Betty: A Portrait Drawing Humanoid Robot Using Torque Feedback and Image-based Visual Servoing

by

Meng Cheng Lau

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science
University of Manitoba
Winnipeg

Thesis advisor

Jacky Baltes and John E. Anderson

Author

Meng Cheng Lau

Betty: A Portrait Drawing Humanoid Robot Using Torque Feedback and Image-based Visual Servoing

Abstract

Integrating computer vision into a robotic system can provide a closed-loop controlled platform that increases the robustness of a robot's motion. This integration is also known as visual servo control or visual servoing. Visual servoing of a robot manipulator in real-time presents complex engineering problems with respect to both control and image processing particularly when we want the robot arm to perform complicated tasks such as portrait drawing. In my research, the implementation of torque feedback control and Image-based Visual Servoing (IBVS) approaches are proposed to improve previous open-loop portrait drawing tasks performed by Betty, a humanoid robot in the Autonomous Agent Lab, University of Manitoba.

The implementations and evaluations of hardware, software and kinematic models are discussed in this document. I examined the problem of estimating ideal edges joining points in a pixel reduction image for an existing point-to-point portrait drawing humanoid robot, Betty. To solve this line drawing problem, two automatic sketch generators are presented. First, a modified Theta-graph, called Furthest Neighbour Theta-graph (FNTG). Second, an extension of the Edge Drawing Lines algorithm (EDLines), called Extended Edge Drawing Lines (eEDLines). The results show that

the number of edges in the resulting drawing is significantly reduced without degrading the detail of the output image.

The other main objective of this research is to propose the extension of the drawing robot project to further develop a robust visual servoing system for Betty to correct any drawing deviation in real-time as a human does. This is achieved by investigating and developing robust feature (lines and shading) extraction approaches for real-time feature tracking of IBVS in combination with adequate torque feedback in the drawing task.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking everyone who has helped me along the way during my years at University of Manitoba. Especially my advisor Jacky Baltes, whose input and perseverance has been instrumental in helping me to pursue my doctoral degree. I greatly appreciate his guidance was instrumental for accomplishing my thesis, as well as his unending patience. I would like to thank John Anderson, who has been my co-advisor for the past several years and was always available when I needed help. Special thanks to the members of my thesis committee, Wai-Keung Fung, David Scuse and Dean McNeill for their encouragement and helpful advice. I also would like to acknowledge University Kebangsaan Malaysia and Ministry of Higher Education Malaysia who supported me financially - this thesis would not have been possible otherwise. Last but not least I thank my wife, Siew-Chiing, for all her love, support, patience and care and of course to our newborn son Jay-Qi, who added extra inspiration to finish my work.

This thesis is dedicated to my family and friends.

Chapter 1

Introduction

1.1 Introduction

This chapter discusses the motivation of my thesis and introduces the portrait drawing robot research that I developed. This chapter also presents an overview of the methods I used to overcome the problems of current approaches to drawing robots.

1.2 Motivation

Recent research on humanoid robots has devoted significant effort to developing humanoid robots that can match human behaviour on high-level tasks that require integration of sensing, physical motion and intelligence. The current state of the art supports diverse applications involving a wide range of industries, including education, health care, household services, military, entertainment, etc.

A portrait drawing robot requires human-specific skills which are challenging tasks for humanoid robots. Recent results in robotics literature have demonstrated a number of portrait drawing robots implementations [16; 36; 63]. For instance, Gommel *et al.* [36] implemented an industrial robotic arm, KUKA, to draw in Cartesian space. Lu *et al.* [63] developed a special purpose robotic arm platform, IRAS, for replicating and creating works of art. More examples are discussed in Chapter 2. However, none of these robotic systems successfully mimics human-like features. In recent years many researchers (e.g., [16; 56; 58; 61; 75; 84]) have tried to develop a robust humanoid robot that could produce pen-and-ink sketches of portraits. This is usually slow due to complicated motion control and complexity of the input images. It requires high quality and expensive force and torque feedback sensors such as force sensing resistors (FSR), strain gauges, or load cells for drawing accuracy. The motivation of my research is to adopt affordable robotics technologies to create human-like artwork with the capability of observational drawing through visual feedback. The goal of this research is for a humanoid robot to mimic the drawing process of human by reproducing its way of drawing based on limited feedback control.

Creating a drawing robot with high performance industrial robots (i.e. accurate and fast) [16; 36; 56; 61; 75; 84] usually involves a high hardware cost. It is a much more significant challenge to develop an efficient robotic system if the cost is also an important factor. Reducing the cost of the system requires optimization of all aspects to retain its flexibility, reliability and performance. During the design and development of Betty, I only used affordable hardware and open source software to address both cost and performance issues. I developed an adaptive real-time kernel

with a PID controller to optimise servo control. The total cost of the hardware is less than USD 3000 [7]. However due to the hardware performance limitations, it is difficult to achieve high mechanical accuracy. An alternative image-based visual servoing and torque feedback control combination is proposed to overcome the lack to accuracy in Betty's drawing task.

The portrait drawing humanoid robot, Betty (see Figure 3.1 and Figure 3.2), uses OpenCV, an open source computer vision library, to perform face recognition and image processing techniques. These techniques compute a line-art portrait that can be mapped to the kinematics of the arm. In the early stage of development, Betty drew the portrait using a point-to-point drawing mechanism which maps the points from a pixel-reduction image consisting of a set of points, P, as shown in Figure 1.1. This research aims to develop a robust visual servoing human portrait drawing system that enables Betty to autonomously draw the portrait by exploring various research areas. In my research, I implemeted a modified Theta-graph, called Furthest Neighbour Theta-graph (FNTG), an extended edge drawing technique (eEDLines), torque feedback control (TFC) and image-based visual servoing (IBVS). These implementations have constructed a good and simple approximation pen-and-ink sketch of the input image by defining edges joining pairs of points in line segments that utilise outlining and shading techniques.

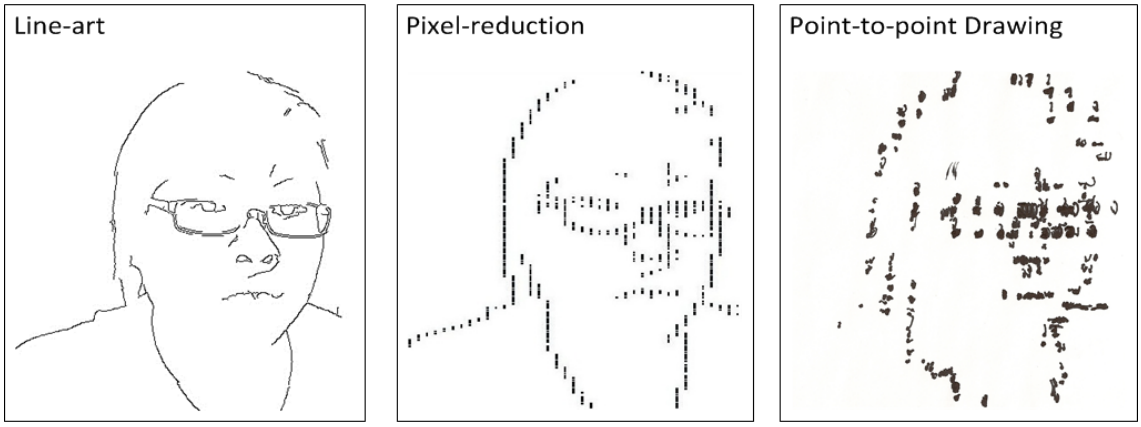


Figure 1.1: ***Left***: Line-art Portrait from Canny edge detection algorithm. ***Middle***: Pixel-reduction to reduce the number of pixel. ***Right***: Actual portrait drew by Betty

1.3 Thesis Overview

Chapter 2 discusses related background of my research. The preliminary hardware and software approaches are described in Chapter 3 as well as the proposed techniques to address the portrait drawing robot problem. It describes the implementation of various automatic sketch generator methods and their definition and the proposed implementation of compensation drawing deviation techniques. Chapter 4 presents the results of the various implementations. Finally, Chapter 5 concludes this work, its contributions and possible future research directions. The overall scope of my research is depicted in Figure 1.2.

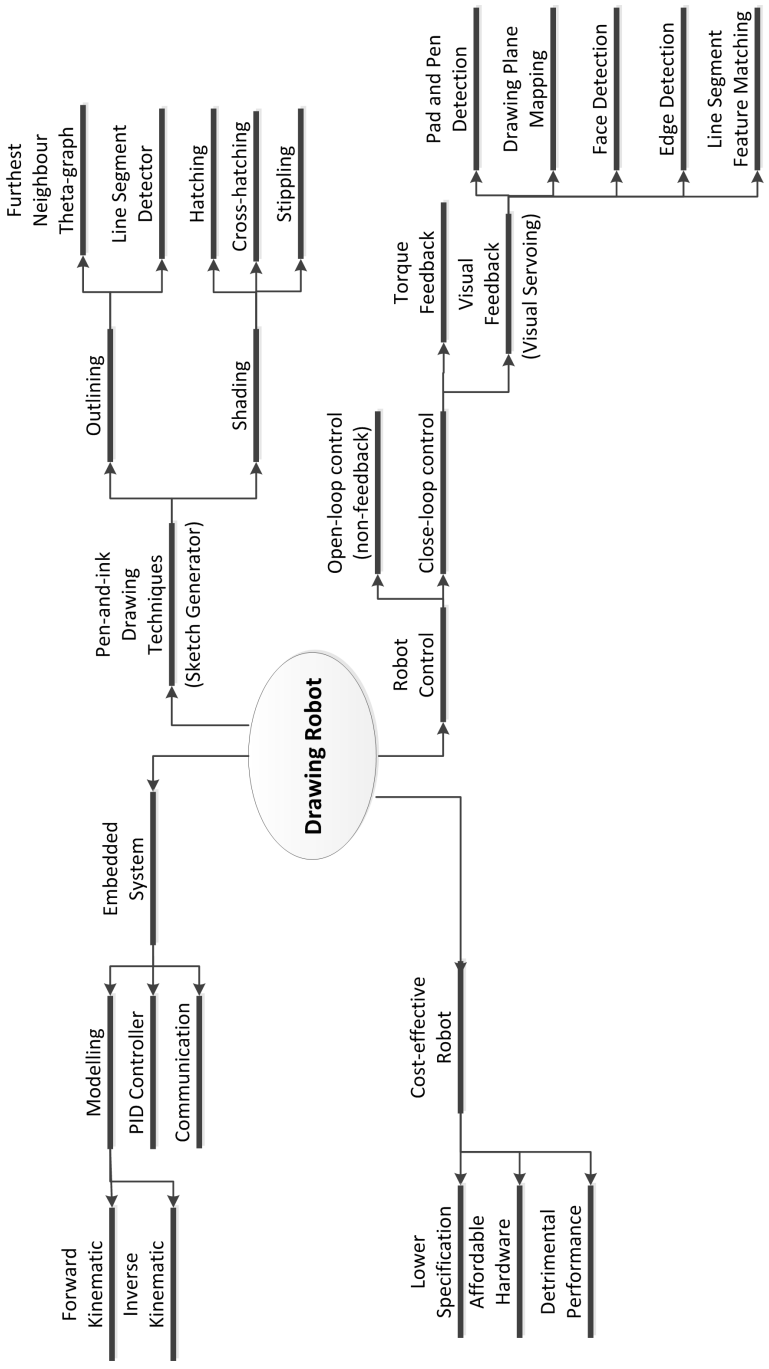


Figure 1.2: Overview of research areas.

