


Figure 28. Eye shape detection by using adaptive mean thresholding to find the area containing the shaded curve above the eye and calculating the top-left boundary thereof. Both a curved and cornered eye are shown.

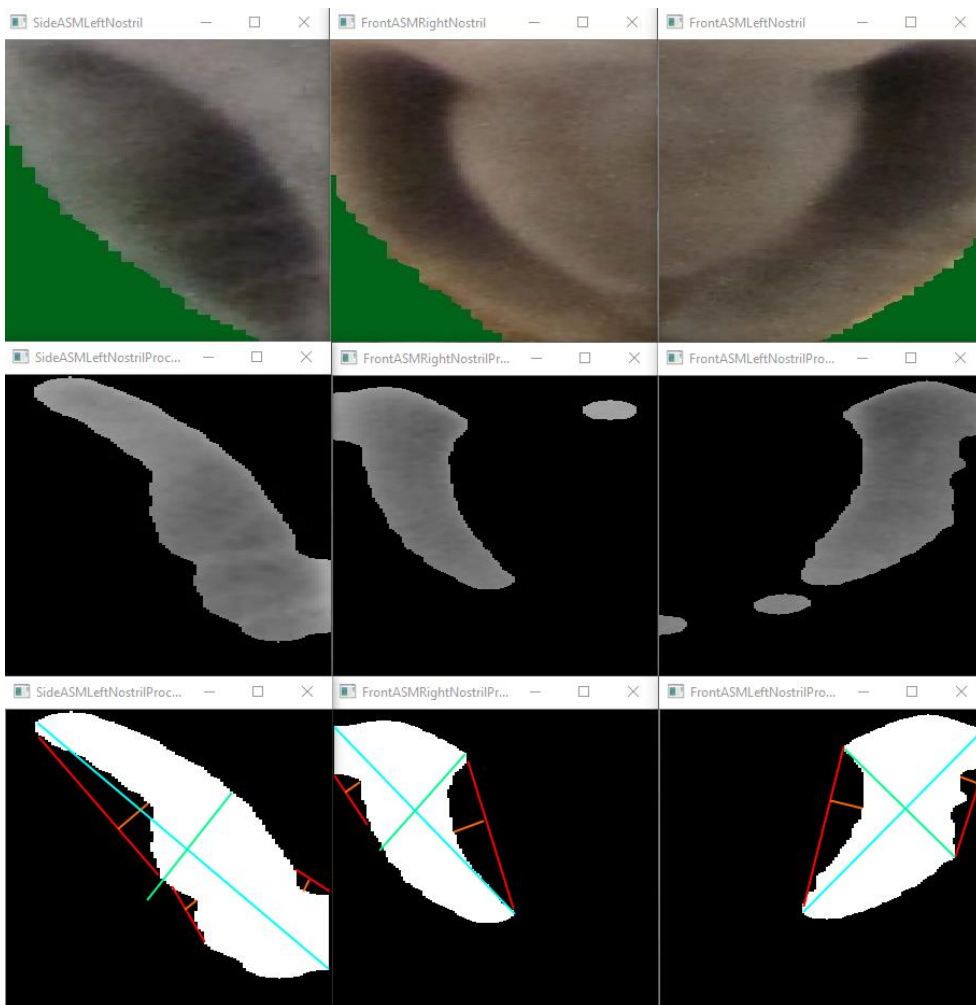


Figure 29. Nostril detection by using adaptive mean thresholding to find the darker areas of the nostril area, smoothing and selection to filter out the nostril and a hand-drawn visualization of the convexity defects shown in orange lines perpendicular to the red lines tracing the “imaginary” concave contour, together with a height and with ratio shown in respectively blue and green lines. Both front and sideways angles are shown.

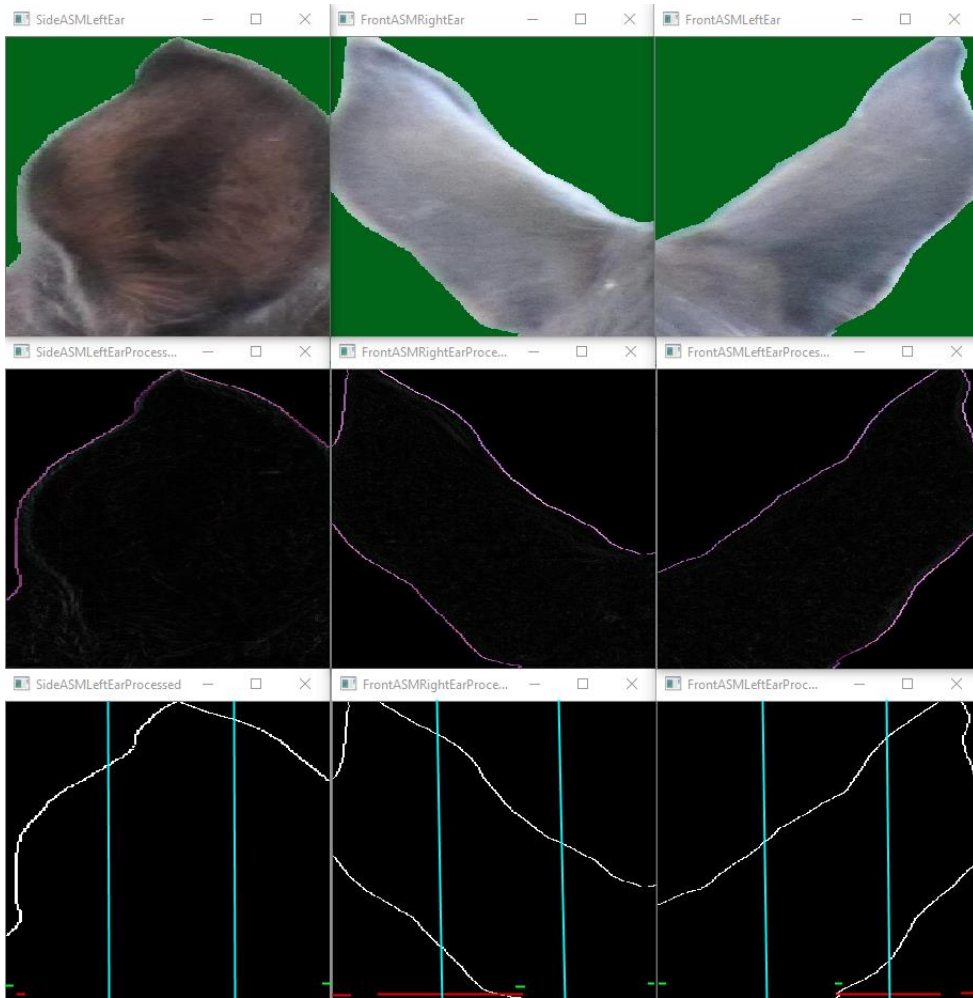


Figure 30. Ear outline detection by using HOG based methods to find the strongest curve(s) through the area of the ears and a hand-drawn visualization showing in blue the lines separating the three bins in each image, in red the horizontal area in which lines overlap, and in green the position of the bases of the ears. Both front and sideways angles are shown.

