
KTH Computer Science
and Communication

# Trajectory extraction for automatic face sketching
# Linje extraktion för automatisk ritning av ansikten

RADU-MIHAI PANA-TALPEANU

Master's Thesis at NADA
Supervisor: Josephine Sullivan
Examiner: Stefan Carlsson

# Abstract

This project consists of a series of algorithms employed to obtain a simplistic but realistic representation of a human face. The final goal is for the sketch to be drawn onto paper by a robotic arm with a gripper holding a drawing instrument. The end application is mostly geared towards entertainment and combines the fields of human-machine interaction, machine learning and image processing.

The first part focuses on manipulating an input digital image in order to obtain trajectories of a suitable format for the robot to process. Different techniques are presented, tested and compared, such as edge extraction, landmark detection, spline generation and principal component analysis. Results showed that an edge detector yields too many lines, while the generative spline method leads to overly simplistic faces. The best facial depiction was obtained by combining landmark localization with edge detection.

The trajectories outputted by the different techniques are passed to the arm through the high level interface provided by ROS, the Robot Operating System and then drawn on paper.

# Referat

Detta projekt består av en serie av algoritmer som används för att erhålla en förenklad men realistisk återgivning av ett mänskligt ansikte. Det slutgiltiga målet är att skissen ska ritas på papper av en robotarm med en gripare som håller ett ritinstrument. Tillämpningen är nöjesorienterad och kombinerar områdena människa-maskin-interaktion, maskininlärning och bildbehandling.

Den första delen fokuserar på att manipulera en mottagen digital bild så att banor i ett format lämpligt för roboten erhålls. Olika tekniker presenteras, testas och jämförs såsom kantdetektion, igenkänning av landmärke, spline-generering och principalkomponentanalys. Resultaten visade att en kantdetektor ger alltför många linjer och att spline-genererings-metoden leder till alltför förenklade ansikten. Den bästa ansiktsskildringen erhölls genom att kombinera lokalisering av landmärke med kantdetektering.

De banor som erhållits genom de olika teknikerna överförs till armen genom ett högnivågränssnitt till ROS, Robot Operating System och sedan ritas på papper.

# Contents

# Chapter 1

# Introduction

The number of existing digital images has increased dramatically over the past years and people have found a myriad of ways of manipulating them. Many of the image processing techniques available today are combined with machine learning algorithms and applied to parts of this massive database of photos. The result is a classifier which can identify the presence of faces, buildings, trees, cars, or whatever object it was trained to recognize. Therefore, the field of image processing plays a pivotal role in the evolution of artificial intelligence and machine learning in particular, as it acts as the sight with which machines see the world. In order for an autonomous robot to move it must be able to get information about its surroundings, which comes primarily in the form of digital images.

This project also focuses on image processing as a way to interact with machines. The main goal is to manipulate a given digital image of a face in such a way that a sequence of contour trajectories is obtained, referred to as a line drawing. These trajectories are then communicated to a robotic arm holding a pen, that will attempt to draw a representation of the human face captured in the photo. The principal focus of the work will be the image processing part, while the robotics will play a smaller role in the research, acting mainly as a limitation on the possible line drawings that can be used.

Tresset and Leymarie's work with Paul the robot [17] was an important inspiration for this project. They explore the possibilities of sketching portraits through the use of various techniques ranging from edge extraction to complex shading behaviour. Although the hardware used by the authors differs from the one this project relied on, the essence remains the same: finding an efficient way to automatically draw human faces.

An initial approach is to use edge detection algorithms such as the Canny method or simpler edge filters. The resulting binary image is manipulated to obtain a line drawing that still captures the essential human facial features, while at the same time eliminates unnecessary clutter. The results were poor for all types of edge extraction algorithms tested, due to the difficulty in emphasizing salient regions of the face. Because of the lack of information about the importance of certain areas such as the eyes, nose or mouth, all edges are treated equally and any attempt to trim the clutter edges will also affect the correct detections. The line drawing will either be overrun by unneeded artefacts or it will lack important areas.

An alternative to the edge extraction techniques is the generative approach of producing lines. If coordinates of visual importance would be available, they could be used in a connect-the-dots fashion to draw curves corresponding to the jawline, eyes, eyebrows, nose and mouth. These points are called facial landmarks and there exist very accurate and efficient ways of obtaining them [21]. Once they are available, splines or regular polynomials can be fitted to them and line drawings can be created generatively. The landmark detections will not be perfect, so improvement can also be done here in a series of ways, such as gradient magnitude correction or outlier detection.

By combining an edge detector with the knowledge of landmark locations, the problem of

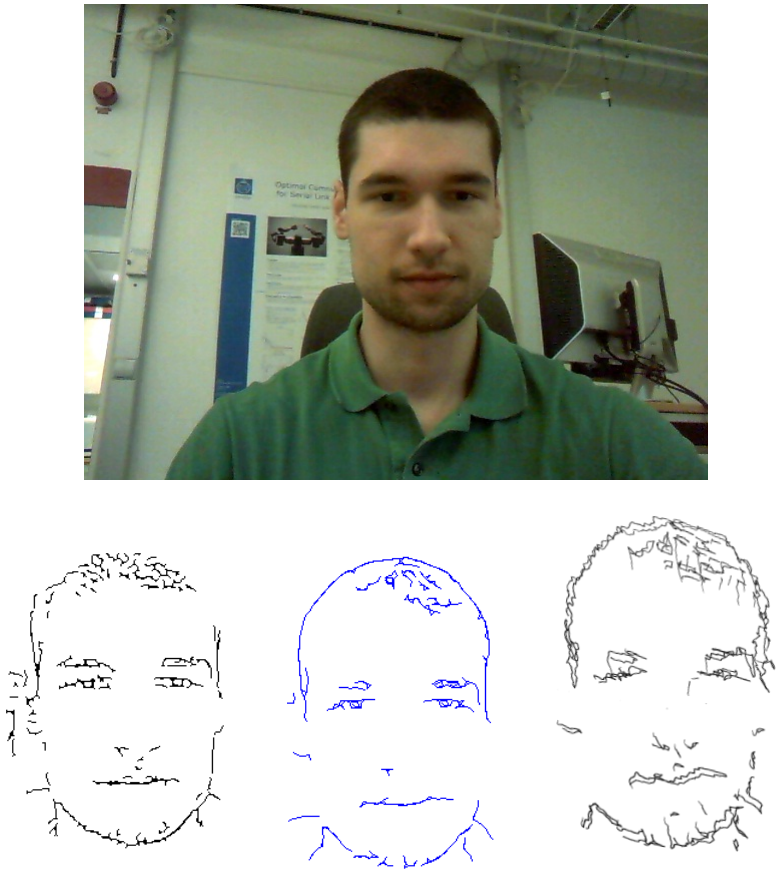

**Figure 1.1.** Overview of the drawing process. Top: original image, bottom: edge extraction, edge trimming, drawn sketch.

clutter edges can be diminished. The landmarks hold information about salient areas, which was lacking when using only edge detection. Further edge trimming by means of imposing a minimum connected component length and applying a skeleton extraction led to visually pleasing results and to the final line drawings sent to the robotic arm.

The robotics part consisted of simulating the arm movement in the framework offered by the Robot Operating System followed by the actual interaction with Dumbo, the dual-arm robot at KTH's CVAP laboratory. The drawing process commences by moving the arm into an initial position of the joints, where the gripper can hold a pen and move it across the paper. Then, movement begins through a series of target coordinates on the drawing plane, during which time the pen is lifted and lowered when needed.

An overview of the entire process is presented in figure 1.1.

# Chapter 2

# Background and related work

The problem tackled is part of the image processing domain concerned with manipulation of human faces, employing techniques that span from detection to the line drawing transformation. The literature of interest therefore ranges from papers on edge extraction techniques to articles on face detection methods. There is also the aspect of the type of representation desired, which can either be photorealistic or non-photorealistic. After a brief exploration of the latter, it was decided to pursue the realistic option, as it is simpler and faster to draw, as it outputs fewer trajectories.

## 2.1 Canny edge extractor

The first paper of interest is J.F. Canny's work on edge detection [1]. His algorithm described in 1986 is considered to give the best overall results and is the most widely used. The steps he takes include an initial smoothing of the image with a low-pass Gaussian filter, followed by computation of the edge gradient and direction for every point in the image by applying an edge detection operator such as a Sobel mask, which returns the value of the first derivative in the horizontal and vertical directions, $G_x$ and $G_y$. The gradient magnitude is therefore

$$G = \sqrt{G_x^2 + G_y^2}$$

and the angle is

$$\Theta = \arctan(\frac{G_y}{G_x})$$

$\Theta$ is rounded to one of four possible directions: 0, 45, 90 and 135 degrees. Next, a non-maximum suppression is carried out in order to remove thick edges and to obtain a binary representation of the edge points. A point is considered to be part of an edge, if its gradient magnitude in direction $\Theta$ is larger than the magnitude in the perpendicular direction.

The final step is the hysteresis thresholding. A high threshold is applied to the binary image in order to be more certain that the edge points are actually correct and not caused by noise (these false edges would have low intensity and will therefore be removed by the threshold). A low threshold is then used in combination with the directional information to permit the drawing of faint edges, as long as a starting point exists. Thus, the final binary representation of edges is obtained.

## 2.2 Robotic drawing of faces

In "Sketches by Paul the robot" [17], the functioning of the drawing robot Paul is described and some sketches are presented. The set-up contains a robotic arm holding a black Biro pen and a web-cam bolted to the table and can be seen in figure 2.1.

**Figure 2.1.** Paul sketching a human subject's face. Reproduced from [17].

The authors use Gabor filters to extract contours, which, in the spatial domain, are Gaussian kernels modulated by a sinusoidal plane wave [10]:

$$g_{\lambda,\theta,\varphi,\sigma,\gamma}(x,y) = \exp(-\frac{x^2+\gamma^2y'^2}{2\sigma^2})\cos(2\pi\frac{x'}{\lambda}+\varphi)$$

where

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\sin\theta + y\cos\theta$$

They suggest that Gabor filters are biologically motivated and contribute to the stylising effect similar to human made drawings.

Afterwards, edge blobs are extracted to which a medial axis transform is applied (skeleton extraction) in order to make them thinner. The authors also mention a shading behaviour, meant to bring more realism to the drawn figure.

Though simplistic, Paul's sketches are aesthetically appealing and the authors credit this to how the basic algorithms are combined.

## 2.3 Non-photorealistic rendering

One of the domains of computer generated drawings is the one where imagination is more important than realism. This is where the idea of non-photorealistic rendering (NPR) comes in. It has been a growing field since the early 90s and provides a wide variety of techniques that can be used to accomplish the line extraction task by combining human-computer interaction, computer vision and graphics.

All the following techniques output some form of "artistic" images from input received as digital photos, which are manipulated through various algorithms. The problem, however, is that the obtained images are very difficult to bring out of the computer screen and onto a piece of paper. A robotic arm has a limited number of degrees of freedom to move and can only hold one pen. All the colours and shading obtained through NPR couldn't be captured in this way. Neither could the precise curves which are a key feature. Therefore, non-photorealistic rendering will remain only an interesting theoretical aspect and is included for completeness.

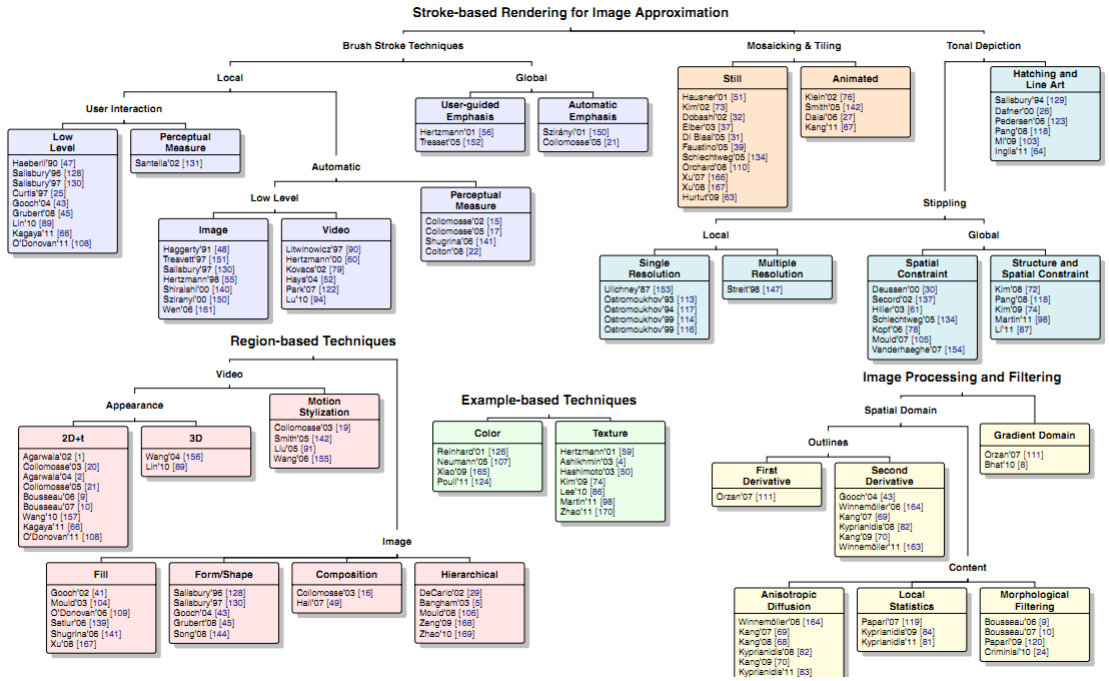


**Figure 2.2.** Taxonomy of SBR techniques. Reproduced from [11].

### 2.3.1 Kyprianidis et al.

A very good presentation of the current state of NPR is given by Kyprianidis et al. in "State of the "Art": A taxonomy of image stylization techniques for images and video" [11]. The part of interest is image based artistic rendering (IB-AR), as the initial input is a 2-D image.