1.4 Summary

This chapter serves as an introduction to my research on portrait drawing robots. It describes some of the limitations of existing approaches to drawing robots with the capability to obtain feedback during the drawing process. Finally, the chapter outlines my overall research area involving the portrait drawing robot.

Chapter 2

Background and Related Work

2.1 Introduction

The development of artist robots has received great interest since 1990 (see Figure 2.1). *AARON*, one of the oldest continuously developed programs in computing history (beginning in the mid-70's) attempted to create paintings that produced human-like artworks [24]. Gommel et al. [36] implemented an industrial robotic arm, KUKA, to draw in Cartesian space. Srikaew et al. [84] created a dual-armed humanoid robot, *ISAC*, that used the control of the end-effector with force sensors in the process of drawing. Ruchanurucks and Kudoh also implemented a multi-fingered robot called *Dot-chan* which was equipped with position- and force-sensing [56; 81] to produce color brush drawing. Katsura and his group have developed a calligraphy robot based on their Motion Copy System [94] which learn the input calligraphy motion (e.g position and pressure) to reproduce the brush strokes [52]. Tresset and Fol Leymarie created *AIKON/Skediomata* a 3-DOF low cost robotic platform dedicated

to drawing sketches [91; 92; 93]. Calinon et al. from École Polytechnique Fdrale de Lausanne (EPFL) used the small humanoid robot HOAP2 for their human portrait drawing task [16]. *Pica* was introduced by Lin et al. for fast human portrait drawing but its drawing results were very simple and lacked contours [61]. Another interesting approach was *Drawbot*, a wheeled-robot with a pen-holder mounted on an arm that allowed the pen to be lifted off the drawing surface [14]. A similar approach is used in a polargraph art robot, *Makelangelo* developed by robotic company called Marginally Clever [23]. It used a wall-drawing approach which draws on a piece of paper on the wall with a pen that adjusts its x-y position based on two strings controlled by two stepper motors. Lu et al. [63] developed a robotic manipulator called *IRAS* based on visual feedback that determines stroke positions and applies local gradient interpolation to guide stroke orientation in its pen-and-ink artwork. However, this was only in a fixed plotter-like setting. An in depth review of these works shows that none of them used visual servoing in humanoid robots as feedback for drawing observation, and none provide drawing correction during the drawing process. This is an important ability in pen and ink drawing. In this research, three closed-loop control techniques, namely torque feedback, IBVS and hybrid control are implemented to address the drawing deviation problem of Betty which is described in Section 2.3 and Chapter 3.

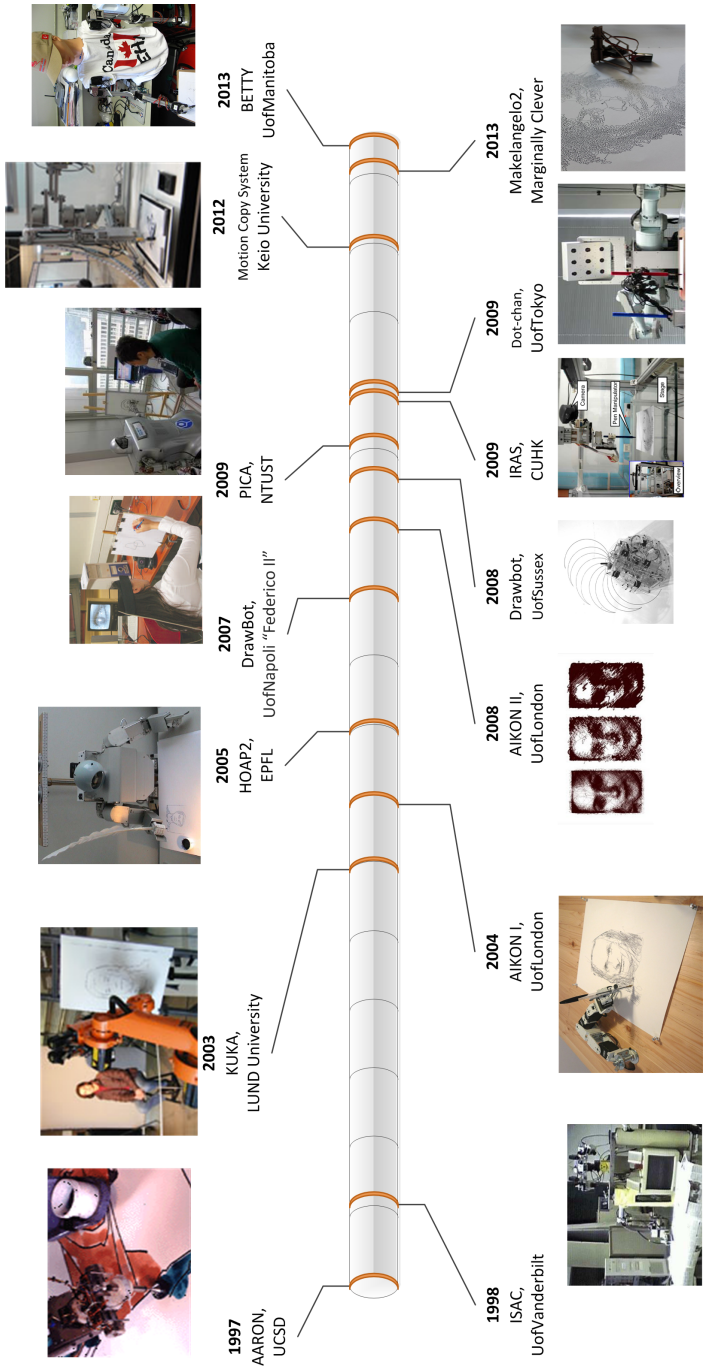


Figure 2.1: Current state of the art for drawing robot.

2.2 Automatic Sketch Generator

One of the main tasks in portrait drawing is outlining to sketch the outline of a portrait. Based on the current implementation of Betty, I am also interested in estimating the ideal edges joining points in the set of points of a pixel reduction image so that the number of lines is minimal. Therefore, I implemented a geometric graph called a Furthest Neighbour Theta-graph (FNTG)which is adapted from Theta-graphs [22; 50; 51] and ordered-Θ-graphs [11]. This novel graph is discussed in Section 3.4.2. A geometric graph is a weighted graph whose vertex set is a set of points, P, in d-dimensional Euclidean space, $\mathbb{R}^d$, and the edges of the graph consist of line segments, each of which joins two vertices. The weight of any edge is the Euclidean distance, L_2, between its endpoints [10].

Geometric graphs are used to model many practical problems in various fields of computer science and computational geometry. Theta-graphs [22; 51] and Yao-graphs [104] are popular geometric graphs that appear in the context of navigating graphs [10]. Theta-graphs and Yao-graphs differ in the way the nearest neighbour is defined. In both graphs, every vertex is joined by an edge to each of its nearest neighbours in each cone. In 2D Euclidean space, each cone forms an angle of $\Theta_k = 2\pi/k$, for some fixed $k > 0$. For Yao-graphs (Y_k-graphs), the nearest neighbour of p in the cone C is simply a vertex $q \neq p$ in C minimizing the Euclidean distance (L_2-distance) between p and q. For theta-graphs (Θ_k-graphs), the nearest neighbour of p is the vertex $q \neq p$ whose orthogonal projection onto the bisector of C minimizes the L_2-distance to p [10]. Bose et al. introduced a new variant of theta-graph called ordered-Θ-graphs, which are built incrementally by inserting the vertices one by one.

This means that generated graphs may be different based on the order in which the vertices are inserted so that the resulting graph depends on the insertion order [11]. They show that specific insertion orders can produce graphs with desirable properties, including low spanning ratio, logarithmic maximum degree and logarithmic diameter. Bonichon et al. introduced a specific sub-graph of the Θ_6-graph defined in $\mathbb{R}^2$, called half-Θ_6-graph, which consist of the even-cone edges of the Θ_6-graph [10]. They show that these graphs are exactly the TD-Delaunay graphs, and are strongly connected to the geodesic embeddings of orthogonal surfaces of coplanar points in 3D Euclidean space.

Line segment detection is an important research area in machine vision in recent years [2; 3; 37; 72; 97; 98]. The most popular method among them is the Hough transform (HT) [46], but the general usage of the HT was popularised after it is used by Duda and Hart [29] in 1972 and Ballard [6] in 1981. Hough transform is a method for estimating the parameters of arbitrary shapes (line is a form of shapes) from its boundary points. Usually HT is preprocessed with edge detection (e.g. Canny edge detector) to generate edges that cause too many false detections on complex images [54]. Line segment detector (LSD) is a linear time parameterless method introduced by von Gioi et al. in 2008 [37; 97; 98] to detect line segments based on Burns et al.'s method [15]. It uses an *a contrario* model from Desolneux et al. approach [27; 28] to validate the false positive of straight edges locally. Also note that there is a minimal length that a line segment must have, and smaller ones cannot be detected [98]. In 2011, Akinlar and Topal [2; 3] introduced a novel Edge Drawing algorithm (EDLines) to extract line segments in linear time and give accurate results,

requiring no parameter tuning, and runs in real-time (up to 11 times faster than LSD). EDLines makes use of the clean, contiguous (connected) chain of edge pixels based on the least squares line fitting method to extract line segments. There are also some specific purpose (e.g. UAV) line segment detection algorithms [60; 62; 72] which are not discussed in my thesis work. In my thesis I propose an extension algorithm based on the EDLines method to generate the sketch outline due to its least expensive processing compare to LSD. EDLines is the best (i.e. fast and accurate) generic line segment detection algorithm to the best of my knowledge. However these line segment detection methods extract each black stroke to produce two detections, one for each white to black transition. Therefore a extended version of EDLines implementation is introduced in this thesis to overcome this problem, namely eEDLines.

2.3 Drawing with Closed Loop Feedback Control

One of the main research questions in my thesis work is whether the proposed closed-loop approaches are significantly superior to open-loop's "blinded" (non-feedback) drawing. Three real-time feedback approaches are implemented to address the deviation correction problem: torque feedback, image-based visual servoing and hybrid control. As discussed in Chapter 1, the non-feedback open-loop approach might not be adequate for a sophisticated implementation of a drawing robot because it does not provide sufficient feedback during the drawing process.

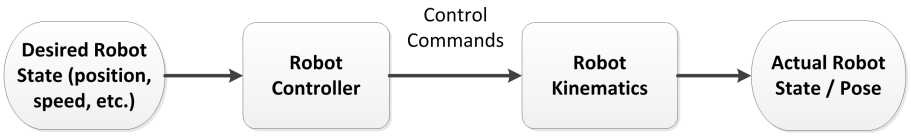


Figure 2.2: Open-loop control diagram

2.3.1 Open-loop Control: Kinematic Model

When using open-loop control in a robotic system, there is no online interaction between the robot and the environment where no data is collected from the robots sensors during task execution which can be seen in Figure 2.2 [33; 48]. At first, it extracts the information from a input image. From this, it generates a robot pose and motion sequence. This allows the control of a robot and image processing to be viewed as two independent tasks [55].