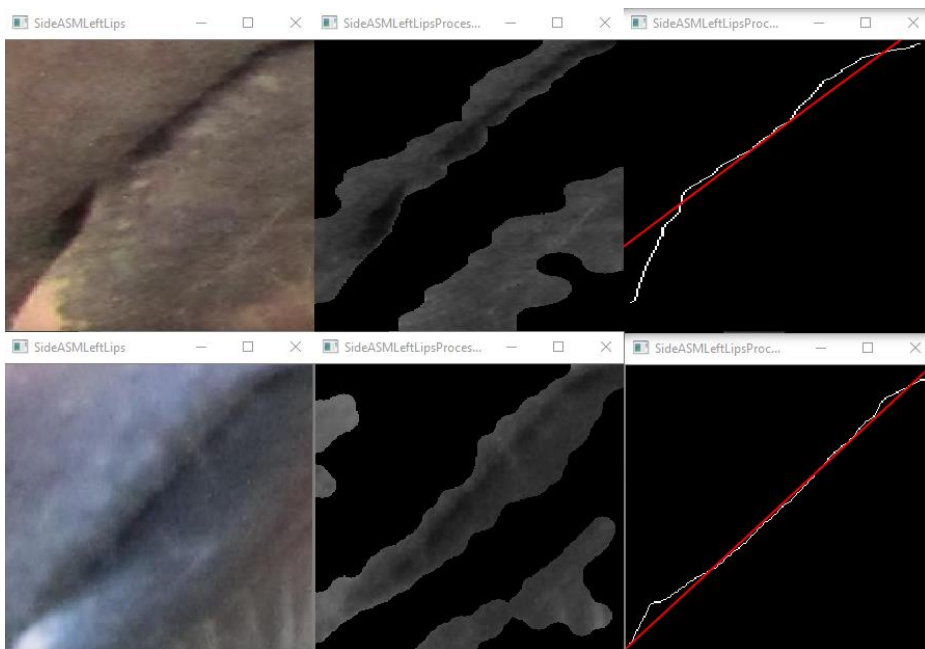
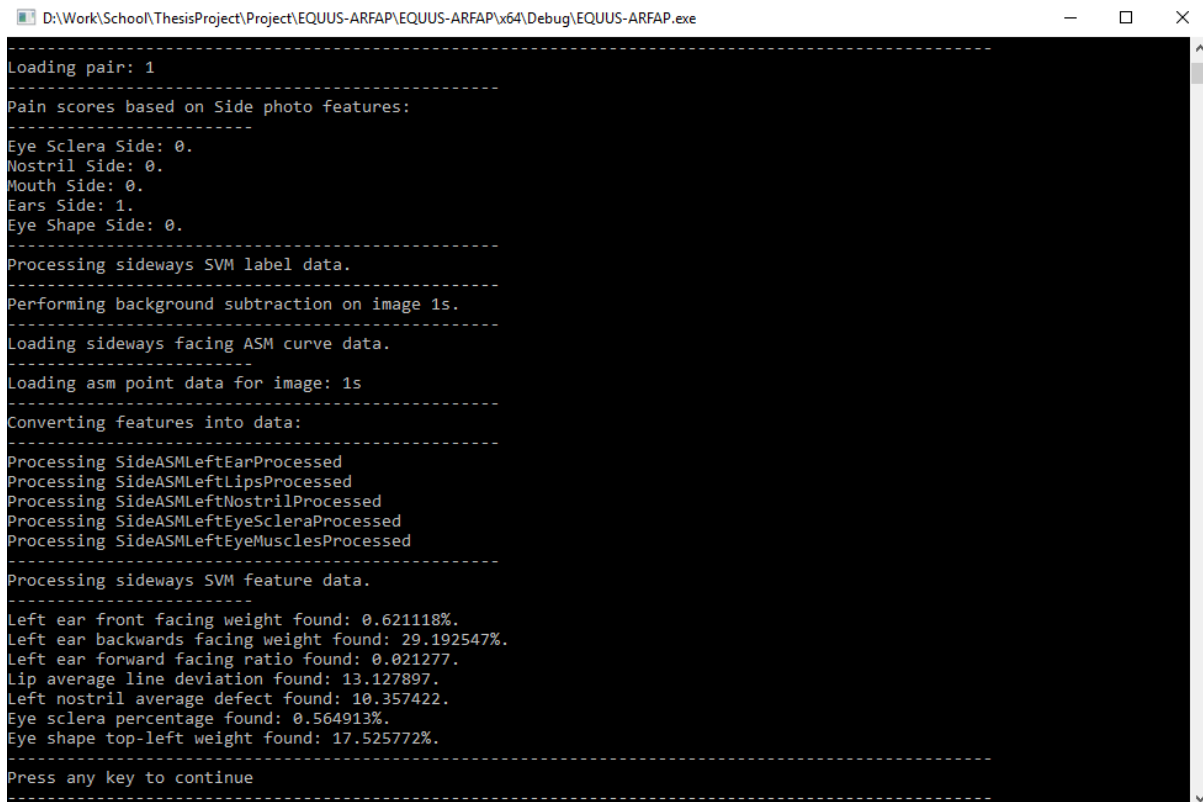


Figure 31. Lip detection by using adaptive mean thresholding and image thinning to find the strongest edge within the area, with hand-drawn visualization of the best-fitting line to compare against. Both a straight and more curved lip are shown.

5.2.3.4 – Data Training

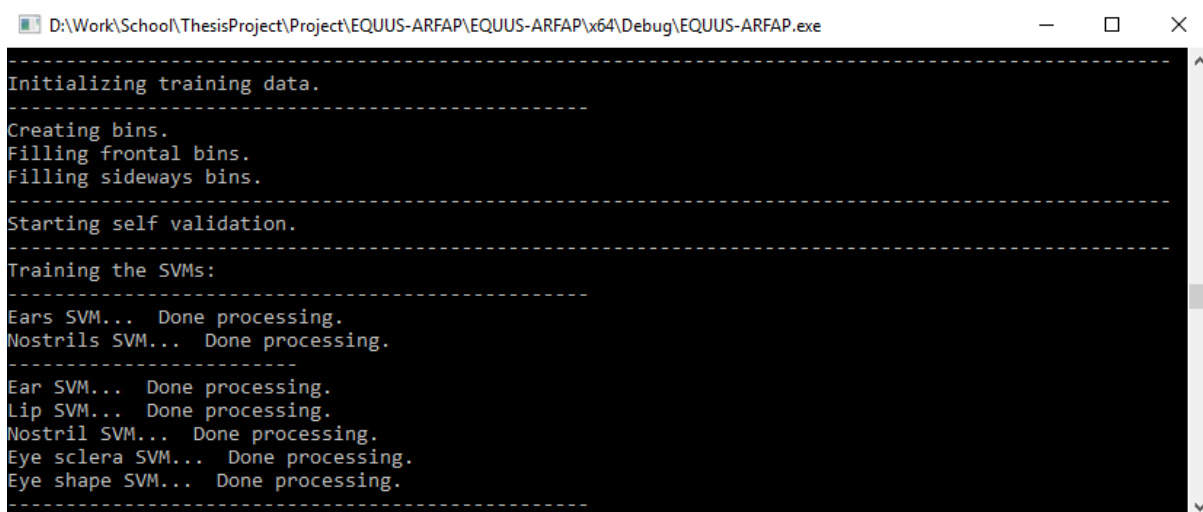
Once all features have been converted into data they are combined with their respective pain-score to create a value-label data-pair that can then be fed into each features' respective SVM. For an example see Figure 32. For some features more than one data-value is used, for example the front ears consist of facing values for both ears separately.

Once all these values have been collected the SVM are trained on them, allowing for testing to be performed, of which an example can be seen in Figure 33.



```
D:\Work\School\ThesisProject\Project\EQUUS-ARFAP\EQUUS-ARFAP\x64\Debug\EQUUS-ARFAP.exe
-----
Loading pair: 1
-----
Pain scores based on Side photo features:
-----
Eye Sclera Side: 0.
Nostril Side: 0.
Mouth Side: 0.
Ears Side: 1.
Eye Shape Side: 0.
-----
Processing sideways SVM label data.
-----
Performing background subtraction on image 1s.
-----
Loading sideways facing ASM curve data.
-----
Loading asm point data for image: 1s
-----
Converting features into data:
-----
Processing SideASMLeftEarProcessed
Processing SideASMLeftLipsProcessed
Processing SideASMLeftNostrilProcessed
Processing SideASMLeftEyeScleraProcessed
Processing SideASMLeftEyeMusclesProcessed
-----
Processing sideways SVM feature data.
-----
Left ear front facing weight found: 0.621118%.
Left ear backwards facing weight found: 29.192547%.
Left ear forward facing ratio found: 0.021277.
Lip average line deviation found: 13.127897.
Left nostril average defect found: 10.357422.
Eye sclera percentage found: 0.564913%.
Eye shape top-left weight found: 17.525772%.
-----
Press any key to continue
-----
```

Figure 32. Example of features being converted into data to be fed into the SVMs.