The first NPR techniques used stroke-based rendering, where the canvas would be painted with primitives called strokes. These could be brush strokes consisting of either short dabs, or long curves placed according to local or global criteria. The content of an image could also be approximated by using other primitives like stipples (small points distributed over a region with the purpose of creating a tonal depiction), or mosaics (tiles patched together).

After the year 2000, region-based techniques came into the picture. They used mid-level computer vision algorithms, instead of lower-level ones and the primitives used were whole regions.

Example-based techniques imply a texture or colour transfer. Texture transferring was done by filling holes in an image with similar patches and colour transferring was achieved by mapping the colour histograms of the example image and the image being rendered.

The classical way of determining the stroke placement was with a function of the Sobel gradient. This led to constant size rectangular strokes that were too regular to give an artistic impression. Hertzmann's idea of curved strokes at different coarse-to-fine scales [8] greatly improved renderings and served as a base for many other techniques.

DeCarlo and Santella proposed a segmentation technique by using a variant of mean-shift at multiple down-sampled resolutions [4, 11]. It was semi-automatic as it relied on interaction with a user to observe the location and duration of the viewer's gaze. The important areas were then painted with finer detail. In the case of human faces this step can be replaced by an automatic detection of regions of interest, such as the eyes and mouth and thereafter paint them in finer detail than the rest of the image. This was a step in the direction of artistic rendering as the representations were no longer based on gradient magnitudes but on perceptually important aspects.

Another view of NPR is from a local/global standpoint. Most of the algorithms discussed

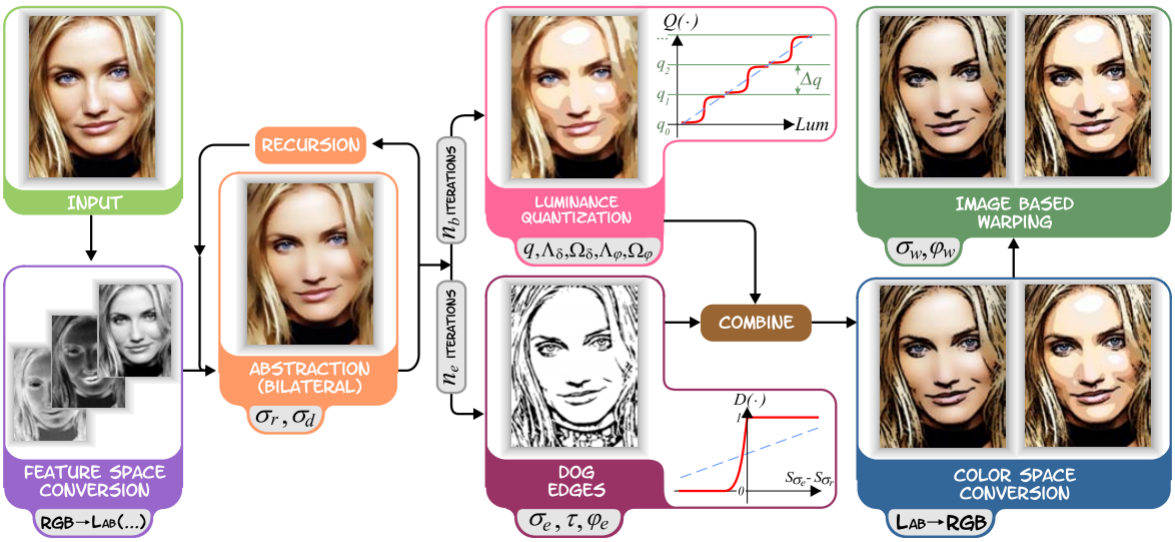


**Figure 2.3.** Automatic cartoon stylization pipeline. Reproduced from [20].

so far approach the problem in a local manner. Global techniques place the rendering elements such as to minimize a global objective function. This function however turned out to be tied to the realistic world most of the times and led to a return to photorealism.

Machine learning and IB-AR come together in the form of rendering by example [9]. Hertzmann et al. considered that if a computer was presented with a pair of images, A and A' and extracted the function that connects them, then, when it would be presented with a new image B, it could create B' that is correlated to B in the same way as A' was connected to A. This can be done either by colour matching, or texture transfer, the latter being the most common approach.

### 2.3.2 Winnemöller et al.

In the paper "Real-time video abstraction" the authors discuss a highly parallel automatic pipeline for cartoon stylization of images and video [20]. The algorithm is depicted in figure 2.3.

Firstly, the RGB coordinates are transformed into CIELab coordinates, which are used to better approximate human vision [19]. Bilateral filtering is applied twice, followed by luminance quantization done in parallel with the extraction of edges using difference of Gaussians. The two results are combined and the outcome is converted back to RGB coordinates, thus providing a cartooned version of the original image.

### 2.3.3 Hertzmann

Aaron Hertzmann describes a method for obtaining an image with a hand painted appearance from a photograph by using a series of spline brush strokes [8]. The procedure follows a pyramid scheme: the bottom of the pyramid is represented by a coarse rendering of the photograph and the higher levels are tied to finer details reserved for important patches of the image. This coarse to fine approach is obtained by using different sizes of brush strokes. Hertzmann also implemented curved brush strokes to obtain a more artistic impression. They are placed according to the image gradient and connect points via a cubic spline. Different styles are also discussed, such as impressionist and expressionist and methods of obtaining them are presented

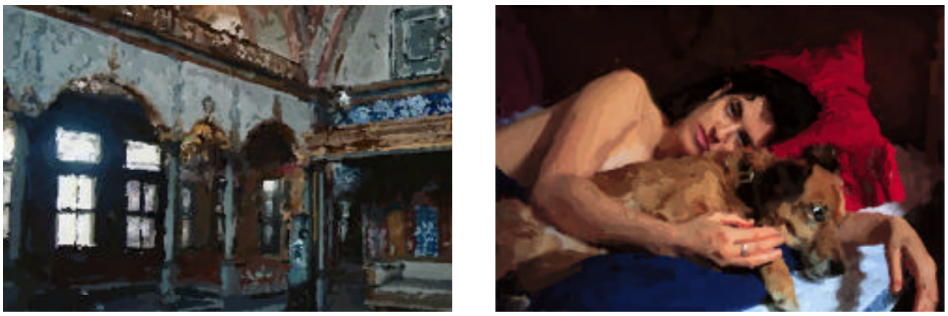

**Figure 2.4.** Two expressionist paintings. Reproduced from [8].

by modifying the parameters of the algorithm. Results obtained by Hertzmann using this method and the impressionist style are presented in figure 2.4.

# Chapter 3

# Obtaining line drawings

The project's principal focus is on finding ways to obtain simplistic but discriminative representations of human faces, referred to as line drawings. This chapter discusses and compares various methods of getting from an input image to a line drawing. All the methods discussed in the following sections are applied in MATLAB and many of them use predefined functions for obtaining some initial starting points. The line drawings are represented as binary images (0 marking the absence of a line pixel and 1 marking the presence of one). They should preferably contain thin, continuous components that could easily be drawn by linearly connecting the pixels of value 1 which make up each component.

## 3.1 Edge extraction

The fundamental way of moving from an average photo to a line drawing is to perform an edge extraction on the grayscale image. Two algorithms are explored further and used throughout the project: the Canny method [1] and the Gabor filters method, employed by the authors of [17].

### 3.1.1 Tresset & Leymarie edges

In their work on Paul the robot [17], the authors explore the use of Gabor filters with up to 8 orientations in order to obtain what they call "salient lines" at different resolutions, starting from coarse and going up to the original scale. The grayscale image is convoluted with the filters, leading to 4 or 8 representations of the different edge orientations (the lowest level of resolution uses only 4 orientations while the other 3 use 8). After thresholding and thinning, these binary images are added together and the salient line figure at that particular scale is produced. The final result does indeed resemble an artist's sketch, as seen in figure 3.1, but it may not be a good candidate for drawing by the robotic arm, as it is quite detailed. An alternative approach could be to use less resolution levels, or less orientations, therefore yielding fewer edges.

### 3.1.2 Canny edge detection

Another option for the task at hand is the one proposed by Canny in 1986 [1]. Unfortunately, the automatic setting of the hysteresis thresholds led to far too many edges. Through empirical observations, a new value for the high threshold was obtained. Even with this modification, some clutter edges remained, problem that was tackled by combining edge and landmark detection, described in the next section. Figure 3.2 shows how much the extracted edge images can vary by modifying the Gaussian blurring variance and/or the hysteresis threshold.

Although it captures edges with high fidelity, there are times when Gabor filters produce better results in terms of human face depiction. Since there is no obvious winner between the

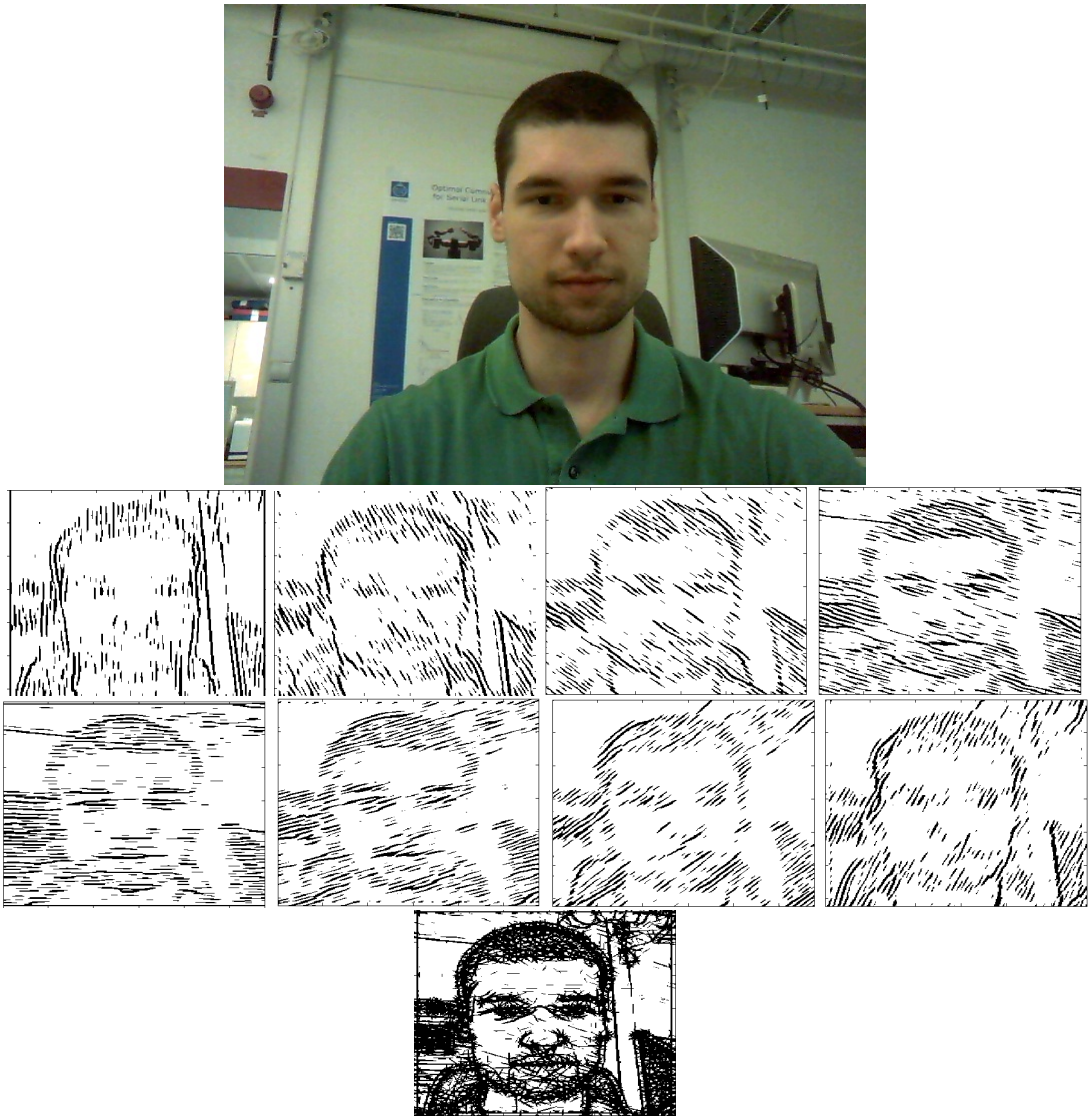

**Figure 3.1.** Results after applying Gabor filters. Each black and white image shows the output from the convolution between the original picture and a filter with a specific orientation. The final image is the sum of the 8 prior ones.

two methods, the decision on which one to use must be made at runtime, by the user, after being shown both options.

### 3.1.3 Results and discussion

The line drawings outputted by the edge extractors mentioned above are similar in complexity, but the Canny algorithm leads to more general results, while the Gabor filters approach yields more sketch like images, as seen in figures 3.1 and 3.2. A tweak in the Tresset algorithm can lower the complexity of the output by only using the original scale. The resulting black and white image is shown in figure 3.3, before and after the skeleton extraction performed for thinning.