One of the main disadvantages of this approach is that the robot will not be able to correct its state if there are any changes in its environment during the process [48]. For instance, in my earlier implementation of portrait drawing, the portrait of a person is recognized and extracted, then its pixels position is computed relative to the camera (robot) coordinate system. This Cartesian space information is used to move Betty's arm to the desired position relative to the drawing pad. With the eye-to-hand camera and inverse kinematic model, the position of the pen is computed based on the visual information extracted from the camera frame. It then maps the pixels of the image to Betty's inverse kinematic model. Next, Betty executes the drawing task by performing open-loop movements which assumes the environment remains static after the drawing has started. However, without the real-time feedback of

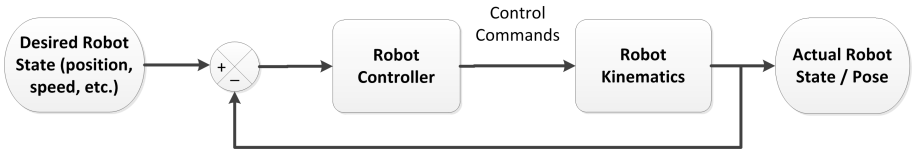


Figure 2.3: Closed-loop control diagram

what has been drawn by Betty, it is impossible to make corrections to ensure the desired output sketch is obtained at the end. Therefore, other closed-loop control methods are discussed in the following sections. Figure 2.3 shows an example of a closed-loop control system.

2.3.2 Closed-loop Control: Torque Feedback

In many applications [41; 53; 76; 89; 99; 105], a robot must explicitly control the force it applies to the object it is manipulating, i.e., the actuators must be controlled to achieve the desired forces. Force control using feedback of joint torques is limited by the accuracy over a resistor to estimate the input current of the motors [1; 64]. This indirect measurement has several errors, including the variations in the losses in the motor itself and the gearbox. To obtain accurate control of the force vector at the end effector, a wrist force sensor is placed between the tool plate and the end effector to measure end-effector force [5; 59; 75]. The force transform from the sensor to the end effector is simple but these load cell sensors are usually expensive. Torque in joint space is controlled by controlling the torque applied by each actuator (servo). Torque can be measured using a sensor (accurate) or estimated from armature current (simple).

2.3.3 Closed-loop Control: Visual Servoing

Visual servoing is a common closed-loop visual feedback control approach for object manipulation in robotic research e.g. [12; 21; 82; 95; 96] and industry e.g. [43; 73]. In contrast to open-loop control, a closed-loop control system uses vision as the feedback sensor to continuously track the environment, which provides estimation and updates for robot control and positioning. Therefore the control sequence is generated based on a consistent visual feedback of the error between its current and desired poses of the end-effector.

The terminology of *visual servoing* was introduced in 1979 by Hill and Park, [42] prior to the general term "visual feedback" [47]. Generally, visual servoing techniques are classified into three types; image-based visual servoing (IBVS), position-based visual servoing (PBVS) and advanced visual servoing [18; 19].

Figure 2.4 which was adapted from Kragic and Christensen [55], illustrates the general idea of a visual servoing system to estimate the differential changes between robot joint space and the image space with a coupled robot-image Jacobian.

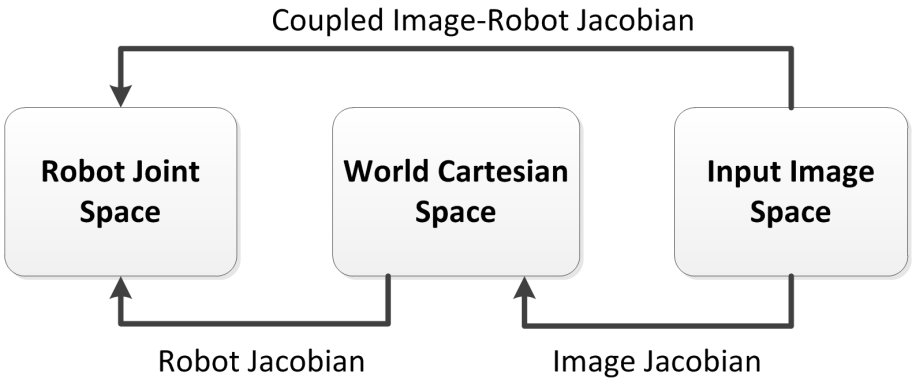


Figure 2.4: General processes of visual servoing. Robot Jacobian relates the knowledge of robot kinematics and the image Jacobian refers the differential changes between image features on a image space and the pose of the robot.

Position-based visual servoing (PBVS):

PBVS processes the error signal from the image features in order to compute the control sequence of the robot based on the relative 3D pose between the camera and the end-effector [25; 102]. PBVS solves the 3D localization problem where 3D information of the scene, including a geometric model of the target, and the parameters of the particular camera model (camera calibration) is known. PBVS is considered to be a 3D vision sensor. Usually there are two tasks in a PBVS system. First, as mentioned, it requires a model between camera and end-effector to estimate the pose of the target. Second, it has to estimate the desired velocity screw of the robot, which requires precise camera-robot calibration to achieve accurate positioning [55]. Figure 2.5 shows the general design of a PBVS system.

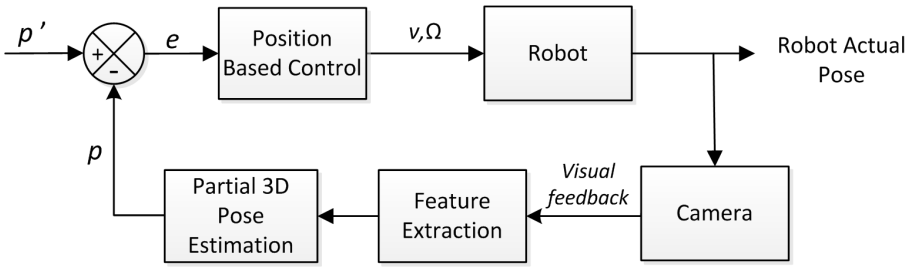


Figure 2.5: Block diagram of PBVS. p is the actual pose of the robot and p' is its desired pose. In this case, positioning task (velocity screw, which can be denoted as $\dot{P} = [V; \Omega]^T$) is described by its pose error, e as presented by Kragic and Christensen [55]

Image-based visual servo systems (IBVS):

IBVS was introduced by Weiss and Sanderson [100; 101] to compute the pose of the robot by processing the error signal from the image features parameters e.g. lines and image areas directly as its feedback control signals [25]. IBVS involves the computation of the image Jacobian or the interaction matrix [47; 55]. In contrast to PBVS, IBVS is considered as a 2D vision sensor. IBVS eliminates PBVS's requirement of perfect knowledge of the camera calibration matrix and its complex interpretation of image features to derive 3D world-space coordinates. Figure 2.6 shows the general design of an IBVS system.

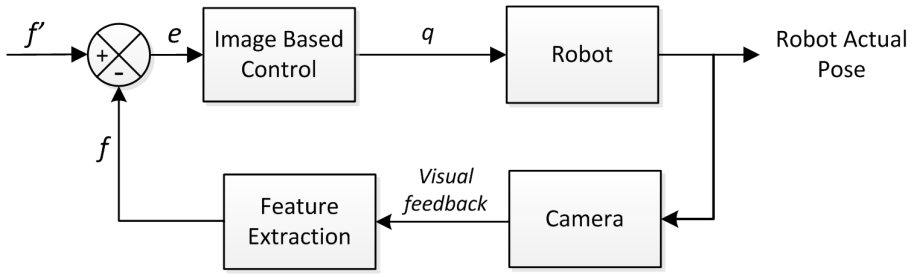


Figure 2.6: Block diagram of IBVS. f is the actual location of feature and f' is its desired location. The positioning task is described as location error, $e = f - f'$ and q represents the coordinate of the end-effector.

Advanced visual servoing:

There are some advanced visual servoing systems (e.g. *Hybrid Visual Servoing*, *Partitioned Visual Servo*, etc.) which have been proposed to address the drawbacks of PBVS and IBVS [19]. For instance, *Hybrid Visual Servoing*, also known as 2 1/2-D visual servoing is presented by Malis et al. [67] in 1998. This approach combines the basic PBVS and IBVS. As compared to PBVS, there is no complex geometric 3-D model of the object required in the hybrid approach. On the other hand, hybrid visual servoing ensures the convergence of the control law in the whole task space (faster to reach the desired position) [68]. However, in some approaches, the visual servoing model may be estimated by either analytically (nonlinear least square optimization) or learning to completely remove the calibration step [55]. Figure 2.7 shows the general design of an advanced visual servoing system.

In my portrait drawing problem, a 2D image (from the drawing pad) is used directly to estimate the desired pose of Betty's arm. This is a typical task of IBVS, where the image distance error (drawing variation) between the current and desired

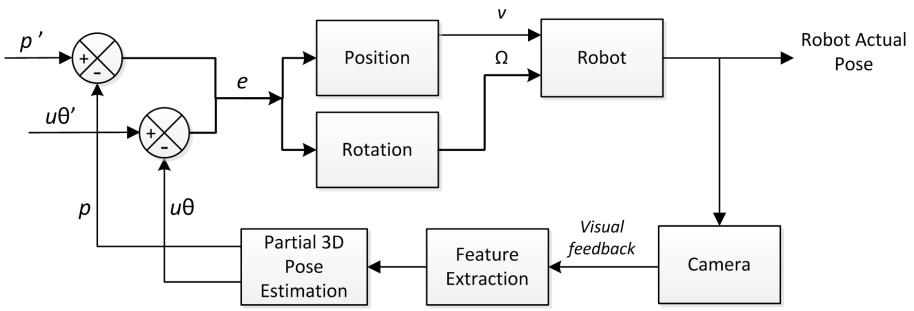


Figure 2.7: Block diagram of advanced visual servoing. p and $u\theta$ are the actual pose and orientation (u is rotation axis and θ is rotation angle) of the robot and p' is its desired pose. In this case, positioning task (velocity screw, which can be denoted as $\dot{P} = [V; \Omega]^T$) is described by its pose error, $e = [f - f' u^T \theta]^T$ as presented by Kragic and Christensen [55]

image features in the output image plane are tracked. I therefore proposed to use IBVS to address this problem in my work.

Generally, IBVS can divided into two separate systems. First, a dynamic look-and-move system. This system performs the control of the robot in two stages: the vision system provides input to robot controller that then uses joint position feedback to control the end-effector. As pointed out by Hutchinson et al., nearly all of the reported systems adopt this approach [47]. Second, direct visual servo systems. Here, the visual controller directly computes the input to the robot joints and robot controller is eliminated (no position feedback) [55]. As discussed by Chaumette and Hutchinson, there is no strategy that provides perfect properties based on their stability issues [18]. However, the coarse estimations in IBVS will only imply perturbations in the trajectory performed by the robot to reach its desired pose (longer convergence time) and will have no effect on the accuracy of the pose reached. On the other hand, IBVS has several advantages over PBVS. In IBVS, a 3-D environment model

is not required. Its target always stays in the field of view of the camera, and its performance is robust with respect to camera calibration errors and noise when a monocular camera is used [12; 18; 25]. Therefore, I use this image-based eye-to-hand visual servoing to control the relative pose between the drawing tool and the drawing pad with a single camera mounted on Betty's head.

For the drawing task, a camera (right eye) is used as a global vision sensor (eye-to-hand configuration). It retrieves and tracks the position of the features (lines and shading) in the image relative to the Betty's inverse kinematic frame. In contrast to PBVS, great robustness with respect to calibration error can thus be expected.

Table 2.1: Comparison of visual servoing techniques.

	IBVS	**PBVS**	**Advanced VS**
Feedback control signals	Image features	Image features (Camera and end effector)	Partial of both
Vision sensor	2D	3D	2 1/2D
Pros	No perfect camera calibration and complex interpretation of image features	3D model of the observed object to be known	Advantages of both
Cons	Visibility limitations from coarse estimation	Sensitive to camera calibration	Complexity of implementation

Table 2.1 shows the comparison of the three different visual servoing approaches based on their feedback control signals, vision sensor, advantages and disadvantages.

2.4 Summary

In this chapter, I have given an overview of current approaches to drawing robots. I discussed few line segment detectors which I use as sketch generator, and the existing feedback techniques for robot control. I have addressed how each of these robot control approaches applies to my research questions.