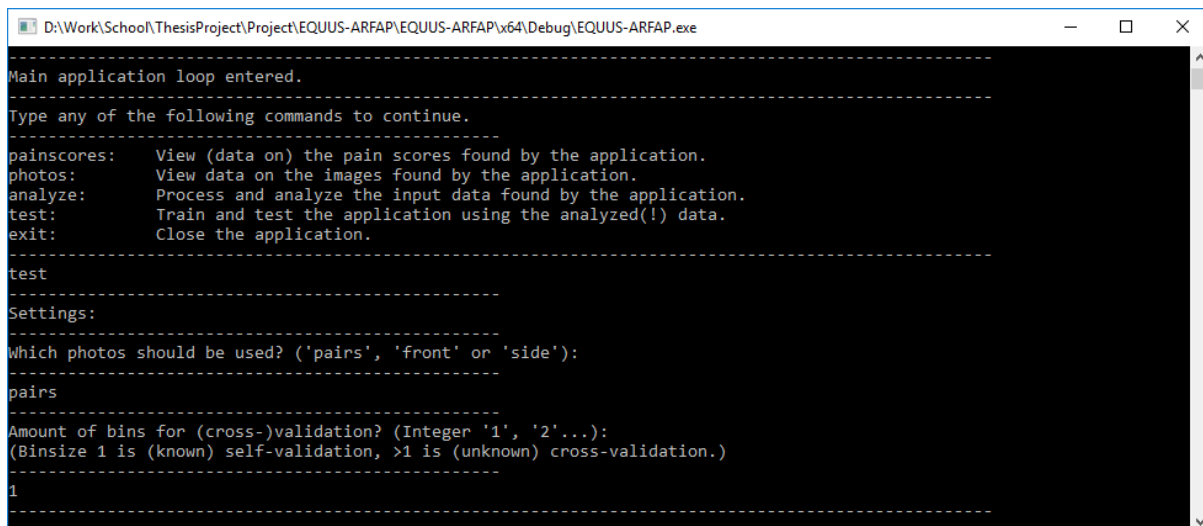
```
D:\Work\School\ThesisProject\Project\EQUUS-ARFAP\EQUUS-ARFAP\x64\Debug\EQUUS-ARFAP.exe
-----
Initializing training data.
-----
Creating bins.
Filling frontal bins.
Filling sideways bins.
-----
Starting self validation.
-----
Training the SVMs:
-----
Ears SVM... Done processing.
Nostrils SVM... Done processing.
-----
Ear SVM... Done processing.
Lip SVM... Done processing.
Nostril SVM... Done processing.
Eye sclera SVM... Done processing.
Eye shape SVM... Done processing.
-----
```

Figure 33. Example of the training of the SVMs.

5.2.4 – Validation

As the application now has a saved set of every analysed feature together with its respective pain-score, training sets can be easily created and validated from the data by making specific subsets thereof and running their features through the trained SVMs predict functions, checking the calculated pain-scores against their actual/ground-truth determined scores.

The application has been built to both encompass self-validation, where all data is used to train on and this trained data is then run through the SVMs once again to test the accuracy of its generalization, as well as cross-validation, where a part of the data is left out of the training data and this untrained data is then run through the SVMs, to calculate scores on what is effectively unknown data. For the cross-validation the user can specify the number of bins they want to test with (for an example of these settings see Figure 34). When running the test, it then splits all data into the bins and continuously trains the SVM with all but one bin. It then starts calculating the accuracy of the SVM when calculating pain-scores of the final unknown bin. Once this has been done for all bin combinations the overall accuracy of all these tests gets averaged to give a general accuracy of the application on unknown data under the given settings.



```
D:\Work\School\ThesisProject\Project\EQUUS-ARFAP\EQUUS-ARFAP\x64\Debug\EQUUS-ARFAP.exe
-----
Main application loop entered.
-----
Type any of the following commands to continue.
-----
painscores:  View (data on) the pain scores found by the application.
photos:      View data on the images found by the application.
analyze:     Process and analyze the input data found by the application.
test:       Train and test the application using the analyzed(!) data.
exit:       Close the application.
-----
test
-----
Settings:
-----
Which photos should be used? ('pairs', 'front' or 'side'):
-----
pairs
-----
Amount of bins for (cross-)validation? (Integer '1', '2'...):
(Binsize 1 is (known) self-validation, >1 is (unknown) cross-validation.)
-----
1
-----
```

Figure 34. Example of the settings used for the data validation.

5.2.5 – Data Subsets

Due to the features of interest on the equine-face not all being visible from a single facing it was decided to split the input data into two facings, frontal and sideways. This allowed for better measuring of certain scores/features, although it does require almost all steps of the application to be done twice (two ASMs, different descriptors for the features, twice as much photos as pain-moments...). The visible features on the frontal photos are the Ears and Nostrils, while those on the sideways photo are the Eye (both the Sclera and the Muscles thereof), Ear, Nostril and Lip. For examples of these two facings see Figure 35 and Figure 37.

Due to an issue with inconsistency of input-data, the frontal facing images have been further split into two more facings; frontal and frontal tilted. This happened due to some of the frontal photos being taken under such an angle that certain features were not symmetrically visible anymore (so only one of the eyes or nostrils for example). This may be of influence on the final results (as the training set has effectively been halved due to this split, this should however only affect the two features that can be viewed from the front, being the Ears and Nostrils), but it is an unfixable issue unless a new data-set is created. For an example of this third facing see Figure 36.

As the frontal and sideways facings count as separate data-moments/implementations, the application can choose either to combine them into one global pain moment (called a Pair), or to analyse them as fully separate datasets (which gives a bit of extra data for each facing as some images for which there is no respective pain-moment pair partner of the other facing do exist).



Figure 35. Frontal facing equine face with marked features.



Figure 36. Frontal tilted facing equine face with marked features.



Figure 37. Sideways facing equine face with marked features.

5.2.6 – ASM Training/Creation

To allow the feature detection to be done within the main control loop of the application, several ASM have been set up to allow for the generic faces and features to be found within each image. As the program deals with multiple facings a different ASM has been set up for each of them, resulting in 3 ASMs to be present within the application.

A lot of files for the ASM have been premade in accordance to the tutorials provided by in the [UoMASM Literature](#). For example, files like curve descriptions are ground-truth type files that do not need to be generated when training a new ASM, therefore it is needed for there to be a pre-existing version of these kinds of files to be able to train new ASMs later if desired.

For the current version of the application a “ground-truth” ASM has been made using data based on manually placed facial mask placements on all images (see Figure 38). Making use of the guaranteed accuracy of this ASM as a comparison/base-line and using subsets of these premade placed masks, it could potentially be possible to use a cross-validation like manner to train new ASMs using smaller subsets of the training data. This would allow the ASMs to be tested on the unknown images and create a general average accuracy this method. The code and required files to do this have been made and implemented into the application, however these tests have not yet been performed and have been saved for a later research.



Figure 38. Example of manual ASM masks placed on images of the frontal, frontal tilted and sideways facing respectively.

5.2.7 – (Intermediate) Result Saving

Due to the substantial amounts of (often unchanging) data to process by our application, the choice has been made to make it save intermediate results to disk when processing every step described in the main control loop. See Figure 39 for an example of these kinds of results. This allows for the saving of a large amount of computing time on later runs, due to the application checking whether or not it must process previously processed data, which then instead loads it from disk immediately, as a trade-off for disk memory.

This approach also has the added advantage of allowing for the inspection of every single step of the application separately once it is done running, allowing for more easy debugging and the isolation of certain issues within the application.