An important observation is that parameter settings that work well for a range of images don't exist. By varying the thresholds and blurring of the Canny edge extractor, one can obtain nearly perfect edges when sketching a certain photo, but most likely these settings will not retain optimality in other cases. The same can be said about the Gabor filters and the number of orientations and resolution levels that should be used. There is no way of knowing

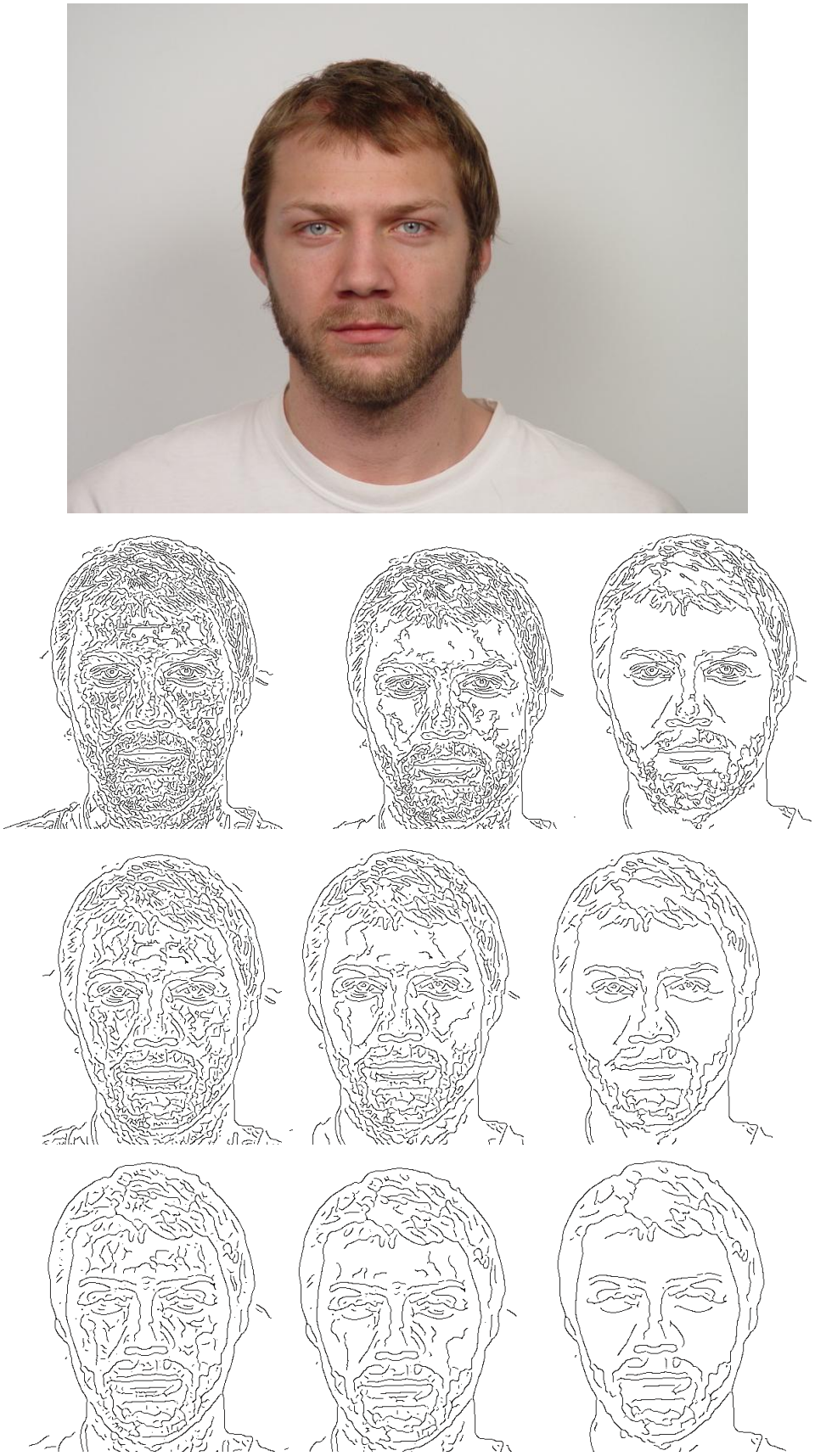

**Figure 3.2.** Canny edges extracted from an image. Thresholds increase from left to right and blurring increases from top to bottom.

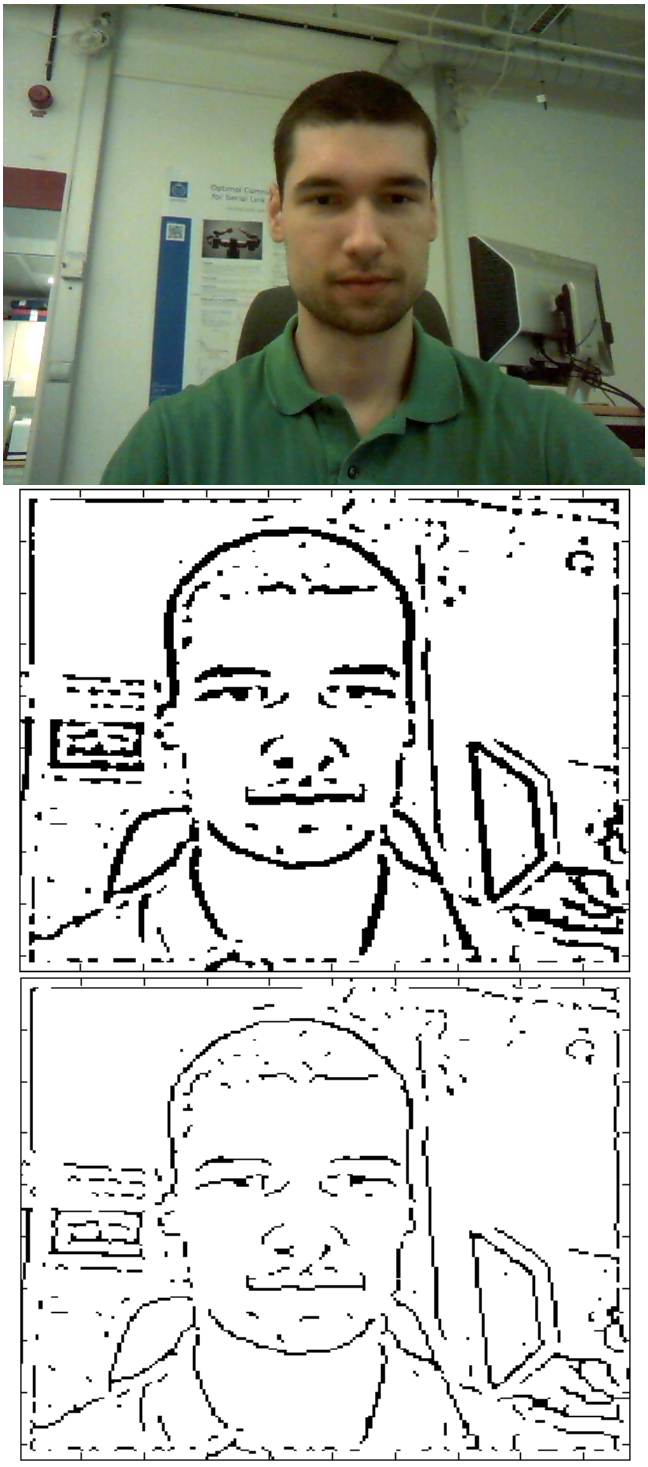

**Figure 3.3.** Results after applying Gabor filters to only the original scale (before and after thinning).

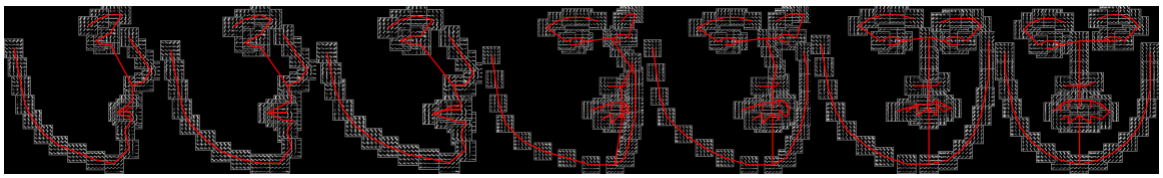

**Figure 3.4.** Mixture of trees to represent topological changes due to viewpoint. Reproduced from [21].

beforehand which edges to keep, without some kind of local information. Therefore, computer vision techniques are needed to help these basic edge detectors to trim clutter.

## 3.2 Detecting facial landmarks

The cornerstone of the final trajectory extraction algorithm to be used with the robotic arm is a basic edge detector. However, on its own, it falls short in respect to the desired outcome, mainly because it does not take into account any local information about the face. Therefore, improvement needs to be built up on top of these basic detectors, which comes in the form of facial landmarks. The goal is to use known salient positions of the subject's face (jawline, eyes, eyebrows, nose) to alter the original edges and create a better looking representation. "Face detection, pose estimation and landmark localization in the wild" [21] describes a method that jointly accomplishes face detection and landmark localization as well as pose estimation. The latter, while useful, is not needed as we only consider frontal images in this project. The authors' results are excellent, surpassing the Viola-Jones detector in terms of finding faces and many state-of-the-art commercial landmark detectors in terms of landmark localization.

### 3.2.1 The landmark extraction model: FMP, flexible mixture of parts

The graphical model is based on a mixture of trees with a shared pool of parts $V$, where each part is a landmark. Topological changes due to viewpoint are captured by global mixtures, like the ones in figure 3.4. Every viewpoint out of the total of $M$ has the same number of landmarks, which will be denoted $N$.

A tree belonging to mixture $m$ (seen from viewpoint $m$) is denoted as $T_m = (V_m, E_m)$ where $V_m$ are the vertices, $V_m \subseteq V$ and $E_m$ the edges. If $I$ is an image, then $l_i = (x_i, y_i)$ is the pixel location of the $i$-th part (landmark). A configuration of parts $L = \{l_i, i = 1, .., N\}$ is scored as follows:

$$S(I, L, m) = App_m(I, L) + Shape_m(L) + \alpha^m \tag{3.1}$$

where

$$App_m(I, L) = \sum_{i \in V_m} w_i^m \cdot \phi(I, l_i) \tag{3.2}$$

$$Shape_m(L) = \sum_{(i,j) \in E_m} (a_{ij}^m dx_{ij}^2 + b_{ij}^m dx_{ij} + c_{ij}^m dy_{ij}^2 + d_{ij}^m dy_{ij}) \tag{3.3}$$

Equation 3.1 scores a configuration of parts by combining an appearance and a shape score. The appearance is scored by placing a template vector $w_i^m$ for part $i$, tuned for mixture $m$ at location $l_i$. $\phi(I, l_i)$ is a feature vector, such as a histogram of oriented gradients [3], extracted from pixel location $l_i$ in image $I$. Equation 3.3 scores a mixture specific arrangement of parts $L$, where $dx_{ij} = x_i - x_j$ and $dy_{ij} = y_i - y_j$ are the displacements along the $x$ and $y$ axis of pairs of landmarks. Every term in the sum is a spring that introduces constraints between a pair of parts. $\alpha^m$ is a bias term associated with mixture $m$.

An interesting aspect of the shape equation is that the location variables $l_i$ only appear in linear and quadratic terms, thus making it possible to rewrite the equation:

$$Shape_m(L) = -(L - \mu_m)^T \Lambda_m (L - \mu_m) + constant$$

where $a, b, c, d$ are re-parametrized to $\Lambda, \mu$. $\Lambda_m$ is a block sparse precision matrix with non-zero entries corresponding to connected pairs of landmarks in the edge set. Shapes that deform from the ideal $\mu_m$ are penalized. The eigenvectors of $\Lambda_m$ with the lowest eigenvalues correspond to configurations with low penalties [21].

**Training**

To train the model, the authors used a supervised discriminative framework with a maximum margin approach. Positive images with annotated landmarks and a given viewpoint, as well as negative examples are presented for learning. First, the tree's edges $E_m$ of each mixture are estimated, by assuming the landmarks are Gaussian distributed and applying the Chow-Liu algorithm [2] to obtain the maximum likelihood tree structure. Positive examples have the form $\{I_n, L_n, m_n\}$, while negative ones are given as images $I_n$, where $n$ is the image index.

The goal is now to simplify the model specified by equation 3.1 down to a dot product form, making use of the terms' linearity. A global vector $\beta$ is created to hold all the appearance ($w$) and all the shape ($a$,$b$,$c$,$d$) parameters, as well as the bias term of each mixture, $\alpha^m$. Let us consider a size of $p$ for a feature vector $\phi(I_n, l_i)$ obtained from the landmark at coordinates $l_i$ in image $I_n$ and denote $z_n = (L_n, m_n)$, the given landmark arrangement and viewpoint of image $n$. The appearance parameters are concatenated as

$$[w_1^1, w_2^1, .., w_N^1, w_1^2, w_2^2, .., w_N^2, .., w_1^M, ..w_N^M]$$

where every $w_i^m$ is of length $p$, so, in total, there are $N \times M \times p$ appearance parameters. Then follow 4 shape parameters for every edge of every mixture, $4 \times M \times |E|$ in total, arranged as

$$S^m = (a_{ij}^m, b_{ij}^m, c_{ij}^m, d_{ij}^m)$$

where $(i, j) \in E_m$ are indices of connected landmarks and $m = 1, .., M$. Finally, the bias terms $\alpha^m$ are added. So, the entire $\beta$ vector looks like

$$\beta = [w_1^1, w_2^1, .., w_N^1, w_1^2, w_2^2, ..w_N^2, .., w_1^M, w_2^M, .., w_N^M, S^1, S^2, .., S^M, \alpha^1, \alpha^2, .., \alpha^M]$$

A sparse vector $\Phi$ is created to obtain the score $S(I, z) = \beta \cdot \Phi(I, z)$. Since for training we know which mixture corresponds to image $n$, the $\Phi$ vector has non-zero entries only for that specific mixture, $m$. For example, if image $n$ is part of mixture 2,

$$\Phi(I_n, z) = [\underline{0}, \phi(I_n, l_i), .., \phi(I_n, l_N), \underline{0}, S^2, \underline{0}, \alpha^2, \underline{0}],$$

where the first batch of 0's cancels the appearance scores of the first mixture, the second batch cancels the appearance scores of the following $M - 2$ mixtures and the shape score of the first mixture. The 0's following $S^2$ cancel the shape scores of mixtures $3, .., M$ and the bias term of the first mixture. The last batch of zeros cancels the bias term for mixtures $3, .., M$. The score becomes:

$$S(I_n, z_n) = \beta \cdot \Phi(I_n, z_n) \tag{3.4}$$

Using this new notation, the model to learn is:

$$\begin{aligned} &\arg\min_{\beta, \xi_n \geq 0} (\frac{1}{2}\beta \cdot \beta + C \sum_n \xi_n) \\ \textit{such that } &\forall n \in pos \quad \beta \cdot \Phi(I_n, z_n) \geq 1 - \xi_n \\ &\forall n \in neg, \forall z \quad \beta \cdot \Phi(I_n, z) \leq -1 + \xi_n \\ &\forall k \in K, \beta_k \leq 0 \end{aligned} \tag{3.5}$$

This formulation enforces the constraint that every positive example should score above 1 and every negative example (with any configuration of landmarks and viewpoints) should score less than -1. Violations of these conditions are penalized by using slack variables $\xi_n$. The expression $\forall z$ implies checking every possible configuration of parts $L$, extracted from all viewpoints $m = 1, .., M$ for a negative image $n$. $K$ are the indices of the quadratic terms $(a, c)$ and the associated negativity constraints ensure that $\Lambda$ is positive semi-definite. After solving the optimization, $\beta$ holds all the trained model's parameters.