Chapter 3

Design and Implementation

3.1 Introduction

As described in previous chapters, I am attempting to develop a robust humanoid system to create sketch like drawing with limited hardware. No force sensor but basic torque feedback from servos to estimate pressure apply on drawing pad. I also use visual information (e.g. drawing pad pose, pen location and actual sketch) to correct drawn output in a 2D X-Y planar.

In this portrait drawing problem, vision is an important component which is used in both knowing what to draw (human portrait) and how to draw it (line sketching). Hence, I implemented the Furthest Neighbour Theta-graph (FNTG) and the extended EDLines (eEDLines) to solve the current portrait sketching problem of Betty. However, the implementation is not limited to a graph and line segment detection algorithms but it also includes several image processing algorithms, such as thresholding, Canny edge detection [17] and features extraction based on image mo-

ment. The development took place on a Linux platform with Qt Creator and OpenCV library. Qt Creator is a cross-platform integrated development environment (IDE) which support C++ for Qt program development.

In this chapter I describe the design and implementation of the hardware, software, pen-and-ink techniques, and drawing correction mechanisms of my drawing robot: Betty. I tested the kinematics control by running it on different applications which are simulation, drumming, and plotting [7]. Then, I introduce a novel technique called Furthest Neighbour Theta-graph (FNTG) and extended EDLines (eEDlines) algorithm to generate sketch like output in vectors of line segments. A torque feedback control (TFC), an image-based visual servoing IBVS system, and the combination of both methods are introduced to provide significant feedback for drawing deviation correction throughout the drawing process in a 3D space. The evaluation will be discussed in chapter 4.

3.2 Hardware

The upper body of Betty consists of a twelve-revolute joint system with 12 degrees of freedom (DOF). Figure 3.1 shows the overview of Betty's upper body. Its head has 4 DOFs which are pan, tilt, swing and one DOF for the mouth. Each of its arms provides 4 DOFs, shoulders allow lateral and frontal motions, elbow gives lateral motion and a wrist motion. These joints are constructed by four Dynamixel RX-64 servos in the head and four Dynamixel RX-64 servos for each of its arms as shows in Figure 3.2.

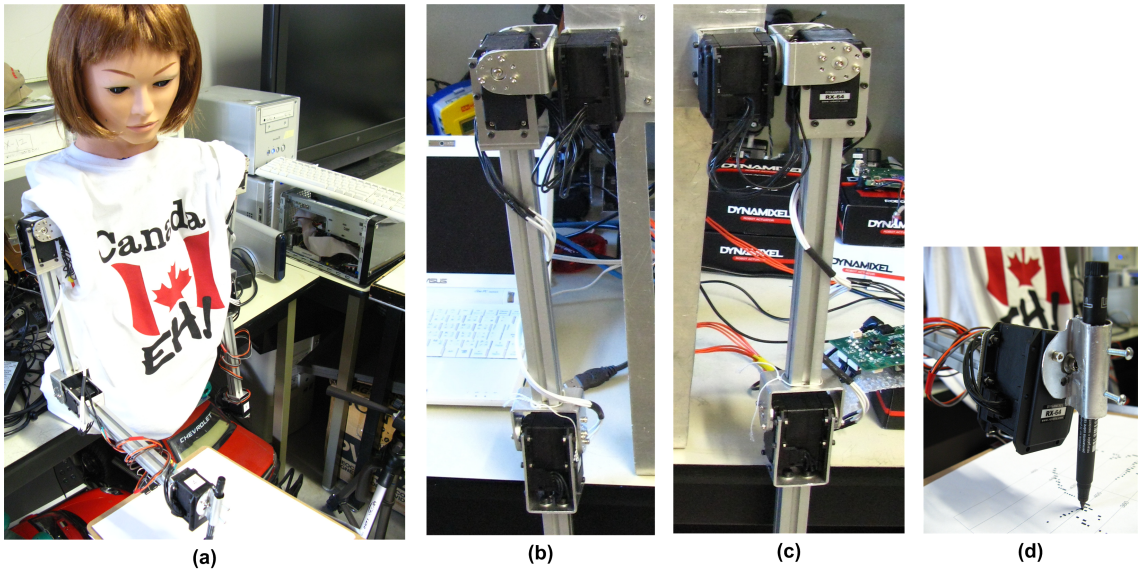


Figure 3.1: Overview of Betty's 12-DOF upper body.

The main reason I chose RX-64 to construct Betty's arms is it has higher final maximum holding torque, 64.4∼77.2 kgf.cm compared to only 12∼16.5 kgf.cm for the AX-12 [79; 80] which were used in Betty's previous design [7; 58]. This improvement allows the RX-64 servos to generate sufficient torque to support the weight of Betty's arms and head.

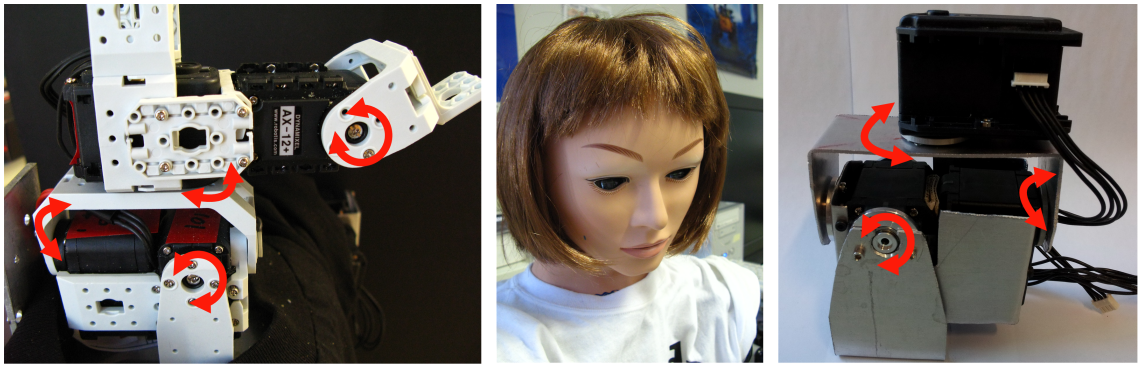


Figure 3.2: Betty's joints construction: **(a)**: Previous AX-12 head; **(b)**: Current RX-64 head.

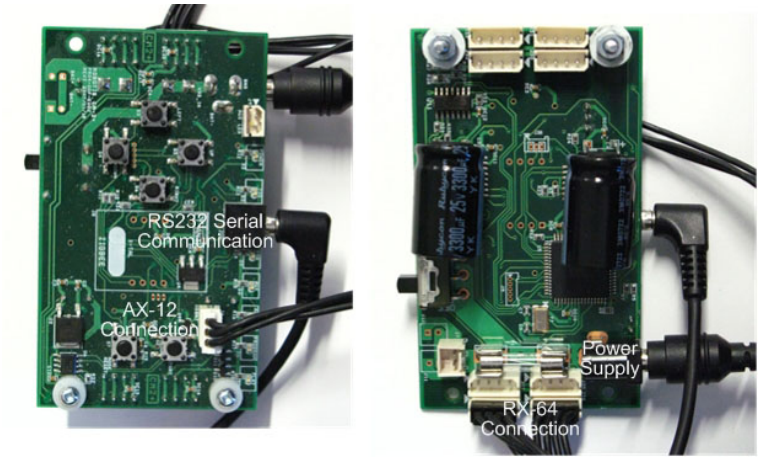


Figure 3.3: CM-2+ Connection.

In order to control the servos, I use Dynamixel's dedicated controller, CM-2+ from Robotis as the central control unit with its `AVR ATmega128` microcontroller. The real-time OS establishes communication with the servos. Figure 3.3 shows the connections on the CM-2+. For Betty's vision, it has two Logitech QuickCam Orbit MP, 1.3 Megapixel, pan and tilt motorized webcam. The total cost of the hardware is less than USD 3000 [7; 58].

3.3 Software

A real-time embedded control system is needed to enable the communication between a controller program and the servos to respone from events such as position and torque feedbacks within the operational deadlines. Designing and implementing a real-time embedded system is a challenging task in my development. It is crucial to ensure that Betty can responds in the expected period of time. For a robot as complex as Betty, much preliminary experimentation is required to determine the

best settings to minimize latency and jitter. Latency is the time between starting to send a command and receiving a response from the servos where the command has been executed. Jitter is the time variation of latencies. I looked into two types of robot kinematics: forward and inverse kinematics. In forward kinematics, the length of each link and the angle of each joint is measured so I can calculate its end effector (hand) position by using the Denavit-Hartenberg (DH) convention. In inverse kinematics, the length of each link and position of Betty's hand is given to generate the angle of each joint. I use a kinematic diagram to solve the inverse kinematics problem.

I developed a pre-emptive multi-tasking kernel in the Control Program to handle several tasks. Currently the kernel supports four tasks [58]. The first task is an idle task to toggle an LED. This task is always ready to run to ensure that at least one thread is running and the task switcher is working properly. The second task is a communication thread that handles serial port communication between the Motion Editor program and the CM-2+. The third task is a servo thread which prepares commands to send from the CM-2+ to the RX-64 servos. The last task is the torque thread that reads the current torque of each servo and sends it to computer through the communication thread. The pre-emptive multi-tasking kernel uses a timer interrupt to switch between different threads at an initial timer frequency of 10Hz. To execute task switching, the kernel will save the complete task state on the stack and then store the stack pointer in the task control block (TCB).

I chose an interrupted linear queue as the data structure to handle communication data. A queue is a data structure where item insertions are made at the rear and item

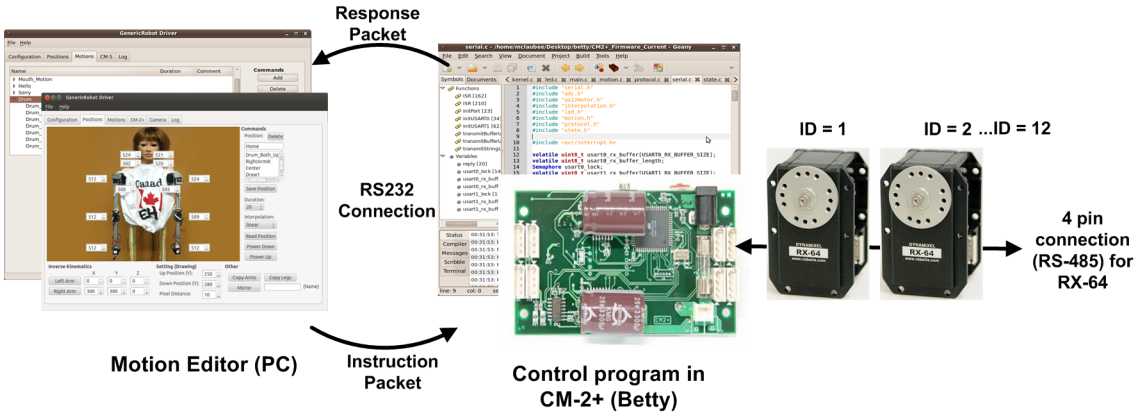


Figure 3.4: General program framework of Betty.

retrievals or deletions are made at the front of the queue. Because the first item added to the queue will be the first one to be removed, it is commonly referred to a first-in-first-out (FIFO) data structures [49]. I implemented a circular queue solution and the experimental results are discussed in [58]. A circular queue provides a few advantages over a linear queue. It can be accessed at both ends of its array and it uses a constant size buffer which minimise memory usage and let the program handles data streaming efficiently [103]. But data in a circular queue can be replaced unintentionally if the algorithms in the program is not designed properly. On the other hand, a linear queue can only be accessed at one end linearly so it is safer and easier to implement compare to circular queue.