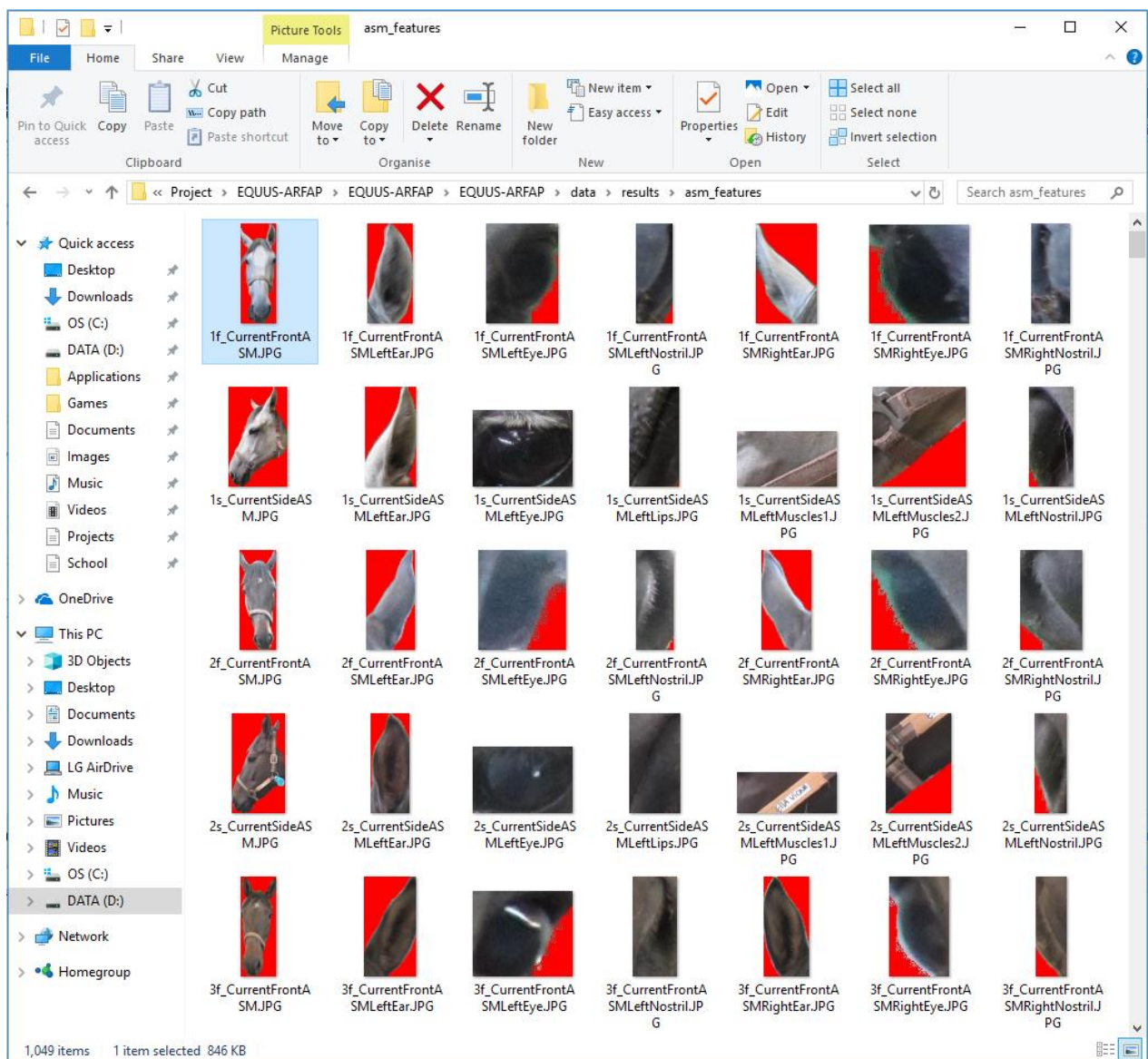


Figure 39. Example of intermediate results saved to drive consisting of background subtraction and facial feature recognition.

6 – Outcome

To answer the original research question(s), the application has been built as described in section 5 – Implementation, and several tests have been run to calculate the accuracies of all possible settings and features mentioned therein, of which the results will be logged in the following sub-section.

As the application does not currently have any restraints on run-time and processing-time, these tests can be replicated using any setup, with the only system requirement being that it would have to be able to run [Microsoft Visual Studio](#), as the program is built in and runs on this IDE.

6.1 – Results

Table 1 and Table 2 contain the percent-based accuracy scores of each features pain-score calculation by the SVM when making use of a given amount of bins, where using 1 bin results in a self-validation where all data is both trained on and tested with, and where an x amount of bins results in the application training the SVM on all but 1/x parts of the data. This is then used to test with to allow for unknown data testing (where this is performed x times and averaged over all these runs to test multiple configurations and get more accurate results/have the ability to test on even more unknown/-trained data). The first few rows of the table are the scores of specific features, where the highest score over all bin options is highlighted in bold and underscored for clarification. The bottom two rows, Average and Total, are respectively used to both measure the average of all above-listed features within the table, and to measure the amount of pain-moments where all features were labelled correctly (so the percentage of overall pain-moments with accuracy of 100% as part of the total number of pain-moments).

Both these tables have also been converted into the graphs shown in Figure 40 and Figure 41 for a more clarified/visualized version of the values.

Feature\Bins	1	2	3	4	5	6	7	8	9	10
Front - Ears	40,28	40,28	43,05	40,27	44,44	40,28	<u>50</u>	40,28	41,67	44,44
Front - Nostrils	33,33	41,67	<u>43,05</u>	38,88	30,55	38,89	36,11	41,67	31,94	33,33
Front - Average	36,81	40,97	<u>43,05</u>	39,58	37,5	39,58	<u>43,05</u>	40,97	36,81	38,33
Front - Total	23,61	<u>33,33</u>	31,94	26,39	22,22	30,56	23,61	25	20,83	25

Table 1. Frontal feature accuracy scores (%) calculated using the number of bins tested with (where 1 bin is self-validation and x bins is cross-validation using 1/x parts of the data as unknown and untrained data).

Feature\Bins	1	2	3	4	5	6	7	8	9	10
Side - Ear	70,79	46,07	59,55	68,54	70,79	<u>74,17</u>	68,54	70,79	68,54	70,79
Side - Lip	59,55	49,44	56,18	<u>60,67</u>	59,55	57,3	57,3	58,53	58,43	59,55
Side - Sclera	82,02	82,02	78,65	80,9	<u>83,15</u>	80,9	79,78	82,02	79,78	82,02
Side - Eyeshape	50,56	48,31	48,31	44,94	51,69	49,44	<u>58,43</u>	49,44	49,44	49,44
Side - Nostril	65,17	59,55	38,2	<u>60,67</u>	56,18	56,18	59,55	57,3	55,06	59,55
Side - Average	<u>65,61</u>	57,08	56,18	63,15	64,27	63,6	64,72	63,6	62,25	64,27
Side - Total	23,6	20,22	21,35	<u>28,09</u>	23,6	24,71	22,47	22,47	26,97	22,47

Table 2. Sideways feature accuracy scores (%) calculated using the number of bins tested with (where 1 bin is self-validation and x bins is cross-validation using 1/x parts of the data as unknown and untrained data).