#### Inference

After training, inference is used to obtain the viewpoint and landmark locations with the maximum score in a novel image. This is done by maximizing $S(I, L, m)$ over $L$ and $m$ to obtain the optimal value

$$S^*(I) = \max_m [\max_L S(I, L, m)]$$

by enumerating every possible viewpoint (mixture) and finding the best configuration of parts, which is easily accomplished with dynamic programming due to the tree structure of every mixture $T_m = (V_m, E_m)$. There are $N * M$ possible landmark templates in the model. With a dimension of $D$ for every part and considering $S$ possible locations for it, the cost of evaluating all parts at all locations is of order $N \times M \times D \times S$. The cost of message passing becomes $S \times M \times N$ by using distance transforms [5], which makes the authors' approach linear in image size and number of landmarks [21].

#### Practical application

For the purpose of extracting line drawings from face images, the open-source software provided by the authors of [21] was used. Their implementation runs in MATLAB and also makes use of a few C source files. They offer 3 trained models on the MultiPIE dataset [6], two for faces larger than 150 by 150 pixels and one for smaller faces. Depending on the situation, both options were used, but the final code made use of a larger face model. With respect to tuning, there is an available threshold for detection and a possibility to select the number of mixtures, but the default values are sufficiently good. The number of detected landmarks is 68, distributed as follows: 17 along the jawline, 5 for each eyebrow, 6 for each eye, 9 for the nose area and 20 for the mouth. Figure 3.5 shows varying landmark detection results on images similar to the ones intended for subsequent line extraction.

## 3.3 Improving landmark localization

One can, instead of trying to eliminate false edges, approach the problem in a generative way, by using the landmarks to recreate the essential facial features from scratch. An idea is to use splines to approximate curves corresponding to the jaw, eyes, eyebrows and mouth. These spline approximations could then play the role of the trajectories to be drawn. Natural cubic splines proved sufficient for the task. Spline interpolation can also be used to correct the initial locations of the landmarks.

Experiments were done on the eyebrows and jawline as they were the best candidates for spline interpolation, having landmarks aligned such as they could naturally be connected by a smooth curve. Then, improving the eye landmarks was attempted. The nose was not a target of improvement, as the initial detections were correct for most of the test images.

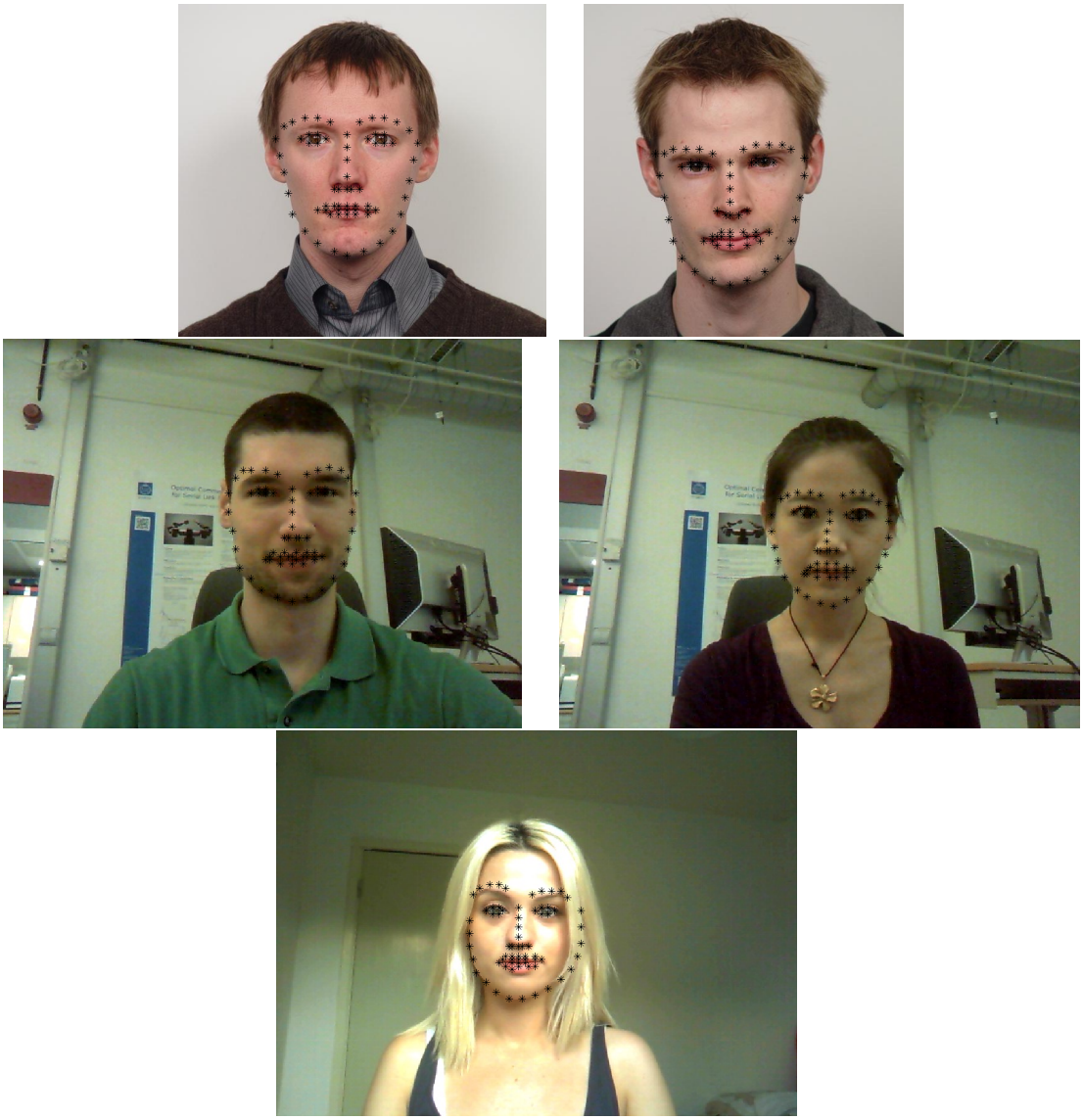

**Figure 3.5.** The 68 facial landmarks of frontal face images, obtained by using the flexible mixture of parts model described in [21]. The large face model was used, which implies that a larger HoG patch around each pixel was extracted.

### 3.3.1 Better jawline landmarks through gradient magnitude image

For the jawline case, because every point of the 17 is a knot and needs to be part of the spline and some of them are not perfectly positioned due to the initial detection, some images worked better than others. Therefore, the landmark positions needed to be improved. This was accomplished by selecting each landmark, looking at its neighbourhood in the gradient magnitude image, preferably in a rectangular area determined by the gradient direction and selecting the point where the gradient magnitude value was maximal. This point became the landmark's new position. The new spline obtained with these landmarks will be referred to as the corrected spline. For most of the test images, results improved, but sometimes it was the case that the search neighbourhood was too large and the landmark "jumped" off the face onto the person's shoulders for example, where there was a higher gradient magnitude. The opposite can also happen: some of the initial landmarks can be so far away from their intended positions, that the search neighbourhood is too small to correct them. So, the size of the neighbourhood was chosen as to minimize both these errors. Also, the jawline landmarks close to the ear are not eligible for this improvement, because the gradient magnitude has large variations around

the ear, so it was best to start modifying landmarks below the ears.

### 3.3.2 Better eyebrow landmarks through intensity image

The eyebrows were less of a problem to fix, because the initial detections were almost correct on all the test images. They were spatially higher than needed, so a correction was done by searching downward from every landmark for the lowest value in the grayscale image, corresponding to dark pixels. Then, the landmark was moved to that location, because for the vast majority of human faces, the forehead is brighter than the eyebrows.

For testing, a database of 120 frontal faces annotated with ground truth landmark positions was used [15]. Unfortunately, the available annotations differed from the ones obtained by using the algorithm described in [21] and finding a linear mapping between them proved unfruitful. Therefore, the images were manually annotated with the 68 landmarks that fit the used detection method. Error measurements were done by computing the distance between the real landmarks and the detected ones: $d_i = \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}$, for the $i$-th landmark. The eyebrow errors are given in table 3.1. Two qualitative comparable images are also given in figure 3.6, before and after updating the landmarks.

| Eyebrow landmark | Initial detection error | Error after correction |
|---|---|---|
| 1 | 14,84 | 8,02 |
| 2 | 15,47 | 8,43 |
| 3 | 15,48 | 8,05 |
| 4 | 13,94 | 7,33 |
| 5 | 12,40 | 6,71 |
| 6 | 15,91 | 8,52 |
| 7 | 15,52 | 8,04 |
| 8 | 15,52 | 8,24 |
| 9 | 19,06 | 11,41 |
| 10 | 16,50 | 10,03 |

**Table 3.1.** Errors for initial detections and corrected landmarks for the eyebrows, tested on the IMM database [15].

### 3.3.3 Better jawline landmarks with outlier detection

Further improvement can be obtained by checking for outliers within the landmarks, points that break the smoothness of the spline by being too far inside or outside the jawline area. Landmarks were randomly removed from the corrected spline, a new spline was obtained and then a distance measure was computed between the removed point and this new spline. If this distance was higher than a threshold, the removed landmark point was replaced with the closest point on the new spline. Another way of scoring the new spline is to sum up gradient magnitude values along its path and choose the spline with the highest score as the correct one.

### 3.3.4 Better eye landmarks through corner detection

The eye landmarks are among the hardest to improve as they are spread across an area with a lot of overall texture and even a small deviation from the ground truth can lead to visible misshaping of the ideally oval curve that connects the 6 eye landmark coordinates. These are laid out as: one at the left extremity, one at the right extremity, 2 just above the eye evenly spread between the extremities and 2 just below the eye on the same vertical line as the ones above. Most of the errors occur when the left and right extremities are detected

**Figure 3.6.** Jawline and eyebrow correction described in sections 3.3.1, 3.3.2. Left: initial detections, middle: search directions for better landmarks, right: improved locations.

slightly inside the eye. To improve this, corner information within the eye area can be used. If the wrong landmarks were replaced with the correct corners, then the results would be more visually pleasing. The first corner detection method tested is the one proposed by Harris and Stephens [7], which is also the default implementation in MATLAB. The initial eye corner landmarks were replaced with the closest detected corner in the grayscale image in terms of Euclidean distance. This failed to achieve the goal of putting the landmarks exactly on the ground truth corners and the results were no better than the input. In an attempt to make the desired corners more visible to the detector, the image was sharpened by using an unsharp filter $H = \begin{pmatrix} -0.166 & -0.666 & -0.166 \\ -0.666 & 4.333 & -0.666 \\ -0.166 & -0.666 & -0.166 \end{pmatrix}$ and convolving the image with H. Even though the images became a lot clearer, the corner detections around the region of interest didn't visibly change.

After the poor results obtained by using the default Harris detector, a new method was tested, the FAST corner detection algorithm proposed by Edward Rosten [13, 14]. It allows the setting of a threshold as well as non-maximal suppression, similar in effect to the one applied by the Canny edge detector [1]. Tweaking of the threshold led to a value of 20 for the minimum score needed for a point to be considered a corner. Detections around the eyes are presented in figure 3.7. All the processing was done with the sharpened image as input. Visual results on a few images suggested that this approach would yield better landmark locations, but after processing the entire set of 120 files, the mean error between the ground truth and the detections rose. Therefore, it was concluded that eye landmark correction should be used on an image to image basis. If the results are not satisfactory, the photo can be taken again, or the correction can be bypassed.