The Motion Controller program is a program running on the PC to control the robot's motion based on its vision input. Figure 3.4 shows the overall framework of the system that explains the connection architecture of Betty's Motion Editor, Control Program, CM-2+ and servos. I use Absolute Position as the communication protocol: it has the advantage of simplicity in implementation compared to the other protocols.

Table 3.1: Communication Protocols.

Protocol	Command Structure
Absolute Position	FF0101198282828282828282E5 FF: Header 01: optional torque reply 01: SyncWrite command 19: Speed 82: Position E5: Checksum = lower byte of ∼(01+19+82+82+82+82+82+82+82+82+82+82)
Sliding Resolution	FF01011988888888883C (First frame) FF01010219222222222239 (Second frame) FF: Header 01: optional torque reply 01: SyncWrite command 01: Highbytes or 02: Lowbytes 19: Speed 88: combination of 2 Highbytes 22: Combination of 2 Lowbytes 3C: Checksum = lower byte of ∼(01+01+19+88+88+88+88+88)
Difference	FF0101192222222223B FF: Header 01: optional torque reply 01: SyncWrite command 19: Speed 22: Each 4-bit represents the difference between the desired and the previous positions. 1 to 7 represent increment and 8 to F represent decrement 3B: Checksum = lower byte of ∼(01+19+22+22+22+22+22)

The differences between various protocols were discussed in [58]. Based on these protocols, the Motion Editor will send instructions to the CM-2+'s Control Program. I use a `SyncWrite` command to move the servos to specific positions synchronously. Table 3.1 shows the structures of the Absolute Position protocol where a `SyncWrite` command were sent to move all servos from position 512 to 520 at the speed of 100. I divided the positions and speed by 4 so each of them can be wrapped into one byte in hexadecimal. According to Table 3.1, each `SyncWrite` command of the Absolute Position protocol will send 15 bytes of instruction in one packet.

As seen in Table 3.1, I also tested two different protocols namely the Sliding Resolution and Difference protocols. The Sliding Resolution protocol sends its command in two separate packets which are high and low packets. The high packet consists of the most significant four bits of every position whereas the low packet contains the least significant four bits. The advantage of this protocol is to start the servos as soon as a command is received. So while the Control Program receives the high packet, all servos will start moving towards their target positions then the final target positions will be determined once the low packet has been received.

In the Difference protocol, the Control Program receives two types of instruction packets, Absolute and Relative. The Absolute packet is similar to the one described in the Absolute Position protocol. Then the Relative packet will be sent with only four bits per position which encodes the differences between current and new target positions. Although the Difference protocol showed better performance in both latency and jitter experiments. The relative performance was not significant in most cases. The Difference protocol provide better performance only if the differences be-

tween new and current positions is less then 28 unit for each servo movement as seen in Appendix A. In contrast, the Absolute Position protocol is simpler and easier to implement and it offers more reliable performance when compare to the other two protocols. The Absolute Position protocol is chosen based on its reliability and simplicity compare to the other two protocols. Their performances are discussed in Appendix A and [58].

A Proportional-Integral-Derivative, PID controller is the most popular feedback control algorithm [69]. Implementation of a PID controller in the Control Program is based on the feedback of current latency and jitter from torque responses which will modify its context switching frequency. The context switch frequency is the frequency of storing and restoring computing processes in an embedded system. It ensures the pre-emptive kernel is robust enough so the Control Program can modify its context switching frequency in real-time. Figure 3.5 shows the basic components of feedback control in a PID controller.

In the PID control loop, the control output (co) is calculated based on the following equations:

$$e(t) = sp - pv \tag{3.1}$$

$$P_{out} = K_P e(t) \tag{3.2}$$

$$I_{out} = K_I \sum_{t=0}^{n} (e(t)) \tag{3.3}$$

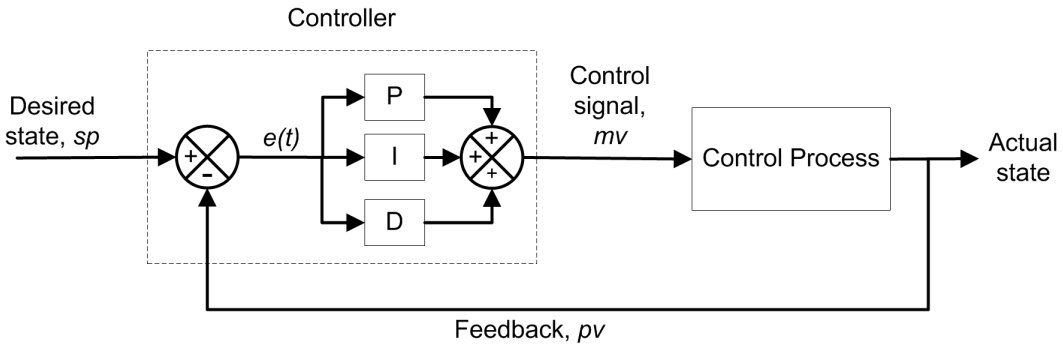


Figure 3.5: Block diagram of PID controller.

$$D_{out} = K_D[e(t) - e(t-1)] \tag{3.4}$$

$$co = P_{out} + I_{out} + D_{out} \tag{3.5}$$

Firstly, the error, $e(t)$ is calculated in equation (3.1) by subtraction of the set point (sp) and the process variable (pv) which is the latency of current run (t) where $t = 0, 1, 2...n$ and n is the current time. Then the proportional, integral, and derivative terms of the error are determined by equations (3.2), (3.3) and (3.4). Finally, all PID outputs are summed to calculate co as equation (3.5). K_P, K_I and K_D are the constants called proportional, integral and derivative gain respectively which are tuned on trial and error basis in the case of Betty. The measured latency is the process variable (pv), the desired latency is called the set point (sp) and the manipulated variable (mv) is the context switching frequency of Betty's embedded system. The difference between current measured latency pv and defined latency sp is the error(e). In the control process, the context switching frequency, latency mv is adjusted by

calculating the error $e(t)$ at time t between the measured pv compared to the desired set point (15000 μs as seen in the Appendix A). The steady state of context switching frequency is determined if over 10 consecutive occurrences of pv remain between the control limits of $\pm 5\%$ from the sp are measured.

3.3.1 Kinematics

I solved Betty's kinematics problem with the Denavit-Hartenberg (DH) Convention in the forward kinematics and employed a geometric approach in the inverse kinematics. These are the reasonably general approaches that allows Betty to participate in a variety of applications, such as portrait drawing, Wii drumming and 3D model simulation [58]. Figure 3.6 explains the coordinate frame system attached and assigned to each arm.

The Motion Controller computes the forward kinematics to calculate the global position of Betty's hands (end-effector). Although, a simple two-joint planar manipulator analysis is possible to solve with some simple transformation matrices, the accumulated kinematics analysis of a multiple-joint manipulator can be extremely complicated. Denavit and Hartenberg [26; 40] used screw theory in the 1950's to show that the most compact representation of a general transformation between two robot joints required four parameters. These parameters are now known as the Denavit and Hartenberg convention (D-H convention) and they are the de facto standard for describing a robot's geometry [44]. Thus, I applied the DH-convention to solve the forward kinematics analysis problem by introducing a simplified presentation to calculate the position and the orientation of the end effectors. This approach is widely

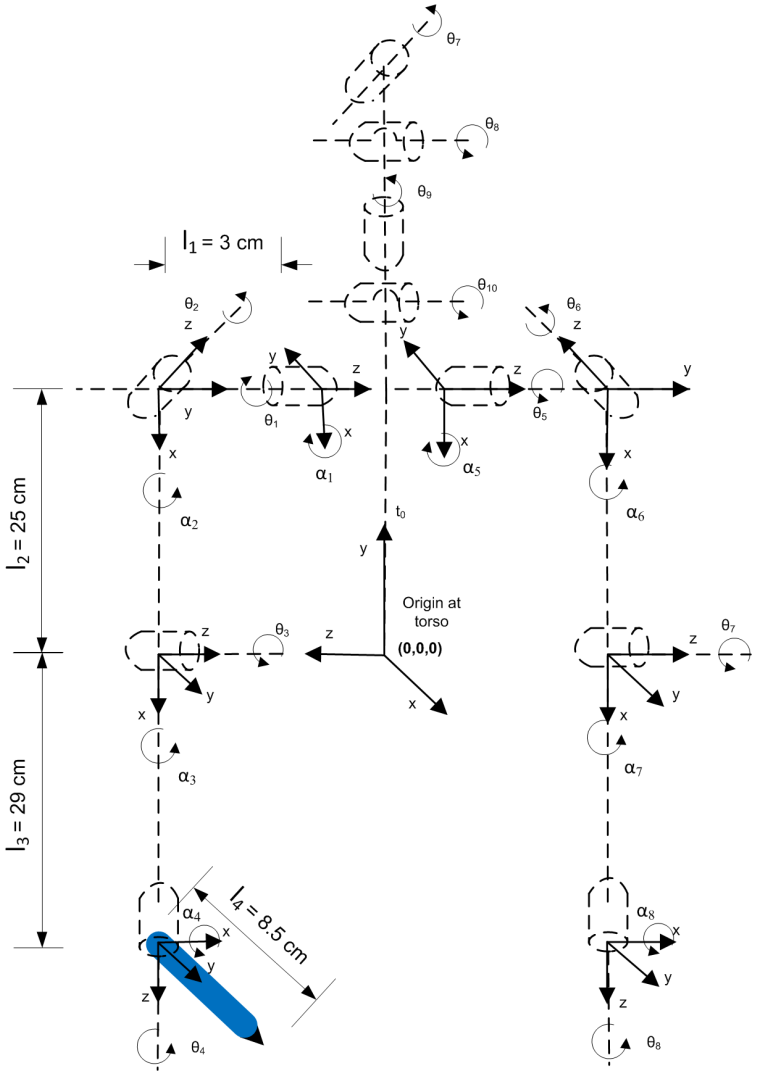


Figure 3.6: Kinematic diagram of Betty's coordinate frame system

used in robotics and animation. In this convention, each homogeneous transformation, D_i is represented as a product of four basic transformations matrices to get the global position of any particular point at joint i [83].

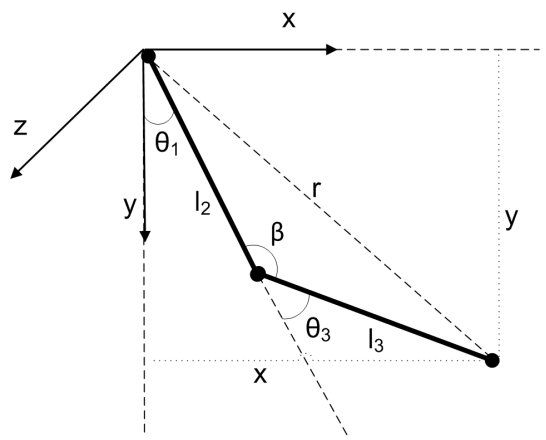


Figure 3.7: XY-plane projection of Betty's right arm.

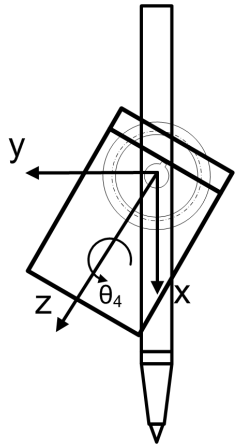


Figure 3.8: Pen and wrist orientation compensation where $\theta_4 = -\theta_2$.

$$
\begin{aligned}
D_i &= R_{z,\theta_i} T_{z,d_i} T_{x,a_i} R_{x,\alpha_i} \\
&= \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{i1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.6}
$$

In equation (3.6), R_{z,θ_i} is the rotation matrix for angle θ_i by the z-axis, R_{x,α_i} is the rotation matrix for angle α_i by the x-axis, T_{z,d_i} is the transformation matrix for length d_i by the z-axis, and T_{x,a_i} is the transformation matrix for length a_i by the

Table 3.2: The DH-parameters for both arms.