Front feature correctness percentage

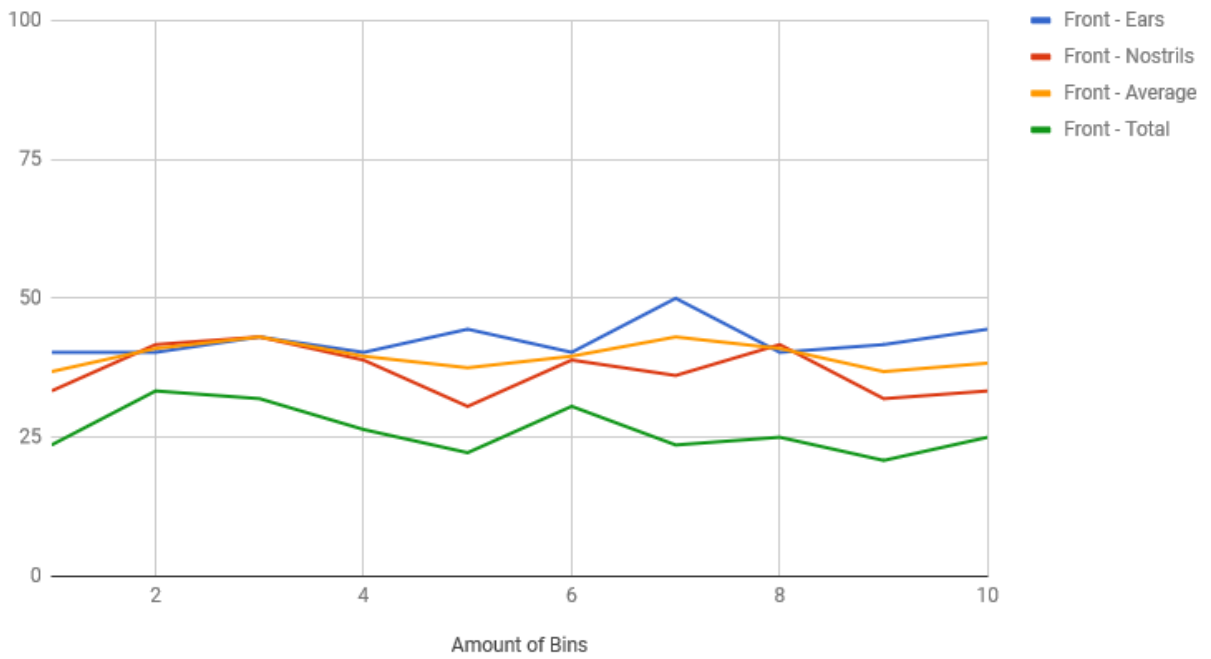


Figure 40. Frontal feature accuracy scores (%) graphed out over the number of bins tested with (where 1 bin is self-validation and x bins is cross-validation using 1/x parts of the data as unknown and untrained data).

Side feature correctness percentage

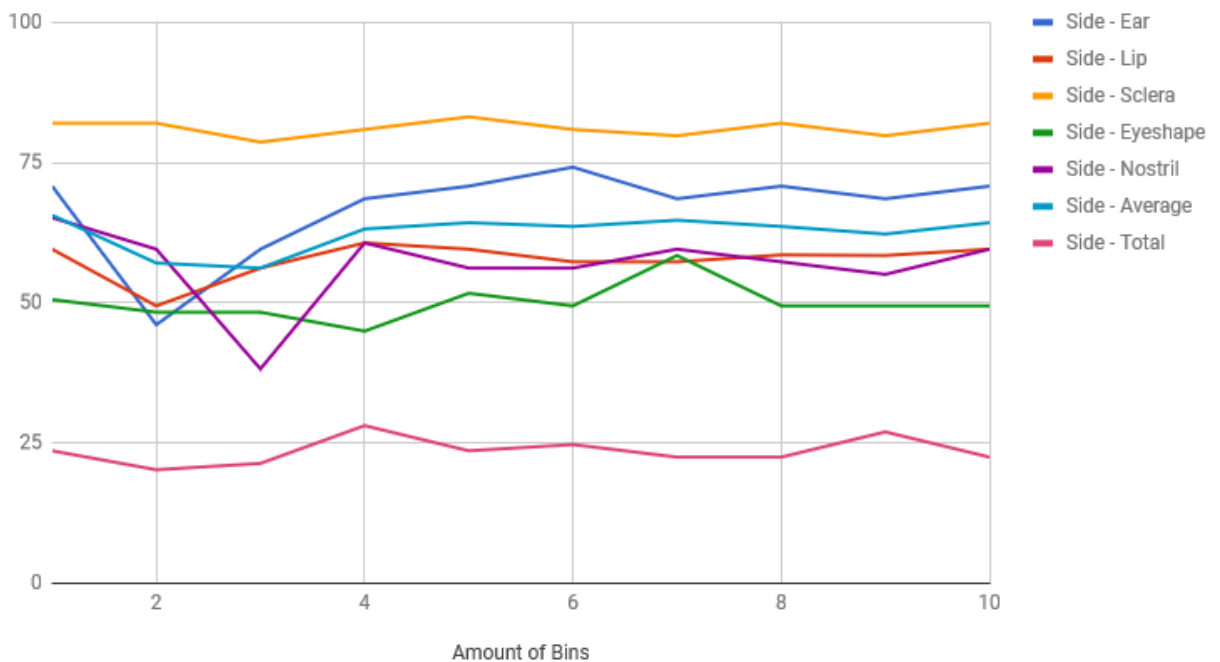


Figure 41. Sideways feature accuracy scores (%) graphed out over the number of bins tested with (where 1 bin is self-validation and x bins is cross-validation using 1/x parts of the data as unknown and untrained data).

6.2 – Evaluation

Before drawing any conclusions, an evaluation comparing these scores to those of similar projects is performed, as well as to make some general remarks that become apparent when looking at the results and to list some of the strengths and weaknesses of the application and its features.

6.2.1 – General Remarks

There are three important general remarks that can be made looking at the results of the application.

The first remark to be made is that the frontal feature accuracy seems to be significantly (about 20 to 25% of overall correctness) lower than that of the sideways features, even though its making use of the same features and algorithms. Even more so it is important that, as these features have a general accuracy lower than 50%, no accurate conclusions can be drawn concerning this data, as any conclusions concerning these would most likely be influenced by their inaccuracy. This does however mean that, considering the considerably higher accuracy on the sideways versions of these features, it might be possible to reduce the program from using multiple images to describe a pain moment to only a single one, which would help solve some other issues that come with this approach, as will be further discussed in section 7.1.2 – Single Facing. For a slightly deeper analysis on why it is thought that this accuracy drop has appeared see sections 6.2.2.3 – Nostrils and 6.2.2.4 – Ears.

The second one is that there is an observed drop in results when using a small number of bins in cross-validation (so around the 2 and 3 bins marks), which can be seen especially well in the sideways feature accuracy graph (Figure 41). This should however not be a problem, as this is actually expected behaviour. This is because when using a very small number of bins the actual training data becomes a very small subset (only 50% of all data at 2 bins), causing the SVM to not have sufficient data to train itself with. This is further proven by the accuracy for all features showing that this behaviour seems to stabilize around a certain value after reaching about 4 or more bins, showing that at this point they have passed a minimum threshold of training data such that more training data does not significantly influence the results anymore.

The third remark to be made is that in all cases, the self-validated accuracy seems to equal the average accuracy of the cross-validation (once enough training data is used). This is both a good and a bad thing depending on how you look at it; the SVMs works by creating a generalizing function for mapping data to scores out of the input data it is given. During self-validation the already trained on data is also fed through this generalizing function, meaning that if a self-validation does not result in a score of 100% there are no clear divisions between the values linked to the pain-scores, which means there are either values calculated in an incorrect way, the training data included wrongly labelled scores, these features are compared using the wrong (not properly separable) data, or a combination thereof. It is however known that our input data was made by experienced professionals, so their data being wrong is the more unlikely of the options. However, the fact that these two scores are equal does mean that the algorithms can judge unknown data with the same accuracy as they do known data, meaning that if the self-validated accuracy can be increased it should also directly increase the cross-validated accuracy. This means that the application is very robust in dealing with unknown data, and as this is one of the end goals of the application, this is a very good thing to see from the results. This in turn shows that the applications method should be a good basis to build and improve upon, especially if new/better ways can be found to convert the features into data.

6.2.2 – Strengths and Weaknesses

Outside of the abovementioned fact that the ears and nostrils seem to have poor performance on their frontal angled versions some of the general strengths, weaknesses or other remarks that come with their current implementations shall be listed for each feature that might not be evident by simply looking at their data. This section will also try to include some potential improvements that could be made on the way these features have been implemented.

6.2.2.1 – Sclera

The sclera detection seems to be implemented as well as it could be, which is also reflected in it having the highest accuracy of all the features. The only real problem in this feature is that the amounts of visible eye white that is linked to each pain score sometimes gets labelled as if belonging to another category, meaning that the currently implemented generalization cannot yield any better results. This might be improved on by counting the amount of eye-white over not just the entire eye area but a different subsection thereof, or possibly by also looking at the shape of the found areas.