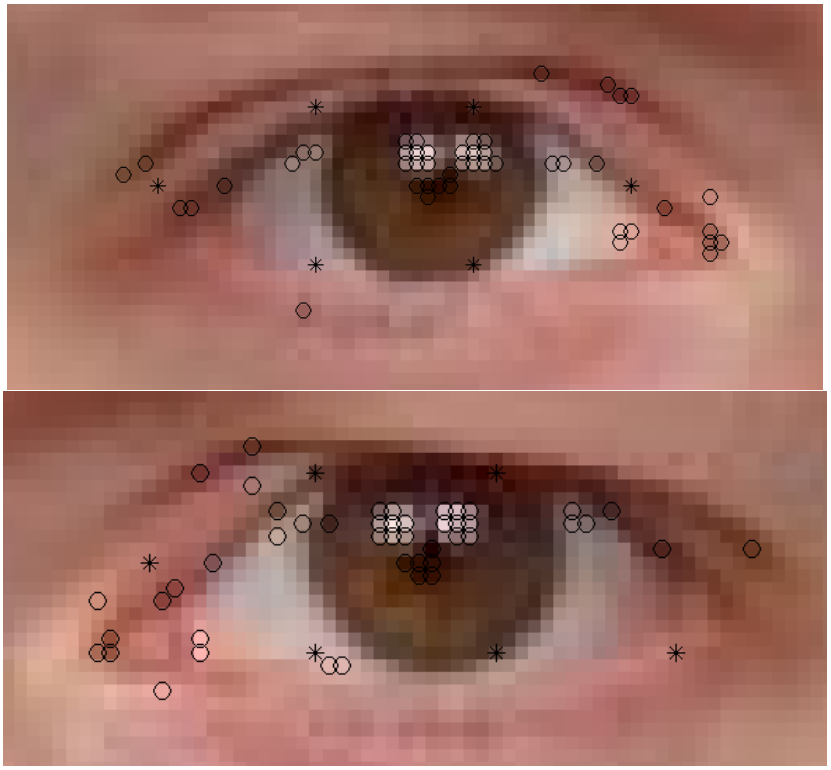

**Figure 3.7.** FAST corner detections for the left and right eyes: stars represent initial landmarks, circles represent the detected corners. The left-most and right-most landmarks are replaced with the closest detected corner, in hopes that it was positioned correctly exactly on the extremity of the eye [13, 14].

| Landmark | Initial detection error | Corrected error |
|---|---|---|
| Right eye corner 1 | 13,347 | 19,341 |
| Right eye corner 2 | 5,895 | 6,167 |
| Left eye corner 1 | 8,112 | 13,608 |
| Left eye corner 2 | 10,075 | 10,241 |

**Table 3.2.** Errors for initial detections and corrected landmarks for the eyes, tested on the IMM database [15].

### 3.3.5 Regularized splines for the jawline

Instead of trying to fit a natural cubic spline to the jawline, a regularized spline fit was attempted, which allows the setting of a smoothing parameter between 0 and 1 that penalizes jaggedness. For the 17 points on the jawline, $x \in \{1..17\}$ and $y = (x_i, y_i), i = 1..17$, a regularized spline $f$ minimizes

$$P \sum_{j=1}^{17} w(j)|y(j) - f(x(j))|^2 + (1-p) \int \lambda(t)|D^2 f(t)|^2 dt$$

The default value for the weights is 1 and for $p$ a good estimate is around $1/(1 + \frac{h^3}{6})$, where $h$ is the average spacing between landmarks. This regularized spline was fitted to the corrected jawline landmarks, as they have been observed to be more accurate than the original detections. Because the purpose was smoothing, the points need not be on the spline, so, in order to obtain new landmarks, the shortest distances from the corrected landmarks and the spline were calculated. The landmarks were moved to the closest point on the spline. The results for both of the jawline corrections are presented in table 3.4. The mean deviation from the ground

truth for all the jawline landmarks is 11.9251 for the gradient magnitude correction and 11.6049 for the regularized spline correction. Even though the numbers are very similar, the smoothing spline delivers a better qualitative representation of the human face curvature, so it will be considered the best spline interpolation method up to this point.

### 3.3.6 Principal component analysis

Landmarks can be used to train a principal component model that may improve their position. In order for this to work, a set of ground-truth points must be created to train the model on. Let us denote the number of landmark pairs by $p$, in this particular case $p = 68$ and the number of images $n$, with $n = 120$. The total number of coordinates (x,y) will be $2p$ and will serve as a feature vector.

The goal of PCA is to obtain a new coordinate system in $2p$ dimensional space where the orthogonal directions capture the largest variance of the data. The initial data is a matrix of size $(2p, n)$, denoted $D$, where each column holds the landmark coordinates for an image. First, the landmarks need to be aligned to a standard shape. This is done by finding a similarity transform between a chosen landmark set-up (the ground-truth in image 1 for instance) and all the other arrangements. Then, the landmarks are warped to the desired shape by means of scaling, translation and rotation and a new matrix $D$ is obtained, containing the aligned landmarks.

The benefit of using such a matrix notation is that the PCA computation is nothing more than a singular value decomposition of the normalized $D$ matrix. Normalization is done by subtracting the mean feature vector $\mu$ from each column and dividing by the standard deviation $\sigma$ along all columns. Then, the singular value decomposition of normalized $D$ is $D = USV^T$. The columns of $U$ are the principal components and the diagonal elements of $S$ are the singular values. The principal components are sorted in decreasing order of their singular value, meaning that the component with the highest value represents the direction of largest variance in $2p$-dimensional space. By selecting only a number $n_c$ of components to rebuild landmarks, we can obtain effects similar to those of a smoothing spline.

When a novel set of landmarks is presented, the following steps take place:

1. The coordinates are warped to the standard shape by applying the appropriate translation, rotation and scaling and used to create the $2p$-dimensional feature vector.
2. $D$ is normalized by subtracting $\mu$ and dividing by $\sigma$: $D_n = \frac{D-\mu}{\sigma}$.
3. $D_n$ is projected onto every principal component: $proj = U^T * D_n$.
4. The estimated feature vector $\hat{D}$ is computed: $\hat{D} = U(1 : n_c) * D_n(1 : n_c)$, where $1 : n_c$ signifies the first $n_c$ columns.
5. The data is transformed back by multiplication with $\sigma$, adding $\mu$ and then reversing the rotation, translation and scaling.

The result is a new set of landmarks that is the best approximation of the original detections by using a linear combination of the first $n_c$ principal components. The model was trained on 110 images and tested on the remaining 10. The number of components required for a precision of 99% was 57, out of the total of 136 possibilities. The results were good, but the training and test images were quite similar so these observations are more of an indicator that the algorithm works as intended, than an actual goodness measure. A few images with ground truth annotations and estimated PCA landmarks are presented in figure 3.8. The error measurement used was the squared sum of distances between the correct values and the PCA estimations. The errors are presented in table 3.3.


**Figure 3.8.** PCA estimates of landmark localization.

| Image | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Error | 0,048 | 0,045 | 0,041 | 0,044 | 0,044 | 0,051 | 0,049 | 0,047 | 0,043 | 0,060 |

**Table 3.3.** PCA estimation errors for the 10 test images.

An interesting experiment is to check what each principal component models. This can be achieved through reconstruction of a landmark configuration by adding a multiple of a principal component to the mean positions. So, if we wanted to observe the impact of the $i$-th principal component, we would reconstruct the following landmarks: $\hat{D} = \mu + U * h_i$, where $U$'s columns are the principal components and $h_i = [0, 0, .., 0, 1, 0, .., 0]^T$, where the 1 is on the $i$-th position. Figure 3.9 shows how the principal components change the shape of the mean face. It appears that the components capture face rotation and size.

The next step is to test how the PCA model works on actual detections, preferably those that visibly diverge from the ground truth. For this purpose, the initial detections obtained with the algorithm described in section 3.2.1 are considered as input and transformed by using the first 35 principal components, as well as the first 50. The expected outcome is to get a set of landmarks that are closer to the ground truth than the initial detections, but this time through the probabilistic method of PCA, instead of tweaking the jawline, eyebrows and eyes individually with various algorithms. The results are presented in table 3.4, alongside the ones corresponding to the previously mentioned approaches of landmark location improvement. The errors obtained on the 120 image dataset are measured as previously stated, by means of the Euclidean distance between the ground truth and the new landmarks.

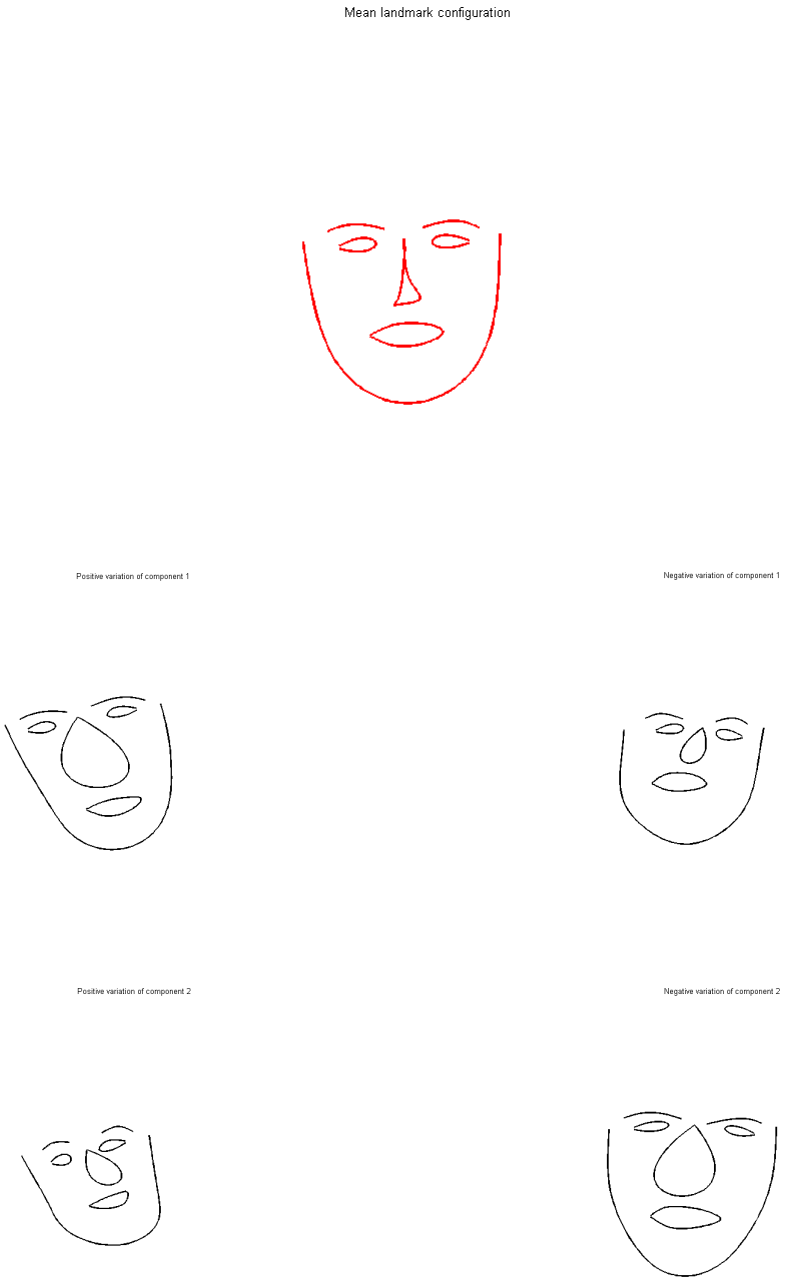


**Figure 3.9.** Top: mean face. Middle: positive variation of principal component 1, negative variation of principal component 1. Bottom: positive variation of principal component 2, negative variation of principal component 2.

| Landmark | Initial detection error | Gradient magnitude correction error | Regularized spline fit error | PCA error 35 components | PCA error 50 components |
|---|---|---|---|---|---|
| 1 | 17,93 | 14,62 | 14,40 | 10,90 | 11,45 |
| 2 | 15,05 | 12,30 | 12,74 | 11,08 | 11,27 |
| 3 | 15,40 | 12,38 | 11,08 | 12,89 | 12,71 |
| 4 | 17,60 | 10,17 | 10,12 | 11,14 | 13,62 |
| 5 | 19,56 | 11,84 | 10,90 | 12,85 | 15,15 |
| 6 | 19,93 | 12,11 | 12,50 | 16,16 | 17,55 |
| 7 | 20,19 | 13,47 | 13,95 | 17,47 | 16,13 |
| 8 | 23,30 | 17,89 | 16,46 | 20,21 | 18,85 |
| 9 | 20,22 | 13,89 | 15,03 | 16,70 | 14,81 |
| 10 | 17,40 | 11,94 | 10,80 | 16,47 | 16,43 |
| 11 | 15,54 | 9,53 | 9,87 | 14,55 | 15,15 |
| 12 | 13,78 | 12,24 | 11,03 | 11,06 | 11,08 |
| 13 | 12,41 | 10,10 | 10,20 | 9,13 | 8,06 |
| 14 | 13,30 | 9,72 | 8,41 | 10,89 | 10,85 |
| 15 | 12,58 | 9,96 | 9,27 | 13,04 | 13,02 |
| 16 | 11,99 | 9,73 | 9,94 | 8,61 | 9,47 |
| 17 | 13,74 | 10,75 | 10,50 | 8,52 | 8,12 |

**Table 3.4.** Errors for initial detections and corrected landmarks for the jawline, tested on the IMM database [15].

The numbers corresponding to the 35 principal components trial are slightly better than the ones for the 50 components case. While the PCA errors are clearly smaller than the ones for the initial detections, it cannot be said that principal component analysis is a better approach than regularized splines. Neither is perfect and it is left to the judgement of the user to decide for a particular case which landmarks are better. By viewing the results from a small set of images (around 10), the eyes and mouth appear to be better represented by a PCA model, while the jawline's shape is better captured with the regularized spline algorithm. Figure 3.10 displays a visual qualitative comparison between the two tactics by computing curves through their respective landmarks.