Joint,i	a_i	α_i	d_i	θ_i
1	l_1	α_1 =90	0	θ_1
2	l_2	α_2 =90	0	θ_2
3	l_3	α_3 =0	0	θ_3
4	0	α_4 =90	l_4	θ_4
5	l_1	α_5 =90	0	θ_5
6	l_2	α_6 =90	0	θ_6
7	l_3	α_7 =0	0	θ_7
8	0	α_8 =90	0	θ_8

x-axis. The detail descriptions of the DH-parameters are:

a_i - the distance between two joint axes measured along the x-axis.

α_i - the relative twist between two joint axes measured about the x-axis.

d_i - the distance between the two perpendiculars measured along the joint axis, z-axis

θ_i - joint angle about the z-axis

According to the coordinate frame system in Figure 3.6, the DH-parameters are shown in Table 3.2. To establish the homogeneous coordinate between the origin and Betty's arms, both shoulders' coordinate frames are translated to the origin at its torso with translation matrices, T_L and T_R for the left and right shoulders respectively where t_y and t_z are translation parameters from shoulder to torso.

$$\text{Where, } T_L = \begin{bmatrix} 1 & 0 & 0 & t_y \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and } T_R = \begin{bmatrix} 1 & 0 & 0 & t_y \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Once the DH-parameters and relative frames are determined as shown in Figures 3.6, 3.7 and, 3.8, I combined the transformation matrices as the product of translation and DH-Convention matrices. The example shown in equations 3.7 and 3.8 are the calculation of the right hand's global position, denoted as P_R. This is based on the rotation angles of θ_1, θ_2 and θ_3 from the origin O with the transformation matrix M_R. θ_4 at the right wrist adapts the inverse angle of θ_2 to compensate the frontal motion of the shoulder which retains the perpendicular orientation of the pen and the drawing pad.

$$
\begin{aligned}
M_R &= T_R D_1 D_2 D_3 \\
&= \begin{bmatrix} 1 & 0 & 0 & t_y \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta_2 & 0 & -sin\theta_2 & l_2 cos\theta_2 \\ sin\theta_2 & 0 & cos\theta_2 & l_2 sin\theta_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & l_3 cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & l_3 sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.7}
$$

$$
P_R(x, y, z) = M_R O \tag{3.8}
$$

Previously we discussed how to calculate the global position with DH-Convention based on the assigned servos' rotation angles. Following is concerned with the inverse

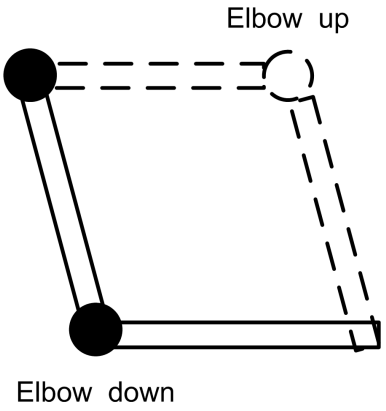


Figure 3.9: Example of elbow up/down configuration.

method of finding each of the joints' angle with a given global position. For the implementation of Betty's hand positioning, I used a geometric approach for solving the inverse kinematics problem. The reason for using a geometric approach is because the design of Betty's arms is geometrically simple. This makes it easier to solve, compared to a general inverse kinematics problem [83]. Generally there are multiple solutions to a inverse kinematics problem, which could be added to the challenge like elbow up/down configuration as seen in Figure 3.9. The $atan2$ function is used rather than simply using inverse sine or cosine to take care of multiple solutions problem. Based on Betty's coordinate frames system in Figure 3.6 and Figure 3.7 , the equations to calculate the servos' rotation angles for Betty's right arm can be solved as shown:

$$\theta_2 = atan2(Z/Y) \tag{3.9}$$

$$r^2 = X^2 + Y^2 \tag{3.10}$$

$$\beta = acos((l_2^2 + l_3^2 - r^2)/(2l_2l_3))...(Law\ of\ Cosines) \tag{3.11}$$

$$\theta_3 = \pi - \beta \tag{3.12}$$

$$\theta_1 = atan2(X/Y) - asin(l_2 * sin(\beta)/\sqrt{r^2}) \tag{3.13}$$

$$\theta_4 = -\theta_2 \tag{3.14}$$

3.4 Sketching

3.4.1 Pen-and-ink Drawing Techniques

There are many different drawing media and techniques for portrait drawings; including pencil, pen-and-ink, crayon-and-charcoal and brush-and-ink [71].

Pen-and-ink drawing has a long history back to the illuminated manuscripts of the Middle Ages, where text is supplemented by decorations such as decorated initials. However, this drawing technique has only became commonly used since the end of the 19th century and has since then developed into its own art form [38]. Pen-and-ink drawing is similar to pencil drawing in that the use of line and technique is closely related. However pen-and-ink has different drawing tools to choose from, ranging from quill pens and dip pens to modern drawing pens like ballpoint, fountain and marker pens. Generally, human faces are organic in nature with delicate features,

and these require thinner drawing lines. Therefore, the most suitable tool to use is a pen with a fine-tip nib like a ballpoint pen and its advantages of no leakage, lower cost and ease of maintenance.

3.4.2 Outlining: Furthest Neighbour Theta-graph

In this subsection, I introduce the formal definition of Furthest Neighbour Theta-graph or Θ-graph (FNTG) and establish some of its basic properties. Let P be a set of n points in the plane and let θ be an angle such that $k_\theta = 2\pi/\theta$ is a positive integer. For each point $p \in P$, partition the corresponding disc D, with radius r into a set of k_θ cones C, each spanning an angle of θ with apex at p; see Figure 3.10. Then, add an edge joining p to the vertex in each cone of C whose projection onto the clockwise cone boundary is the furthest L_2-distance from p. Therefore, for any point set P, the Furthest Neighbour Theta-graph $G = (P, E)$ of P has at most $k_\theta n$ edges of E in disc D.

Using an algorithm analogous to that described by Bose et al. [11] for constructing ordered Θ-graph, for any set P of n points in $\mathbb{R}^2$ and any point in disc D, the Furthest Neighbour Theta-graph G can be computed in $O(n(\log n)/\theta)$ time. The construction algorithm for Furthest Neighbour Theta-graph requires finding the furthest neighbours of each point $p \in P$ bounded by a k_θ-cone disc with apex at p. We can use k_θ range trees [8] for each cone. In each tree, every point q of P is stored using a coordinate transformation to (x, y), where x and y correspond to the respective distances between the projections of p and q on the boundaries of the given cone with apex at p. See Figure 3.10 and 3.11.

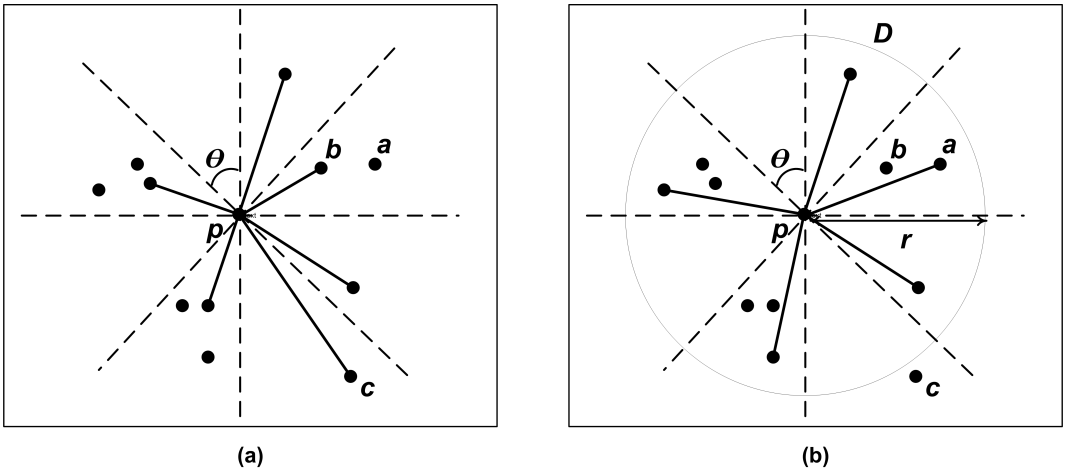


Figure 3.10: Comparison between the theta-graph and a Furthest Neighbour Θ_8-graph with $k_\theta = 8$. **(a)**: Illustration of notations for Θ_8-graph where the edges are defined by nearest neighbours of p, **(b)**:An example of Furthest Neighbour Θ_8-graph at p, where a is the furthest neighbour of p compared to b, and c is not bounded by disc D.

Each vertex requires adding at most k_θ edges, each of which is determined using one range search. Thus, the graph can be constructed using a series of $k_\theta n$ searches in range trees. Each range tree requires $O(n \log n)$ space and supports construction in $O(n \log n)$ time [9], with $O(\log n)$ update time [57]. Using this implementation of range trees, the above algorithm computes a Furthest Neighbour Theta-graph in $O(n(\log n)/\theta)$ time. Figure 3.12 shows the vision pipeline to generate a sketch-like portrait with FNTG.

The first step of the implementation is colour conversion from a four-channel RGBA image to a single-channel grayscale image. Next, it generates a binary image from the grayscale image by applying a fixed-level thresholding to remove noise in the output image where pixels with value beyond the threshold are filtered. After thresholding, a Canny edge detector is applied. In this case, it produces a full size

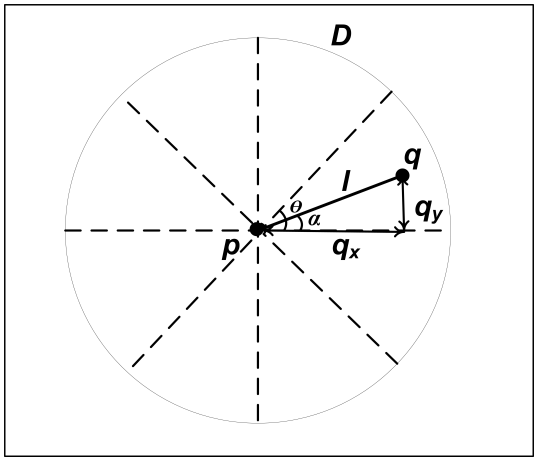


Figure 3.11: Finding the furthest neighbour q of p on disc D with the furthest L_2-distance l in the first k_θ-cone.

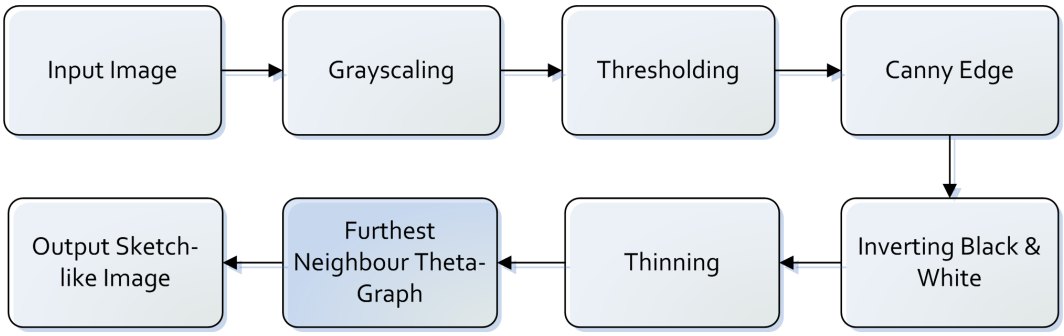


Figure 3.12: Overview of the general vision pipeline.

image as the input image, but with only black (0) and white (255) pixels, which is known as a single-channel boolean image. The algorithm processes each individual edge candidate pixel into intensity gradients by applying an hysteresis threshold to the pixels, where the higher values are more likely to correspond to edges compared to smaller values. Therefore two threshold parameters are required, an upper and a lower threshold. If a pixel has a gradient higher than the upper threshold, then it is accepted as an edge pixel; if a pixel is less then the lower threshold, it is rejected. If the pixel's gradient is between the thresholds, then it will be accepted only if it is

connected to a pixel that is above the high threshold [13].