6.2.2.2 – Eye Shape

The algorithm for calculating the curvedness of the eye seems to work without a problem on its own, however the detection of the exact curve to look at is the challenging aspect of this feature. Due to the strongest curve being either the one directly against the eye or of the muscles a bit above it there already isn't an exact guarantee what part of the eye needs to be looked at. Furthermore, a lot of functions that could be used here run into the problem of there being a lot of irrelevant data nearby that might still get registered. For example, trying to find the curve through a histogram of gradients would most likely result in the eyeball having the strongest curve around it, especially if there is sclera present or if the horse has a colour strongly contrasting the black of its eye (like white). Methods that rely on the dark colour caused by the shading of this curve (like how the line of the lips is detected) would also generally fail due to the pitch-black colour caused by the eyeballs, combined with the potential of being barely visible on black horses or getting confused on those with strong black eyelashes. For these reasons the biggest improvements that can be made here are to find a better/new way to detect the best usable curve to measure the eye-shape, as the analysis of this seems to be fine so far.

6.2.2.3 – Nostrils

The nostril detection seems to work fairly well under its current implementation. The biggest problems it is currently facing is that the different facings of the nostrils require different types of shape analysis to be done upon them, and there are some circumstances where it can be hard to detect the nostril. The sideways facing angle photos are taken from a very consistent angle, however the frontal facing photos have a very inconsistent range of exact facing-angle. This strongly influences the score of the nostrils, as the shapes belonging to each score look differently depending on the angle you look at them with, which is mostly reflected in the frontal nostril scores being so much lower than the sideways ones. However, the second problem the nostrils still face, especially in the sideways one (which is what would most strongly improve the accuracy there), is that the varying nature of the nostrils causes them to be difficult to generalize on. Finding the general area of the nostrils is not the hardest task, but describing its exact area is quite the challenging task due to differently coloured horses having differently coloured nostril insides, and different light-angles can cause the inherent shadows present in the centre of the nostril to disappear or deform as well. Therefore, the biggest improvement to be made to the nostril detection would most likely be a more consistent/concise way of marking the area of the nostril, as this currently sometimes misses multiple spots causing it to miss-guess the shape of the nostril.

6.2.2.4 – Ears

The ear detection for the frontal ears faces the same problem as the frontal nostrils, this being that due to the variance in angles it is hard to make proper generalizations about the angle of the ear. Going to the sideways ears for further analysis an interesting strength and an interesting weakness of the current application is found. The current generalization function of the SVM seems to draw a hard line between forwards and backwards facing ears, causing both of these (so those with scores of 0 and 2) to be found with 100% accuracy. However, the downside to this generalization is that the sideways facing ears (those with a score of 1) are almost never recognized, causing those to have a near 0% accuracy. Therefore, the best way to improve on this feature would be to find a separate way of testing for the ear facing sideways and combining this with or using this to expand upon the current implementation, for example by checking for visibility of the inner-ear shell or by doing some form of shape comparison.

6.2.2.5 – Mouth

The mouth detections analysis technique (of comparing the found line to a best fitting one to measure stiffness/straightness) also seems to be a proper way of performing the analysis. However, the problem in this feature mostly lies in the way the mouth is detected. The line of the lips is in general visible as a (shadow) line of variable thickness. The biggest problem here is that unless this shadow line is of equal thickness a thinning algorithm tends to create a mildly branched skeleton and these branches count as pixels that have a deviation from the best fitting line, even though they are usually not of importance to the actual line itself. Another problem that can occur is that very dark-coloured horses tend to have barely visible lip-lines, which works especially badly when detecting them through colour. However, performing detection through something like (HOG-based) edge detection tends to work less well on the horses where this shadow is properly visible, causing there to be no easy general way to detect the inner lip-line. The best way to improve upon this feature therefore would be to either improve upon the thinning algorithm by implementing a branch removal algorithm, filtering it down to its strongest/longest single branch-line, or by finding some other more consistent method of detecting the lip-line, potentially by using another form of edge detection that has not yet been explored within this study.

6.2.3 – Result Comparison

The best results to compare this application against are those of the similar sheep facial pain recognition project described in [1]. They also used SVMs to generate their results (also in a percentage-based accuracy format), as well as both self-validation and cross-validation based tests to calculate results on. The results of their self-validation can be found in Figure 42, and of their cross-validation in Figure 43.

As their Linear SVM is the one with the highest mean score and therefore the one they chose to continue their tests on, which means their applications mean score in self-validation is 67%, with a value range of 20% - 88%. Their mean cross-validation based score is 49%, with values ranging from 0% to 77%.

These self-validated accuracies of this application are 66% for the sideways angle and 37% for the frontal one, which means that in comparison this projects sideways accuracy is on equal level with theirs, although the frontal accuracy is, as stated before, lacking quite a bit.

For the cross-validated accuracy however the sideways accuracy is ~63% and the frontal accuracy is ~39%, meaning that, for the sideways score, this application is actually more generalizable than theirs (although for the frontal one it is however once again lacking).

Seeing as how this application has similar results to theirs, it is also safe to conclude that their research assumption of their techniques being generalizable to other animals has been proven through this research. This also means that it should also be safe to conclude that the techniques used within this application should once again be generalizable to other animals.

CLASSIFICATION ACCURACY OF OUR 3-CLASS AU CLASSIFIERS COMPARED TO MAJORITY VOTE CLASSIFIER. WE COMPARE SVM LINEAR, REF KERNEL AND SIGMOID FUNCTION. AS SHOWS, LNR OUTPERFORMS RBF AND SIG FOR MOST AU'S. LNR ALSO HAS THE HIGHEST OVERALL DETECTION RATE.

[TRAINED ON SFF, TESTED ON SFF]

Feature	Ear (Left)			Ear (Right)			Nose			Eye (Left)			Eye (Right)			Mean
	AU1	AU2	AU3	AU1	AU2	AU3	AU4	AU5	AU6	AU7	AU8	AU9	AU7	AU8	AU9	
AU Number	210	80	40	200	80	50	100	160	70	230	90	10	220	100	10	-
Sample size	210	80	40	200	80	50	100	160	70	230	90	10	220	100	10	-
Majority Vote	0.64	0.24	0.12	0.61	0.24	0.15	0.30	0.48	0.21	0.70	0.27	0.03	0.67	0.30	0.03	0.33
LNR SVM	0.80	0.61	0.83	0.85	0.65	0.72	0.64	0.49	0.63	0.72	0.82	0.50	0.77	0.88	0.20	0.67
RBF SVM	0.96	0.60	0.80	0.94	0.58	0.76	0.58	0.71	0.59	0.91	0.68	0.10	0.93	0.85	0.00	0.66
SIG SVM	0.96	0.55	0.88	0.97	0.35	0.82	0.47	0.64	0.36	0.85	0.60	0.30	0.82	0.60	0.10	0.62

Figure 42. Self-validation results of the automated sheep pain recognition application, comparing a general Majority Vote based score with their own trained SVMs (with different possible settings). (Image source: [1]).

CROSS-DATASET TESTING, SHOWING THE CLASSIFICATION ACCURACY OF OUR 3-CLASS AU CLASSIFIERS. WE CAN SEE THAT OUR APPROACH IS GENERALISABLE ACROSS DIFFERENT DATASETS [TRAINED ON SFF, TESTED ON SFF]

Feature	Ear (Left)			Ear (Right)			Nose			Eye (Left)			Eye (Right)			Mean
	AU1	AU2	AU3	AU1	AU2	AU3	AU4	AU5	AU6	AU7	AU8	AU9	AU7	AU8	AU9	
AU Number	96	8	13	102	7	8	24	77	16	80	33	4	91	20	6	-
Sample size	96	8	13	102	7	8	24	77	16	80	33	4	91	20	6	-
SVM LNR	0.65	0.63	0.62	0.77	0.43	0.63	0.54	0.65	0.31	0.60	0.39	0.00	0.37	0.10	0.67	0.49

Figure 43. Cross-validation results of the automated sheep pain recognition application trained with the same linear SVM algorithm as mentioned in Figure 42. (Image source: [1]).