## 3.4 Results and discussion

The chapter on facial landmarks can be summarized by a 3 step algorithm: initial landmarks are detected, they are corrected by using various methods, then splines are interpolated through the corrected points. The hope is that the resulting curves will hold enough information to properly depict a human face and distinguish it from any others. Unfortunately, this is not the case, as seen in figure 3.11. The nose lacks detail, due to the nature of the spline interpolation and the absence of a subset of landmarks which could give better results. Besides the lack of differentiating features, the line drawings are "ugly" at best, when they are not seen superimposed onto the actual image. In conclusion, landmark localization should not be used on its own for obtaining the final drawing, but as a helper function to a proper edge extractor.

## 3.5 Edge extraction with landmark localization

The landmark detection method is useful for obtaining a minimalistic representation of the human face but it is not enough to capture the traits of each individual. By using only landmark

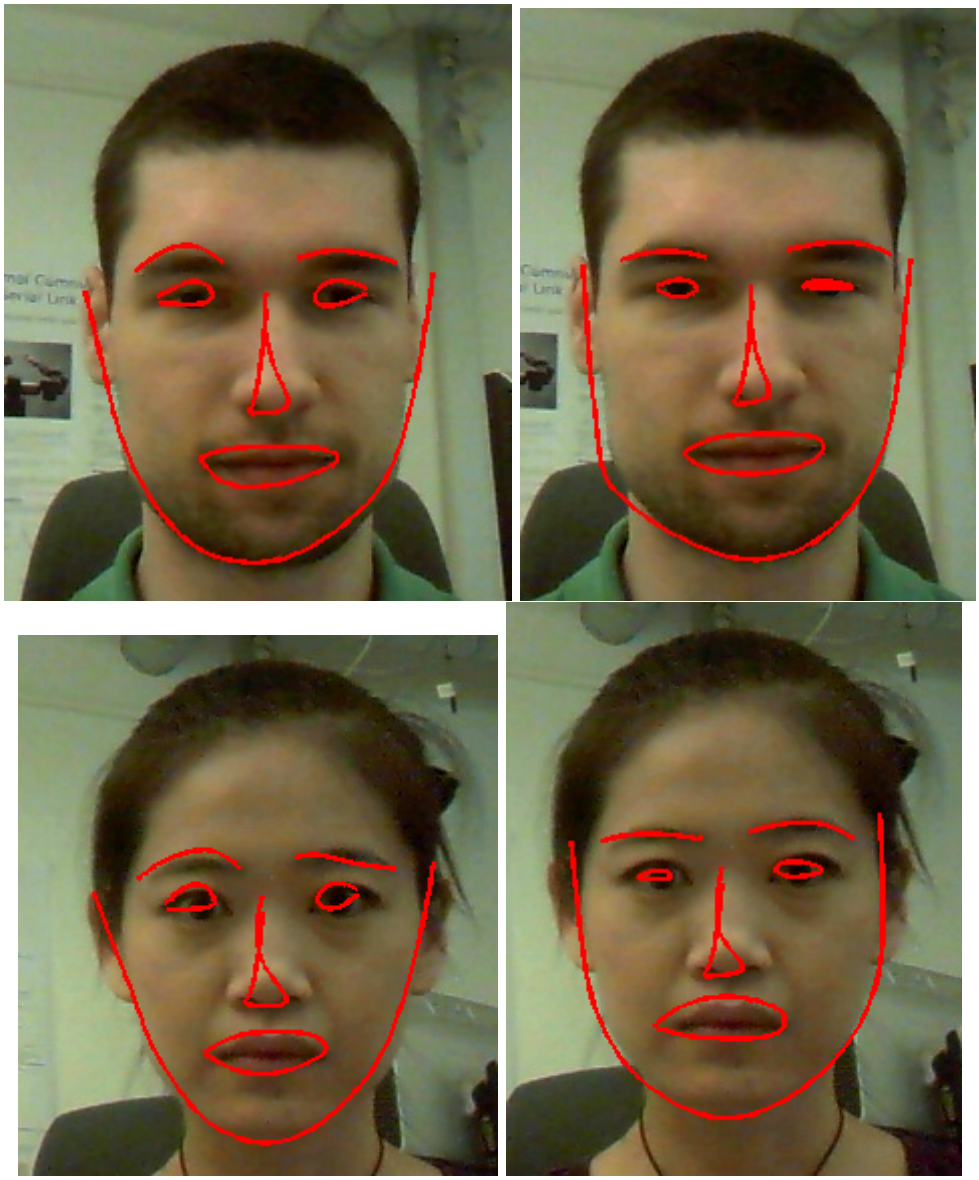

**Figure 3.10.** Curves plotted through the landmarks obtained through the PCA approach (left) and through the regularized spline approach (right).

positions, the curves obtained for two different people could end up being very similar. For this reason, a return to the basic edge extractor is attempted, but this time with the knowledge of 68 landmark locations.

If the Canny detector is used, the automatic high threshold is replaced by a value 2.5 times larger and the low threshold is set at 60% of it. For both the Canny and the Gabor filters algorithms, connected components are extracted from the binary edge image by using an 8-connected neighbourhood, meaning that if a non-zero pixel has a non-zero neighbour above, below or at any of 4 corners, it is considered connected to that neighbour. A connected component is a structure containing non-zero pixels that can be traversed from any point to any other by only visiting non-zero neighbouring values. Knowing these connected components and their size allows a trimming of small, most-likely unwanted edges. A size threshold of 20 pixels is set and all the components shorter than this value are discarded.

So far, some clutter was removed and the focus was placed on long, strong edges but the problem of background now comes into play. If the captured image is zoomed out and only a small portion contains the face, there are 3 available options: the user crops the desired region

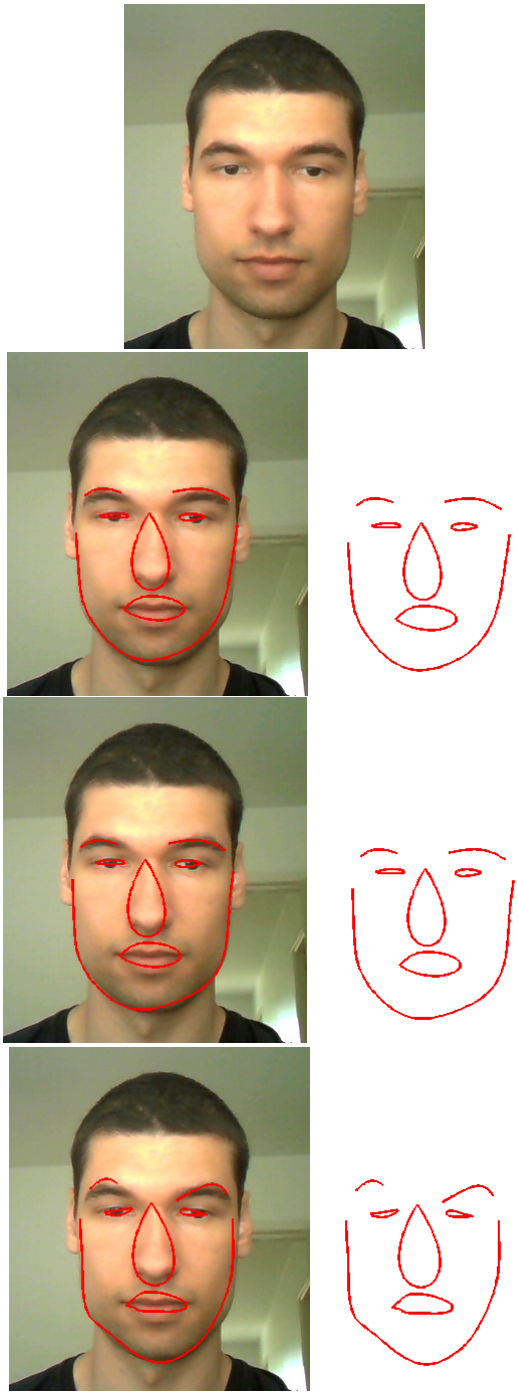

**Figure 3.11.** Cubic spline interpolation for all the facial landmarks. Second row: splines through initial landmark detections, third row: splines through corrected landmarks, fourth row: splines through PCA landmarks.

**Figure 3.12.** Bounding boxes obtained from the corrected landmark locations, which limit the valid edges.

of interest manually, an automatic face detection such as the one proposed by Viola and Jones [18] is performed, or the landmark positions are used. The third option has the added advantage of also removing some of the inner face false edges that may have been left after the connected component trimming.

The landmarks are used to create a binary image, called BW, containing non-zero elements at each of their coordinates. Next, a Euclidean distance transform is computed. For each pixel in BW, the distance transform assigns a number that is the distance between that pixel and the nearest non-zero pixel in BW. Then, by using the distance transform matrix, the closest landmark to each connected component can be determined. If this distance is larger than a threshold set at 0.05% of the original image's area measured in square pixels, then the connected component is discarded. This threshold shouldn't be too low, as this would lead to losing important inner face components.

A synonymous tactic is to place boxes on each landmark. The height and width should be proportional to the image dimensions, but for most of the pictures taken via the available integrated camera, the rectangles were 10 by 10 pixels. Because the jawline landmarks only go up to the ears, the top of the head would not be represented. This was dealt with by fitting an ellipse containing 40 points to the whole head. These points are treated as landmarks and have bounding boxes placed on them. The final valid face area is shown in figure 3.12.

An alternative is to take into account the fact that what needs to be drawn lies to the right of the left ear, the left of the right ear and above the bottom of the jaw, locations which are given by the landmarks. So, if only connected components which fulfil these 3 requirements are displayed, even more background is erased. This approach is equivalent to creating a bounding box around the face and only considering points which fall within the created rectangle.

Finally, to simplify the drawing process, a skeleton extraction is performed on the edge image, which has the effect of thinning all the lines, making them faster to draw.

Results obtained from this approach are both simple enough to be drawn in a reasonable amount of time and complex enough to differentiate between faces. They are qualitatively considerably better than both the initial cluttered Canny edges and the overly simplistic landmark cubic splines as seen in figure 3.13. One observation worth noting is that in some cases the jawline or other parts of the face might not be complete. A solution for this shortcoming could be to use the before mentioned spline interpolations through landmarks to replace the missing edges, but this may also lead to sketches that look artificial.

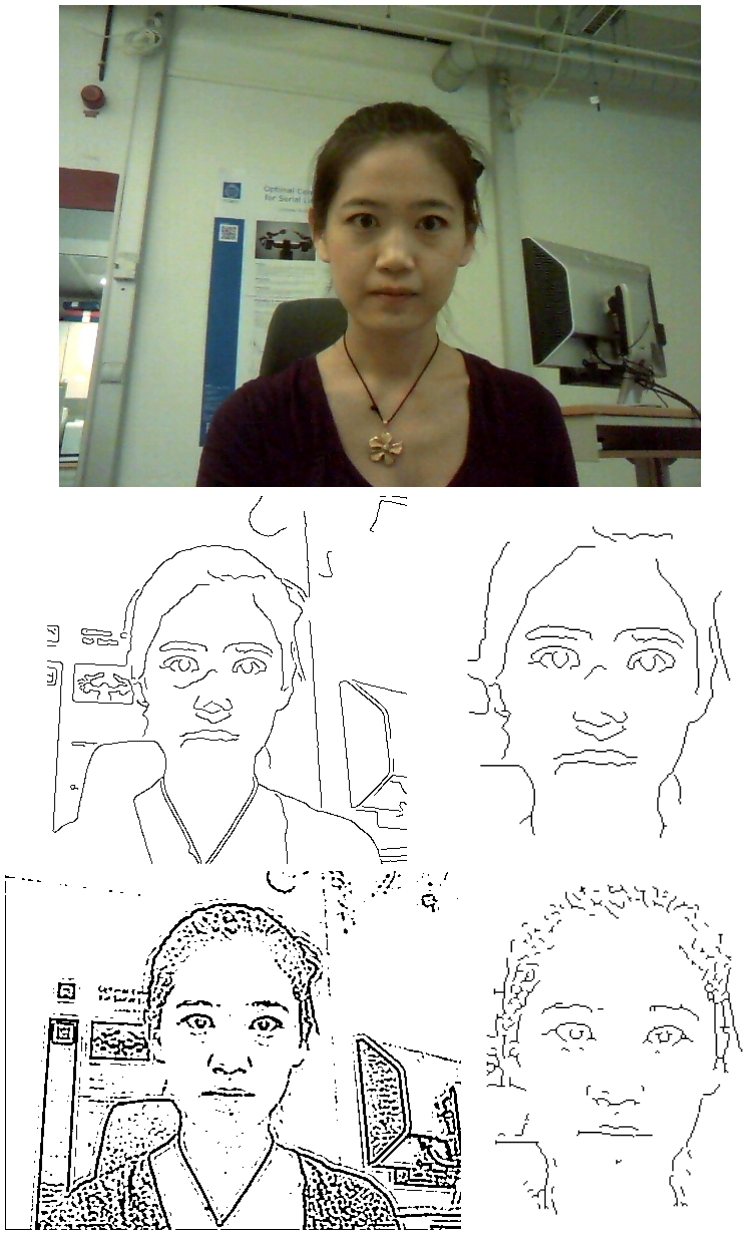

**Figure 3.13.** Upper left: original Canny edges. Upper right: trimmed Canny edges. Lower left: original Tresset edges. Lower right: trimmed Tresset edges.

## 3.6 Conclusions on obtaining line drawings

This section will present and discuss qualitative results for all the above mentioned techniques of obtaining line drawings: simple edge extractors, splines through landmarks, splines through improved landmarks and finally, edge extraction combined with landmark detection.

The iterative process through which the final trajectories are obtained has been thoroughly discussed in the previous sections and is briefly restated:

- An image containing a human face is used as input and scanned for facial landmark locations.
- The initial landmark detections are improved through various algorithms and the user may be asked to select the best landmarks.
- Edges are extracted from the image and are trimmed to eliminate small ones, due to noise. In the case of the Canny edge detector, the hysteresis thresholds are manually selected.
- The best landmarks available are used to remove the edges outside the face.