The next step would be inverting the black (0) and white (255) pixels from the output edges by applying bitwise XOR operation on the image array with the *cvXorS()* function. The bitwise XOR is computed with the constant scalar value [13]. In order to reduce the number of pixels to a reasonable number, an image thinning algorithm has been deployed by a 3x3 convolution matrix and its anchor is placed at the middle of the kernel. The function *cvFilter2D()* applies arbitrary linear filter to the image and then interpolates outlier pixel values from the nearest pixels. Its thinning output is illustrated in Figure B.2.

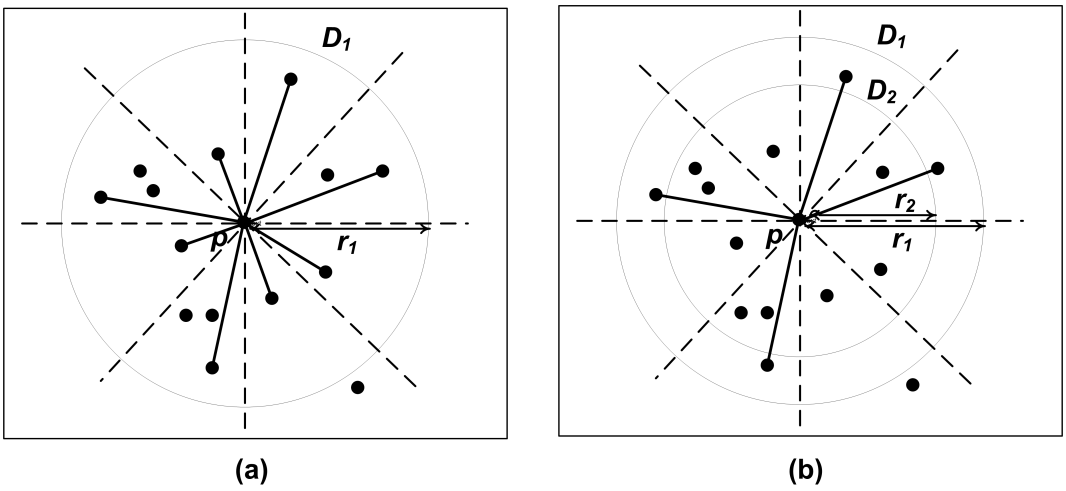


Figure 3.13: Implementation of the Furthest Neighbour Theta-graph. **(a)**: The initial proposed single-disc approach. **(b)**: A dual-disc approach to reduce the number of less significant edges.

In the final step, the line sketch of a portrait is generated with the FNTG algorithm. However, to reduce the number of less significant edges in an input image, a dual-disc approach is implemented. Figure 3.13(a) shows that the initial single-disc approach produces many insignificant lines if the neighbours are too close to p, which

will affect the presentation of the portrait outline. Consequently, Figure 3.13(b) illustrates a proposed dual-disc approach to solve the problem by only considering the points which fall between the inner and outer discs' boundaries. The algorithm for finding the edges of E at any point of P can be described as in algorithm 3.1.

Algorithm 3.1 Generate FNTG

Require: p
Ensure: $e_{p,q} \neq e_{q,p}$

1: Read dark pixels into a sorted list
2: **for all** dark pixel p in P **do**
3: Superimpose D_1 and D_2 centred at p
4: **for all** cone in k_θ cones **do**
5: Search and update the furthest neighbour within D_1 and D_2
6: **end for**
7: Add an edge e to p
8: **end for**
9: **for all** edge e in E **do**
10: Remove overlap edges if $e_{p,q} = e_{q,p}$ where p and q denote the end points of the edge
11: **end for**
12: Draw edges

Figure B.2 in Appendix B shows the implementation of the interactive GUI which displays the FNTG computed from an input. By modifying the parameters of various properties on the graph (e.g, discs' radii and number of cones) an output or sketch with fewer edges could be generated. The results are discussed in subsection 4.2.1.

3.4.3 Outlining: Extended Edge Drawing Lines (eEDLines)

Here I define a line segment of two endpoints A and B ($\overline{AB}$) as a finite line vector. It connects endpoints A and B in AB direction as shown in Figure 3.14. The location of a line segment such as $\overline{AB}$ on the 2-D coordinate plane (line sketch

images) is defined by two endpoints. Their coordinates are known as $A(x_0, y_0)$ and $B(x_1, y_1)$. This line segment links the two endpoints but does not extend beyond them.

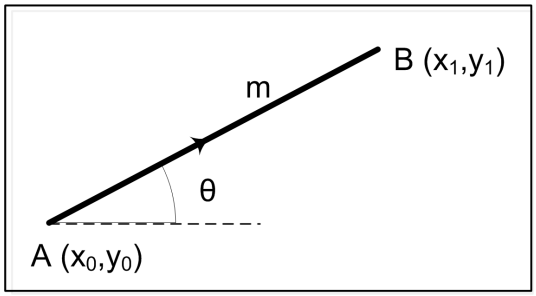


Figure 3.14: Definition of a line segment.

Line Segments Grouping

Based on the defined vector representation, line segments which are near-collinear will be merged. To determine if the line segments are near-collinear, it starts with compares the slopes, θ between the line segments where θ is defined as equation 3.15 and $-\pi \leq \theta \leq \pi$. So, if the difference is within a given threshold angle then these line segments are near-collinear and could be joined if they are close to each other.

$$\theta = \begin{cases} atan2(y_1 - y_0, x_1 - x_0), & \text{if } \theta \geq 0 \\ atan2(y_1 - y_0, x_1 - x_0) + \pi, & \text{otherwise} \end{cases} \tag{3.15}$$

The next problem is determining whether these line segments are close enough to be joined based on their Euclidean distance. The Euclidean distance between any two line segments, $\overline{AB}$ and $\overline{CD}$ is defined as the minimum distance, d between

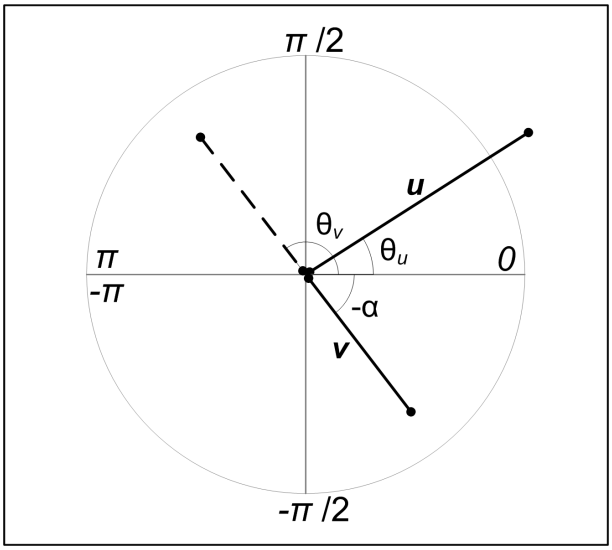


Figure 3.15: Line segment slope using $atan2$. Slope of line segment $\boldsymbol{v}$ is measured as $\theta v = -\alpha + \pi$

any two of their points p and q. This distance can be computed using calculus method [30]. However, I used a geometric approach based on Dan Sunday's geometry algorithms [85]. This approach has fewer cases to check compared to the calculus approach by [30] which is more difficult to implement [85]. There is a similar geometric approach used in Seth Teller's [88] example but it is limited to 3D space and slower then the previous methods [85]. Based on Dan Sunday's approach, the minimum distance between $\overline{AB}$ and $\overline{CD}$ is defined as equation 3.16.

$$d_{p,q} = \min_{p \subseteq \overline{AB}, q \subseteq \overline{CD}} d_{\overline{AB},\overline{CD}} \tag{3.16}$$

In any n-dimensional space, assume the two line segments $\overline{AB}$ and $\overline{CD}$ are closest at unique points p and q. Let $\mathbf{w}(p, q)$ be the vector of unique minimum distance between the vectors u and v as described in the quadratic function (equation 3.17) of

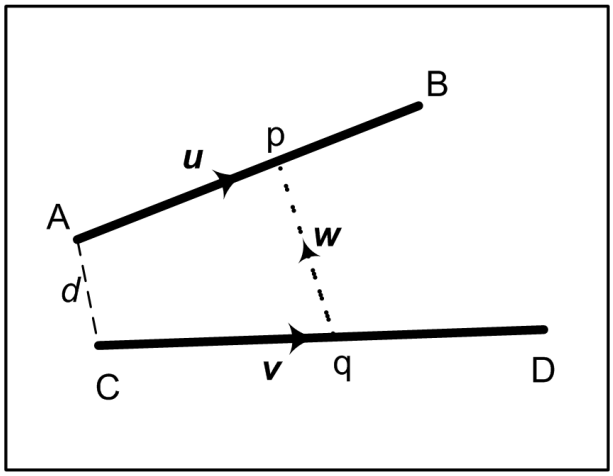


Figure 3.16: Closest distance between two line segments.

p and q where $0 \leq p \leq 1$ and $0 \leq q \leq 1$.

$$\mathbf{w} = p\mathbf{u} - q\mathbf{v} + \mathbf{w}_0, \text{ where } \mathbf{w}_0 = A - C \tag{3.17}$$

When the vector $\mathbf{w}$ is uniquely perpendicular to the vectors u and v as their minimum distance, it is equivalent to satisfy two equations where $\mathbf{u}\cdot\mathbf{w} = 0$ and $\mathbf{v}\cdot\mathbf{w} = 0$. Then substitute equation 3.17 into these two equations to get two simultaneous linear equations as shown in equations 3.18.

$$\begin{aligned}(\mathbf{u}\cdot\mathbf{u})p - (\mathbf{u}\cdot\mathbf{v})q &= -\mathbf{u}\cdot\mathbf{w} \\ (\mathbf{v}\cdot\mathbf{u})p - (\mathbf{v}\cdot\mathbf{v})q &= -\mathbf{v}\cdot\mathbf{w}\end{aligned} \tag{3.18}$$

Then the linear equations are solved as equations 3.19.

$$p = \frac{be - cd}{ac - b^2} \text{ and } q = \frac{ae - bd}{ac - b^2}, \text{ where } ac - b^2 > 0 \tag{3.19}$$

Therefore, the minimum distance, d can be described as the magnitude of $\mathbf{w}$, $d = \|\mathbf{w}\| = \sqrt{w \cdot w}$. Assume the four edges of the two line segments A, B, C and D are given by $p = 0$, $p = 1$, $q = 0$, and $q = 1$. Consider $p = 0$, if $0 \leq q \leq 1$, then the two closest points are A and q. However, if q lies outside of $\overline{CD}$, then C or D is the minimum along $\overline{CD}$. The other edges are treated in a similar manner.