7 – Conclusion

Looking back on the original research questions (2.2 – Research Questions), it can be seen that all 5 of our sub questions have been answered in section 5 – Implementation, and the results can now be used to answer the main research question:

“Can an application be created that, when trained using known professionally scored input data, can automatically determine the pain-level of a horse in an image, at the level of a (trained) veterinary?”

To which, making use of the currently drawn results in section 6 – Outcome, it seems safe to say, “Yes, but there is a but”.

This application has definitely proven that it is possible to automatically estimate horse pain-levels using facial photographs, and that this application is a good basis for this. It has been concluded in section 6.2.3 – Result Comparison that this study has similar and, on some levels, even better generalizable results than other similar projects, which is what was necessary as a minimum way of proving the feasibility of this research. This means that there is definitely a basis for exploring the future possibilities of this application/research.

However, there are still quite a lot of things to expand upon before it reaches “the level of a (trained) veterinary”, and future research and expansions will definitely be required to get there. This is why the rest of this section will be used to list some of the conclusions that were drawn that should be considered when making these kinds of improvements/further researches, and also list

the possible improvements and expansions to be made to this method that currently seem to be of use for this purpose.

7.1 – Drawn Conclusions

Analysing the results and creation-process of the application, two main points have been found that should definitely be considered for any follow-up projects on this subject (or at least be taken from this one). Those being the subset of features this project ended up using, and the image facings it made use of.

7.1.1 – Selected Features

Combining the static features of the HGS and EQUUS-FAP pain-scoring scales resulted in the application using a subset of six features that could be used in automatic facial pain-analysis; Eye Sclera, Nostrils, Mouth, Ears, Eye muscles and Chewing Muscles (see Figure 18). Out of these six features, the final version of the application ended up using only five of them.

The reason the application ended up not using the sixth original feature, Strained Chewing Muscles, is because after extracting it and running it through several possible ways of converting the data, no reliable enough method was found to allow the application to say anything about the state of this feature. This is mostly due to muscle activation in this area not actively (de-)forming a part of the face like the Eye Tension muscles do, meaning the only tangible way to measure this would be through shadows or surfaced veins. However, due to the standard harness of horses covering most of this part of the face in almost all situations, this feature is just not clear/visible enough and contains too much occlusion and interference to reliably use in the application. For some examples of filtered feature areas of the Strained Chewing Muscles see Figure 44.

This is why, after repeatedly discussing the issue, it was decided that the final version of the application would not make use of this feature and ended up with a pain-scoring system based on only five of the six mentioned features. Future research to be done on this project should therefore also start from these five features instead of the original 6, resulting in the following proposed pain-scoring scale as used in our final implementation, which can be seen in Figure 45.



Figure 44. Example of extracted Strained Chewing Muscles features.

EQUUS-ARFAP Proposed facial pain score of a photo:

Eye Sclera Visibility

- Eyes normally opened, no sclera visible. 0
- Eyes normally opened, sclera can be seen. 1
- Eyes more/obvious opened, full edge of sclera visible. 2

Nostril Openness

- Nostril half open, no muscular tension. 0
- Nostril more open, some muscular tension. 1
- Nostrils fully open/flaring, strong muscular tension. 2

Mouth/Lips Straightness

- Mouth shape more curved, no muscular tension. 0
- Mouth shape more lifted/straighter, some muscular tension. 1
- Mouth shape fully lifted/straight, strong muscular tension. 2

Ears Facing

- Ears facing forward, distance between ears normal. 0
- At least one ear facing (halfway) backwards, distance between ears may appear wider. 1
- Both ears pointing (stiffly) backwards, distance between ears may appear wider. 2

Eye Shape

- No visible "frown", rounded shape of eye/eyebrow area. 0
- Somewhat visible "frown", straighter shape of eye/eyebrow area. 1
- Visible "frown", hard corner (nearing 90degrees) in eye/eyebrow area. 2

Total ----- .../10

Figure 45. Proposed pain-score to use in automated equine pain-detection applications.

7.1.2 – Single Facing

The other major conclusion that can be drawn from the final implementation is that it seems the application could be simplified to using only a sideways facing angle, skipping the frontal angle altogether, and there are several reasons for this.

The first and most obvious is the result of the application; It is clearly visible that the methods that were used perform about 25% worse when used on the frontal angle. There are several possible reasons this might have happened, most of which have already been covered in sections 6.2.1 – General Remarks, 6.2.2.3 – Nostrils and 6.2.2.4 – Ears. A quick summary of these shows that the frontal pictures were taken under too inconsistent angles, causing the generalization on them to be very poor (which is otherwise one of the strong points of the implementation).

Furthermore, all features read from the frontal facing can also be read from the sideways one, meaning that it adds no new/unique information by being present. Combining this with the abovementioned fact that the versions read from the sideways angle have better accuracy scores than their respective frontal counterparts, this means that both in quality and quantity of results the frontal angle has either no or a negative impact on the results of the application as a whole.

On top of that there is also the problem that splitting a single pain-moment into two separate photos comes with the immediate problem that, unless using some pretty intricate setup, the pictures won't be taken at the exact same instant, meaning that there could be differences between

these two pictures causing the pain-moment to not truly be a single pain moment. Which has already often resulted in pain-moment pair scores in the application having different ear/nostril scores depending on whether the frontal or sideways image is analysed.

Taking all of these facts into consideration it can be concluded that it would be better for a future implementation to only make use of a single (sideways) facing. As adding a frontal facing adds no new information to the pain-moment analysis, and even if it did, it would still cause the pain-moment to require analysis from two photos instead of one, which would cause a much less realistic/valid moment capture.

7.2 – Future Work

Throughout the creation of this application and while analysing the results several potential improvements have been found that could be made to further enhance the accuracy of this implementation, as well as potential expansions that might improve upon the application by taking certain parts into entirely new directions.

7.2.1 – Improvements

The first and most obvious improvement that could be made to the application would be to increase its accuracy by improving the feature data conversion and analysis. This has already been covered for each individual feature in section 6.2.2 – Strengths and Weaknesses, of which the general summary is that the best way to currently improve the application would be by increasing the accuracy of the feature recognition parts of the data conversion.

Another clear improvement, although not one that would increase accuracy but rather one that drastically improves the generalizability of the application, would be to actually implement a background subtraction method and to train the ASM to generate its own shape masks. These are both currently done using ground-truth values, and both have foundations / a basis to be built and tested upon in this application. To make the application work on completely unknown images these methods do have to be properly implemented, as it also needs to work on files for which no ground-truth background-mask and ASM exist yet.

If for some reason it would be preferred to keep the frontal facing instead of discarding it then another improvement to the application could be to expand upon the ASM functionality by using a Multiview [42], which is an expanded version of ASM that works with multiple facings. However, it has already concluded that it would be best to only use a single facing, and on top of that a Multiview implementation would require more training data than there currently is available to properly work (especially due to the varying frontal facings).

7.2.2 – Expansions

One of the possible expansions/alterted version of this application that has been discussed during its creation would be to, instead of turning the photos into 2-Dimensional ASMs, turn the photos into 3-Dimensional representations using Morphable Models [9], which might open up whole new ways of analysing the training data. This was however both way out of scope for this project, as it would require training a new generalized equine face model from the ground up and would require a vast database of 3D horse imagery, which was not available during this study. This would also be a very large research step to take on an application for which the viability had not yet been proved. However, now that this application has done so this might be a worthwhile investment in some future implementation if such a dataset ever becomes available.

Another big expansion that has been discussed would be to use a Neural Network [43] instead of the implemented SVM based method to have the application teach itself how to recognize the pain-

scores. This was once again way out of scope for this study and would require immensely more data than was available. However, as this application has proven that an automated form of equine facial pain-detection should be possible, this might be a worthwhile field to explore if a dataset large enough to do so ever becomes available.

One final and much smaller/easier expansion to make would be to use (sideways facing) short videos instead of pictures when calculating the pain moments, especially once the application has some more improved-upon results on single images. This is because, as the current implementation is only looking at a single instant in time, the resulting pain-score might not actually be a realistic representation of the amount of pain a horse is actually in. For example, the horse could have a single moment where the amount of pain shown is much less/worse than its overall/average level of pain. Using a short video, filtering out some equally spaced out frames and analysing and averaging the pain-scores of each frame could help create much more realistic and accurate pain assessments once an application reaches the level to actually be used on a commercial level.