Although it is now obvious that the best trajectories are obtained with a combination of edge and landmark detection, this was not a clear target from the start. With each new method came downsides that were gradually removed. This improvement process is depicted in figure 3.14, where the evolution of line drawings is exemplified. The main starting points are the edges extracted by the Canny method [1] and the Gabor filter method proposed by Tresset & Leymarie [17].

In figure 3.14 and in many other situations, the Gabor filters method seems to produce better looking results. This however may not always be the case and therefore the Canny edge extractor remains a possible initializer for any novel images.

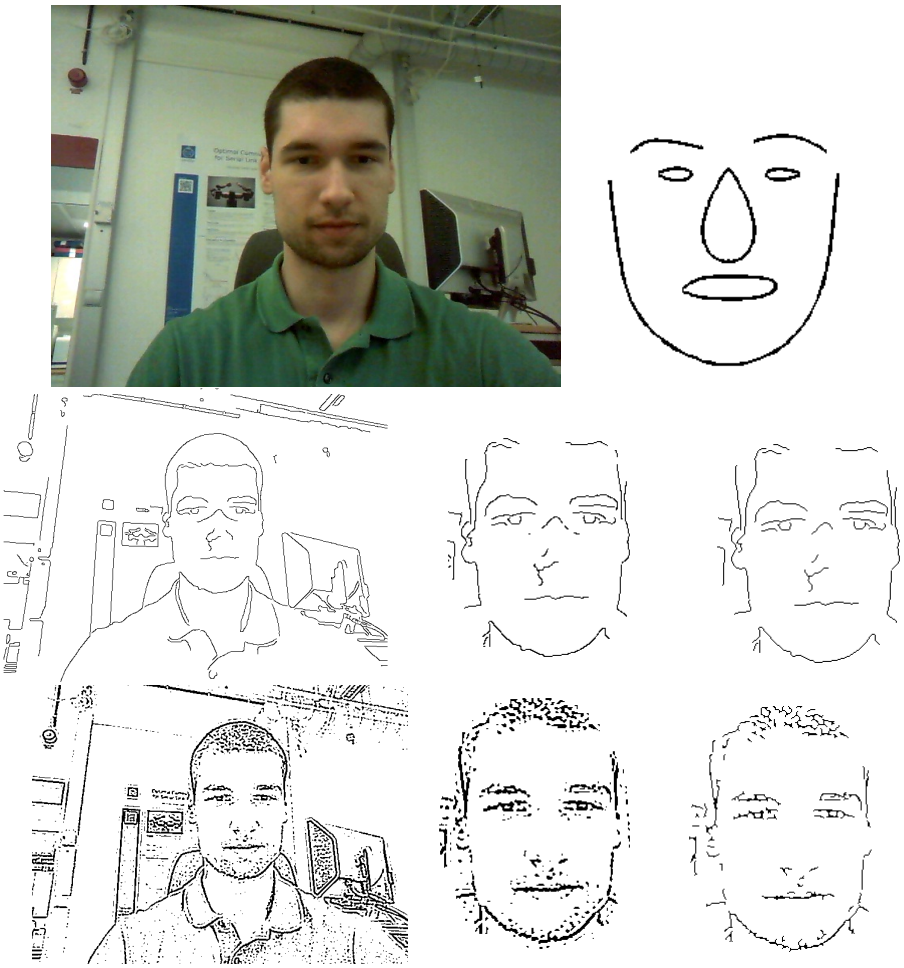

**Figure 3.14.** Line drawings obtained throughout the improvement process. Top row: original image and spline interpolation through best landmarks. Middle row: initial Canny edges, trimmed Canny edges by using landmarks, trimmed Canny edges by using minimum connected component length and thinning. Bottom row: initial Tresset edges, trimmed Tresset edges by using landmarks, trimmed Tresset edges by using minimum connected component length and thinning.

# Chapter 4

# Sketching the trajectories

This chapter presents the mechanical details of the hardware used for sketching portraits and explains how the image processing part of extracting trajectories interacts with the physical world in which they are drawn.

## 4.1 Dumbo

Dumbo is the name given to the semi-anthropomorphic dual arm robot in the CAS/CVAP laboratory at the Royal Institute of Technology. Each of the two arms has 7 degrees of freedom and is made up of Schunk rotary modules that send velocity commands via a CAN bus. The joints have encoders mounted on them from which angle measurements are read. For the purpose of drawing on a 2D plane the left arm is used, as it is the only one with a gripper to hold the drawing instrument. Many of the available degrees of freedom are not used due to the mechanical simplicity of the task. The gripper should remain parallel to the paper and move according to the given trajectories, while the drawing instrument stays in a vertical position.

## 4.2 ROS

ROS is an acronym for Robot Operating System, the open-source software used not only for interacting with the physical arm, but also running simulations prior to the actual application. ROS provides among others, hardware abstraction, device drivers, libraries, visualizers, and message-passing [12]. A deep understanding isn't needed for a task as simple as moving an effector over a plane but a few details need to be given. The version used was ROS Fuerte and the platform on which it was installed was Ubuntu 11.10. The essential components of ROS are listed below.

**Figure 4.1.** Dumbo, the dual arm robot being idle on the left and drawing on the right.

- Nodes are executables that can communicate with each other. They are part of a ROS package and are written in C++ (Python can also be used). After building the package, the nodes can be run by issuing a "rosrun package node" command. Basically, all the code needed for moving the arm will be stored in a ROS node, defined in a cpp file.
- Topics act as buses over which nodes exchange messages. If a node is interested in a topic, it subscribes to it and receives information from nodes that publish data on that topic. The transport is TCP/IP based.
- Messages are the way nodes communicate. They are published on a topic.
- Services provide an alternative to the publish/subscribe model, adding a request/reply framework. A providing ROS node offers a service under a string name, and a client calls the service by sending the request message and waiting for the reply. [12]

Apart from these basic aspects, there are many packages that add functionality. Two of the more important packages used are rviz, for visualizing the robot and the world, and TF which allows the user to keep track of coordinate frames and compute transforms between them. Details such as how the robot parameters are stored and how configuration files are written for simulations and practical implementations are beyond the scope of this paper and can be found at www.ros.org, along with many tutorials that proved useful for learning the basics needed for this project.

## 4.3 Drawing in a simulated environment using ROS

ROS offers powerful visualization tools useful for simulation. Dumbo is defined in ROS by using the Unified Robot Description Format (URDF) and can easily be imported into rviz. Before any manipulation of the robot can take place, it must be positioned into a start state, specified by a number of goal joint positions given in radians, with values between $-\pi$ and $\pi$. Since the left arm is the one of interest, 7 constraints must be given, corresponding to each link in the arm. To determine these constraint goals, the arm is manually moved into an acceptable position and then the "joint_states" topic is read. The chosen starting pose is shown in figure 4.2.

The obvious way of sketching a face by using a robotic arm is to have the arm "connect the dots", where the dots are Cartesian coordinates given in the appropriate frame. Since the robot has quite a few moveable parts, each of these has its own view of the world, or frame. Care needs to be taken when specifying locations, so as to not use the wrong frame of reference. The arm needs to successively move to a series of pose goals given by the trajectories. First, the appropriate joint must be selected. In the case of Dumbo, it will be the left arm wrist joint. Then, the $[x, y, z]$ goal coordinates must be specified, along with the $[x, y, z, w]$ orientations, leading to 7 values. The above will be written in a loop over all the points that need to be connected. As long as these points are dense enough, the trajectory should be linear between adjacent coordinates.

Unfortunately, it can happen that a pose goal cannot be reached, due to physical limitations. If this happens, an inverse kinematics error occurs, meaning that ROS cannot calculate the joint movement needed to reach the Cartesian coordinate goal. To explore the causes of this error, an interactive "warehouse" environment was used, where the user can point and click to define goals for any arm link.

It was observed that for fairly large displacements of the end effector, the goal coordinates do not change so drastically from the initial coordinates. This is because all the values used in ROS are given in meters, while the coordinates obtained from the image processing phase are integer pixel locations. Therefore, when the trajectories are passed from the image processing block to the robotic arm, they should be normalized by a value that converts pixels to meters.

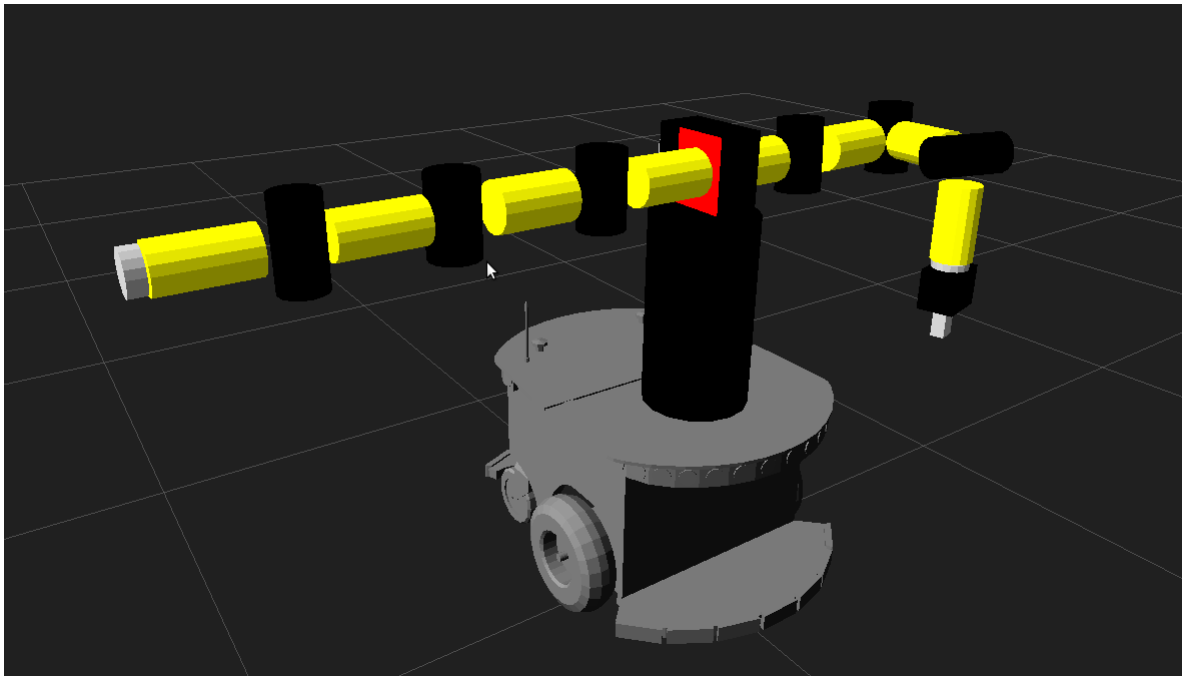

**Figure 4.2.** Initial pose, seen in rviz.

Another aspect to consider is that a pose goal must be specified by 7 values, 3 for the Cartesian position and 4 for orientations, but only Cartesian coordinates are of interest. So, the orientations are kept constant and are equal to the ones corresponding to the initial pose.

If no errors occur, the process of passing trajectories and moving the arm through them is straightforward. Input is received from the image processing block in the form of connected components. When passing a component, the points must be given in the correct order. If they are left unaltered, the result would be the line drawing shown in the first snapshot of figure 4.3. This happens because MATLAB's connected component extractor outputs randomly ordered points. The image is still decipherable as a human face, but it could be a lot better. The correction was straightforward, as MATLAB offers a function that traces the exterior contour of objects in a black and white image. By applying this function to a black background image with the connected components superimposed as white pixels, the needed ordering is obtained. This new ordering yields much better results as seen in the second snapshot of figure 4.3.

The solution to the difference in coordinate frames between MATLAB and ROS is a linear mapping of 2D points from one plane to another. The arm's physical boundaries determine the size of the drawing area, which was experimentally found to spread from approximately ($x_1 = -0.5221, y_1 = 0.476$) as the top left corner to ($x_2 = -0.113, y_2 = 0.264$) for the bottom right corner. These values are given in meters as seen from the arm base link reference frame, where the left arm extends towards the negative x axis, which explains the lower than 0 value of $x$. The initial integer coordinate system in which the points are given spreads from ($x'_1 = 1, y'_1 = 1$) to ($x'_2 = size_x, y'_2 = size_y$), where $size_x$ and $size_y$ are the number of rows and columns of the image. A linear mapping between these points can be written as:

$$\begin{aligned} x'_1 &= ax_1 + by_1 + c \\ y'_1 &= -ay_1 + bx_1 + d \\ x'_2 &= ax_2 + by_2 + c \\ y'_2 &= -ay_2 + bx_2 + d \end{aligned} \tag{4.1}$$

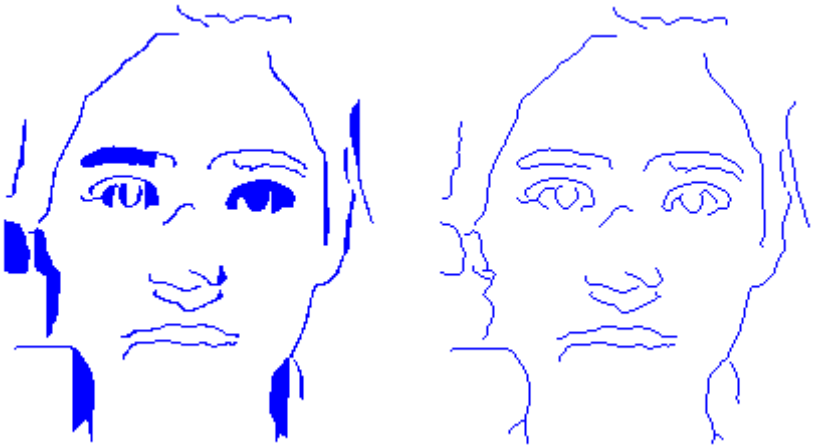

**Figure 4.3.** Line drawing obtained from original and improved point ordering in the connected components.

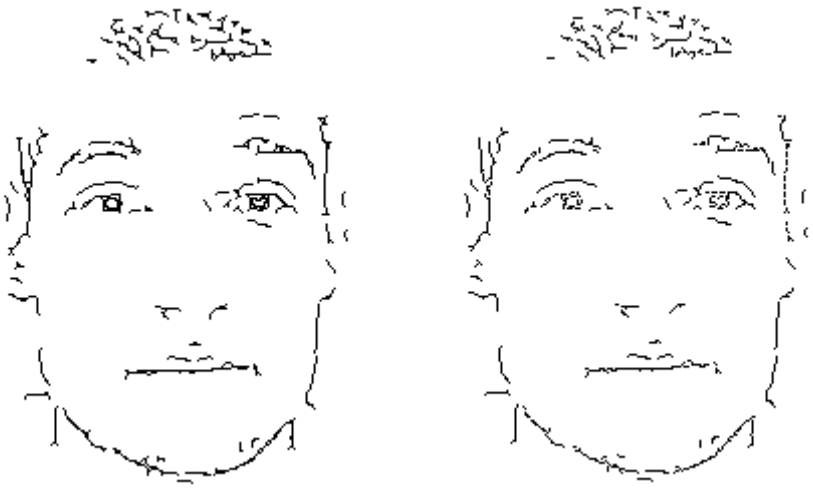

**Figure 4.4.** Comparison between sketches obtained by connecting all the points and by only connecting points that are far enough from each other. First image takes around 1 hour to draw, while second image requires 30 minutes.