Acknowledgements

First and foremost, I would like to thank both my supervisors, Remco Veltkamp and Thijs van Loon, as without them this project wouldn't have been here in the first place. Remco acted as my regular/daily supervisor and has helped steer me in the right technical direction on several occasions and has helped make sure I always knew what to do. Thijs acted as my secondary advisor and his great insight into the veterinary side of this research combined with the training data and custom resources he provided would have made the project impossible without his guidance. Furthermore, I would like to thank Iris Pluim for the research project she did from which the training data used in this application was gathered, and of course also the horses that were used in her (and my) research.

Outside of the research itself I'd like to thank my friends, family and girlfriend, whose continuous support helped keep me sane, healthy and motivated throughout the entirety of the project.

References

- [1] L. Yiting, M. Mahmoud and P. Robinson, "Estimating Sheep Pain Level Using Facial Action Unit Detection.," in *Automatic Face & Gesture Recognition (FG)*, 12th IEEE, 2017.
- [2] J. P. van Loon and M. C. van Dierendonck, "Monitoring equine head-related pain with the Equine University scale for facial assessment of pain (EQUUS-FAP).," *The Veterinary Journal*, vol. 220, pp. 88-90, 2017.
- [3] E. Dalla Costa, "Development of the Horse Grimace Scale (HGS) as a pain assessment tool in horses undergoing routine castration.," *PLoS one*, vol. 9, no. 3, 2014.

- [4] C. L. Gwen, "Faces of pain: automated measurements of spontaneous all facial expressions of genuine and posed pain.," 2007.
- [5] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Active shape models-their training and application," *Computer vision and image understanding*, vol. 61, no. 1, pp. 38-59, 1995.
- [6] P. Pérez, C. Hue, J. Vermaak and M. Gangnet, "Color-based probabilistic tracking.," in *European Conference on Computer Vision*, Springer, Berlin, Heidelberg, 2002.
- [7] M. C. Burl, M. Weber and P. Perona, "A probabilistic approach to object recognition using local photometry and global geometry.," in *European conference on computer vision*, Springer, Berlin, Heidelberg, 1998.
- [8] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325-376, 1992.
- [9] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces.," in *26th annual conference on Computer graphics and interactive techniques.*, 1999.
- [10] Craig, D. Kenneth, M. Kenneth, Prkachin and G. V. Ruth, "The facial expression of pain," in *Handbook of pain assessment 2*, 1992, pp. 257-276.
- [11] S. van Rysewyk, "Nonverbal indicators of pain.," *Animal Sentience: An Interdisciplinary Journal on Animal Feeling*, vol. 1, no. 3, p. 30, 2016.
- [12] P. Ekman and W. V. Friesen, "Facial action coding system," 1977.
- [13] H. Jihun, "Automated facial action coding system for dynamic analysis of facial expressions in neuropsychiatric disorders.," *Journal of neuroscience methods*, vol. 200, no. 2, pp. 237-256, 2011.
- [14] Y. Gizatdinova, Veikko and Surakka, "Feature-based detection of facial landmarks from neutral and expressive facial images.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 1, pp. 135-139, 2006.
- [15] J. J.-J. Lien, T. Kanade, J. F. Cohn and C. C. Li, "Detection, tracking and classification of action units in facial expression.," *Robotics and Autonomous Systems*, vol. 31, no. 3, pp. 131-146, 2000.
- [16] A. Ryan, "Automated facial expression recognition system.," in *Security Technology, 43rd International Carnahan Conference*, 2009.
- [17] A. B. Ashraf, "The painful face-pain expression recognition using active appearance models.," *Image and vision computing*, vol. 29, no. 12, pp. 1788-1796, 2009.
- [18] K. B. Glerup, "An equine pain face," *Veterinary anaesthesia and analgesia*, vol. 42, no. 1, pp. 103-114, 2015.
- [19] K. B. Glerup and C. Lindegaard, "Recognition and quantification of pain in horses: a tutorial review.," *Equine Veterinary Education*, vol. 28, no. 1, pp. 47-57, 2016.

- [20] J. C. de Grauw and J. P. van Loon, "Systematic pain assessment in horses.," *The Veterinary Journal*, vol. 209, pp. 14-22, 2016.
- [21] J. P. van Loon and C. M. van Dierendonck, "Monitoring acute equine visceral pain with the Equine Utrecht University Scale for Composite Pain Assessment (EQUUS-COMPAS) and the Equine Utrecht University Scale for Facial Assessment of Pain (EQUUS-FAP): a scale-construction study.," *The veterinary Journal*, vol. 206, no. 3, pp. 356-364, 2015.
- [22] J. P. van Loon and M. C. van Dierendonck, "Monitoring acute equine visceral pain with the Equine Utrecht University Scale for Composite Pain Assessment (EQUUS-COMPAS) and the Equine Utrecht University Scale for Facial Assessment of Pain (EQUUS-FAP): a validation study.," *The Veterinary Journal*, vol. 216, pp. 175-177, 2016.
- [23] J. Wathan, "EquiFACS: the equine facial action coding system," *PLoS one*, vol. 10, no. 8, 2015.
- [24] A. M. McIvor, "Background subtraction techniques.," in *Image and Vision Computing 4*, 2000.
- [25] D. G. Lowe, "Object recognition from local scale-invariant features.," in *Seventh international conference on Computer Vision*, 1999.
- [26] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features.," in *Computer Society Conference on Computer Vision*, 2001.
- [27] Y.-I. Tian, T. Kanade and J. F. Cohn, "Recognizing action units for facial expression analysis.," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 97-115, 2001.
- [28] X. P. Burgos-Artizzu, P. Perona and P. Dollár, "Robust face landmark estimation under occlusion.," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013.
- [29] S. Milborrow and F. Nicolls, "Locating facial features with an extended active shape model.," in *European conference on computer vision.*, Heidelberg, Berlin, 2008.
- [30] K. Seshadri and M. Savvides, "Robust modified active shape model for automatic facial landmark annotation of frontal faces.," in *Biometrics: Theory, Application and Systems*, 2009.
- [31] D. Navneet, B. Triggs and C. Schmid, "Human detection using oriented histograms of flow and appearance.," in *European conference on computer vision*, Springer, Berlin, Heidelberg, 2006.
- [32] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns.," *Communications of the ACM*, vol. 27, no. 3, pp. 236-239, 1984.
- [33] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.
- [34] E. Osuna, R. Freund and F. Girosit, "Training support vector machines: an application to face detection.," in *Computer vision and pattern recognition, IEEE computer science conference*, 1997.
- [35] R. Brunelli, "Template matching techniques in computer vision," John Wiley & Sons, 2009.

- [36] A. Hill, C. J. Taylor and T. Cootes, "Object recognition by flexible template matching using genetic algorithms.," in *European Conference on Computer Vision*, Heidelberg, Berlin, 1992.
- [37] M. J. Swain and D. H. Ballard, "Indexing via color histograms," *Active Perception and Robot Vision*, pp. 261-273, 1992.
- [38] T. Gevers and A. W. M. Smeulders, "Color-based object recognition," *Pattern Recognition*, vol. 32, no. 3, pp. 453-464, 1999.
- [39] J. Murgia, C. Meurie and Y. Ruichek, "An improved colorimetric invariants and RGB-depth-based codebook model for background subtraction using kinect.," in *Mexican International Conference on Artificial Intelligence.*, 2014.
- [40] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection.," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.
- [41] S. Milborrow and F. Nicolls, "Active Shape Models with SIFT Descriptors and MARS," *VISAPP*, 2014.
- [42] S. Millborrow, T. E. Bishop and F. Nicolls, "Multiview active shape models with sift descriptors for the 300-w face landmark challenge.," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2013.
- [43] L. V. Fausett, *Fundamentals of neural networks: architectures, algorithms and applications.*, Englewood Cliffs: Prentice Hall, 1994.