In matrix notation, $\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ -y_1 & x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ -y_2 & x_2 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$, or $u = Mv$. Therefore, $v = M^{-1}u$. After determining $a, b, c, d$, they are plugged in equations (4.1) for every initial pixel coordinate.

### 4.3.1 Point trimming

Because of the large number of points that need to be connected, it may prove too time consuming to use all of them. Therefore, each non-zero pixel of the connected components is checked against the next one. If the distance between them is smaller than a threshold, then the next point is skipped. This trimming improves the speed by 100% and the toll it takes on the sketch quality is small enough to make its application worthwhile. A comparative view of the untrimmed and trimmed line drawings is shown in figure 4.4.

## 4.4 Drawing on paper

First attempts to sketch human faces were unfruitful for a number of reasons. As expected, in the case of untrimmed line drawings, the arm had to move through about 2500 coordinates. With an average requirement of 1.5 seconds to connect 2 adjacent points, drawing a face would take approximately one hour, far too long in terms of effectiveness. By halving this time, the process became more feasible.

Another problem encountered was caused by the inaccuracies of the path planning and the irregularities of the drawing plane, as even a 1 mm error in the end-effector position can have a large influence. This is seen in one of two ways: either the pen/pencil is pushed too hard into the paper, making it stick in position, not being able to move the small distances required for drawing, or, it doesn't touch the paper at all. These errors are attributed to the robot's tolerance with respect to the target pose accuracy, which cannot be 0, and to the fact that the drawing plane might have irregularities and varies compared to Dumbo's base frame axes. In order to fix these issues, one can construct a drawing instrument with a flexible tip, that retracts when pressure is applied and goes back when it is released. Another option is to use a brush, as it offers a larger tolerance for errors, but it would require constant dabbing for paint, either manually, or by the robotic arm.

The approach that worked was to use a spring mechanism on a regular ballpoint pen, which gives the tip a leeway of around 0.5 cm. Also, there has to be as little space as possible between the gripper and the paper, so that the pen is firmly grasped. It has been observed that the closer the points are, the better the connecting line becomes. This, unfortunately, clashes with the idea of using as few points as possible. A compromise was reached by using around 600 points for one face.

Dumbo's sketches are presented in figure 4.5 and a video of the drawing process can be viewed at http://www.youtube.com/watch?v=CxW33YX9kF4 [16]. The zigzag pattern of the lines is caused by the fact that adjacent points are not connected by a perfectly straight line. Also, some lines are doubled because of slight errors in positioning. These problems are caused by hardware limitations.

Other artefacts can be observed, which are created when the pen moves from one connected component to another. Instead of going straight up, moving horizontally to a new point and then going straight down, it moves both on the vertical and horizontal planes simultaneously to reach the point above the new connected component's starting position. This causes a trail of ink at the beginning of the movement, clearly visible in the right sketch's jawline and hair areas. The problem may be overcome by using a more complex algorithm which would impose a path for moving from point A to point B, instead of leaving the planning to ROS, but inexperience in the field of robotics left this adjustment to future work.

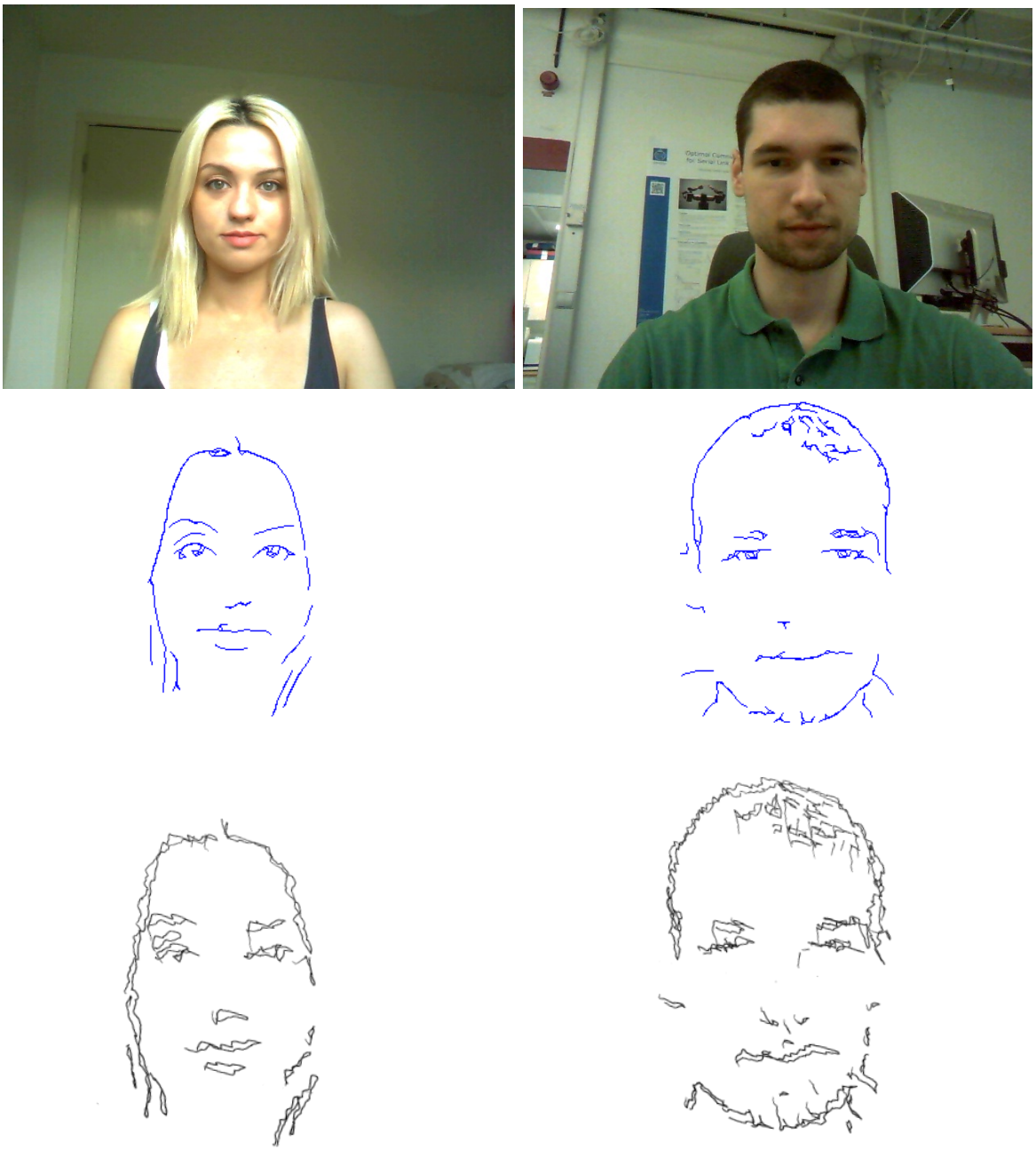

**Figure 4.5.** Dumbo's drawings compared to the MATLAB line drawings. Left sketch used 250 points, right sketch used 600. Drawing times were around 20 minutes. The right line drawing is simpler than the one presented in figure 4.4 because a higher threshold on the connected component length was imposed, in order to decrease drawing time.

# Chapter 5

# Conclusions and future work

This project set out to explore ways of extracting human facial features, convert them to a compact line drawing format and draw them automatically using robotics. Although the sheer number of possible approaches for every part of the process left some aspects untouched in the thesis, a series of feasible algorithms were presented and discussed, yielding good-looking results.

It was observed that a good starting block for the task is an edge image obtained by using Gabor filters at 8 orientations applied to the image at its original scale. For the most part, these edges are better than the Canny ones [1], with respect to human face appearance. Credit goes to Tresset and Leymarie [17] and their work with Paul, for pointing out the usefulness of Gabor filters in face depiction.

Although the results up to this point looked nice on screen, they were far too complex to be drawn in a decent amount of time. Compression was needed and a first step towards a more compact representation was incorporating facial landmarks. Thanks to the algorithm devised by Zhu and Ramanan [21], salient areas of the face, such as the jawline, eyes, nose and eyebrows could be very precisely detected. This knowledge helped trim the edge image by only considering points that lied close to the landmarks, thus removing the background and the inner face edges most likely caused by noise. While on the subject of landmarks, an attempt was made to create splines through the detections, in hopes that the results would resemble the original face. This was not the case and splines remained only a way to improve the initial detections' locations.

The final step before sending the results to the drawing phase was a connected component extraction. The edges are represented by a black and white image of 0s and 1s, where 1 signifies the presence of an edge. Connected components are "islands" of neighbouring edge pixels, separated from one another by background. By checking their size and eliminating very small ones, the final image became less cluttered and easier to draw. To make the edges as thin as possible, skeletonization was performed, which is a morphological operation that thins blobs in a black and white image. Finally, to more or less halve the time required for drawing, edge pixels are eliminated according to the distance to their neighbours. If the distance is very small, such that a line between pixel 1 and 3 would look similar to two lines between pixel 1-2 and 2-3, then pixel 2 is eliminated.

The drawing phase is, in theory, quite simple. Given a set of ordered 2-D coordinates, the arm needs to move its end-effector, holding a pen, through them, leaving behind a trail of ink. In practice however, things weren't so simple. The arm's limited precision forbid the use of rigid drawing instruments such as a pencil or a regular pen, which would either get stuck after being pushed too hard into the paper, or not touch the paper at all. This was counteracted by using a ballpoint pen with a rudimentary spring mechanism that permitted the retraction of the drawing tip when pressure was applied. Then, a compromise had to be made between the quality of the drawing and the time requirement, by deciding upon the number of points

to represent a face. The average value was determined to be around 600. After several failed trials, Dumbo successfully drew his first sketches.

There are many more aspects that can be further investigated in the future:

- Different edge extractors can be used, that may capture human faces better than the Canny or Gabor filters method.
- The landmark correction can be performed differently, perhaps using a more global approach.
- The drawing part might benefit from a frame transformation between the table frame and the robot's base frame. These were considered to have the same vertical axis, which is a fair approximation, but it is obviously not perfect. If points were collected off the drawing plane and the condition of co-planarity enforced on them, then a transformation between the base frame and the drawing plane can be calculated. Then, every point that needs to be drawn can be projected onto the actual table plane.
- Better drawing techniques can be applied, more complicated than simply linearly connecting adjacent points, which could also remove the artefacts caused by erroneous trails of ink.

# Bibliography


[1] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[2] C. I. Chow, Sexior Member, and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 769–776. ACM, 2002.

[5] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[6] R Gross, I Matthews, J Cohn, T Kanade, and S Baker. The cmu multi-pose, illumination, and expression (multi-pie) face database. Technical report, Technical report, Carnegie Mellon University Robotics Institute. TR-07-08, 2007.

[7] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[8] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460. ACM, 1998.

[9] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[10] Judson P Jones and Larry A Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, 1987.

[11] J. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the'art': A taxonomy of artistic stylization techniques for images and video. 2012.

[12] ROS.org. Ros. `http://www.ros.org/wiki/`. Accessed May 3, 2013.

[13] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (to appear)*, May 2006.

[14] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (to appear)*, October 2008.

[15] M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME – a flexible appearance modelling environment. *IEEE Trans. on Medical Imaging*, 22(10):1319–1331, 2003.

[16] Radu Mihai Pana Talpeanu. Dumbo sketching. `http://www.youtube.com/watch?v=CxW33YX9kF4`.

[17] P.A. Tresset and F. Leymarie. Sketches by paul the robot. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 17–24. Eurographics Association, 2012.

[18] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[19] Wikipedia. Lab color space. `http://en.wikipedia.org/wiki/Lab_color_space`. Accessed December 16, 2012.

[20] H. Winnemöller, S.C. Olsen, and B. Gooch. Real-time video abstraction. In *ACM Transactions On Graphics (TOG)*, volume 25, pages 1221–1226. ACM, 2006.

[21] Xiangxin Zhu and D Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.