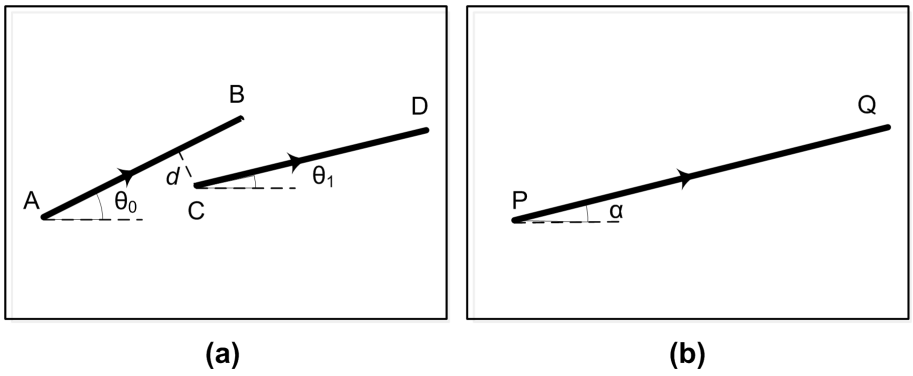


Figure 3.17: Merge two nearest near-collinear line segments.

Figure 3.17 shows the merging example of two line segments. By considering the distance, d and slopes of the line segments, θ_0 and θ_1 as seen in equations 3.20 where the furthest edges A and D are joined to define a new line segment, $\overline{PQ}$. Figure 3.18 shows more line merging examples.

$$d < D_t \text{ and } \mid \theta_0 - \theta_1 \mid < \Theta_t \tag{3.20}$$

where D_t and Θ_t are the thresholds

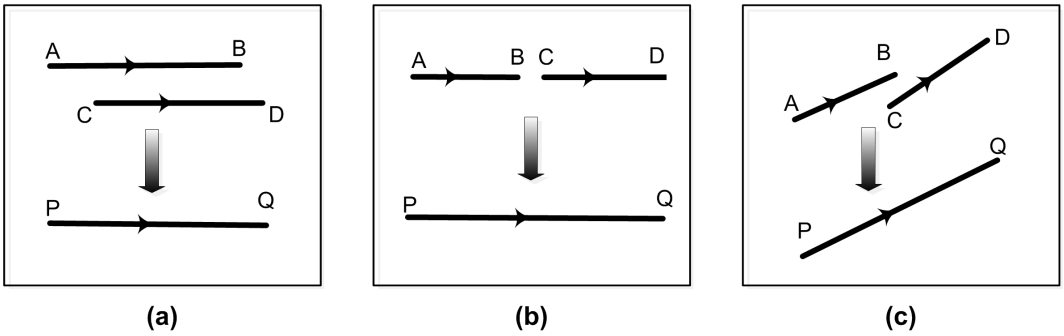


Figure 3.18: Examples of different line segments merging.

Sketch outline generator

As discussed in the previous section, the extended Edge Drawing Lines (eEDLines) algorithm is an extended version of the EDLines algorithm. It eliminates the near-collinear line problem and its data flow is depicted in Figure 3.19.

First, the region of interest (ROI) of a portrait is extracted from an input by the available Haar-Cascades face detector in OpenCV. Then I use the standard EDLines algorithm to detect line segments. These line segments are sorted by their y values using quicksort in a C++ standard library. Next, line segments that lay near the edges of the portrait and short insignificant lines (based on the given threshold, e.g. line segments with fewer than 10 pixels) are removed from the line vectors. The final

step is to merge all near-collinear line segments to minimize the number of significant line segments.

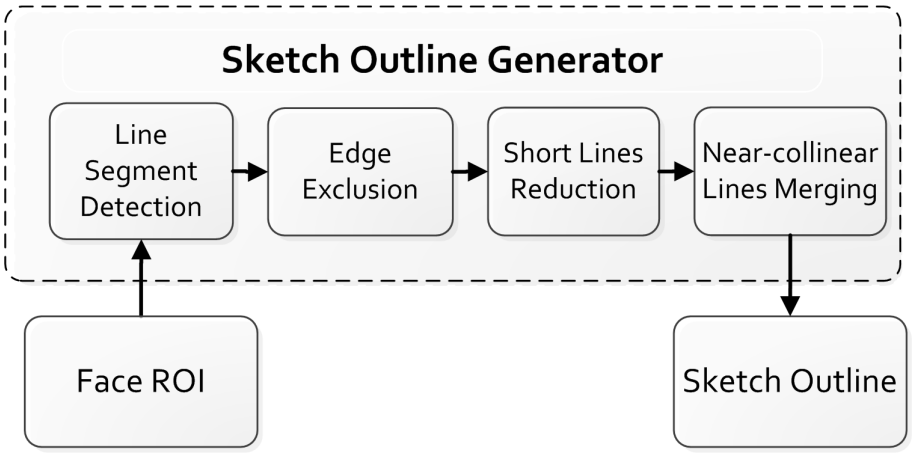


Figure 3.19: Flowchart for extended Edge Drawing Lines (eEDLines) data flow.

Based on Dan Sunday's algorithm [85], algorithms 3.2 and 3.3 describe the way to find the shortest distance between two finite line segments u and v and merge them as seen in Figure 3.16. Algorithm 3.2 shows how I compute the shortest distance, w of two line segments from their infinite vectors u and v. Next, these line segments are merged by algorithm 3.3 if they are near-collinear to each other with a given threshold angle difference between them. This threshold value could be selected based on perception of the output sketch, e.g. 8°in most cases.

Figure 3.20 shows the differences between standard EDlines (b) and the eEDlines algorithms (c). It clearly displays how the near-collinear line segments are reduced. The detail results of various line segment detection algorithms will be discussed in the next chapter.

Algorithm 3.2 Find shortest distance between two finite line segments

Require: line segments, s_1 and s_2

1: Define line segment infinite vectors of u, v and w
2: Compute the line parameters of the two closest points a,b,c,d as seen in equation 3.18
3: Compute vector w the closest points p and q to the vector u and v respectively by solving linear equations as seen in equation 3.19
4: Check the visibility of the finite line segments based on their edges A, B, C and D
5: Calculate distance, $dist$ between p to q as $\sqrt{w \cdot w}$
6: **return** $dist$

Algorithm 3.3 Merge near-collinear line segments

Require: line segments, S

```
1: Read line segments into a sorted list, S
2: for each line segments s_i and s_{i+1} in S do
3:    for each line segments s_{i+1} in S do
4:       if s_i and s_{i+1} are approximately collinear then
5:          Compute the closest distance between s_i and s_{i+1} with algorithm 3.2
6:          if s_i and s_{i+1} distance < threshold then
7:             Merge s_{i+1} to s_i into a new line segment by joining the furthest edges
8:          end if
9:       end if
10:      Remove s_{i+1} from list
11:   end for
12: end for
13: Compare the next line segments
```

eEDLines has an execution time proportional to the number of pixels in the image as suggested in the standard EDLines algorithm, because both the total number of pixels involved and the number of regions are at most equal to the number of pixels. Due to the line segments sorting (quicksort) step in the algorithm, the complexity of its processing time increases to $O(n^2)$ time in the worst case but it is usually faster, which is $O(nlog_n)$ in most cases.

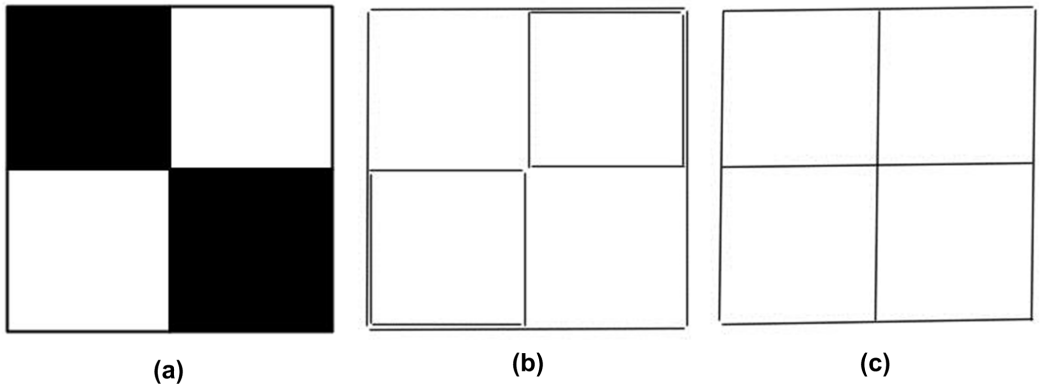


Figure 3.20: Extended EDLines Detector. (a) Input checker image, (b) Output from standard EDLines, (c) Output from extended line segment detector.

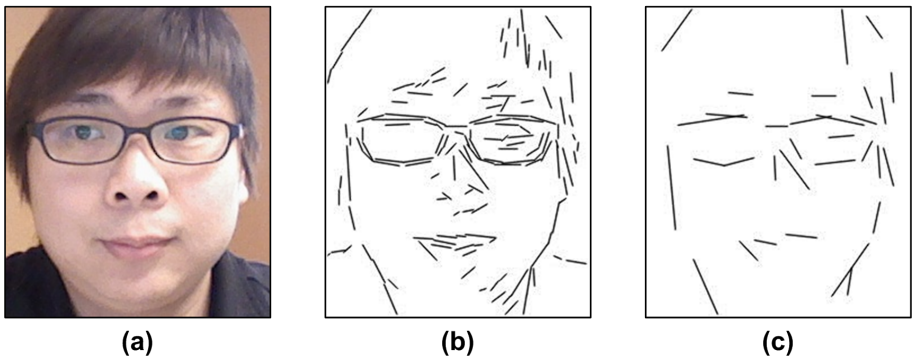


Figure 3.21: Sketch outline output comparison. **(a)**: input image **(b)**: standard EDLines output and **(c)**: eEDLines output

3.4.4 Shading

There are several basic techniques to shade in pen-and-ink to create different tonal value. These techniques generally require different densities of lines or dots to express the relative proportion of light and dark that reflect contour and features of a face. There are three basic shading techniques: hatching, cross-hatching and stippling as seen in Figure 3.22 [66]

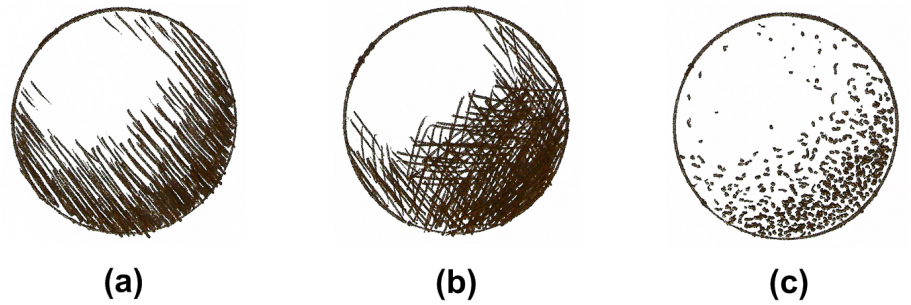


Figure 3.22: Different shading techniques, **(a)**: Hatching, **(b)**: Cross Hatching and **(c)**: Stippling.

Hatching:

Hatching is a technique used to create shading value with straight lines in the same direction for a defined area. When hatching is used, the lines do not cross over each other. Hatching lines can be parallel or they can be used as cross contour lines to help define the form of the object. The closer the lines are to each other, the darker the value. The more space between the lines, the lighter the value.

Cross Hatching:

Cross hatching is just like hatching except that the lines cross over each other. The more that the lines cross, the darker the value. Cross hatching can be used with rigid straight lines or as cross contour lines to define the form of the object.

Stippling:

Stippling is adding dots to create the extent of the shading. The higher the concentration of dots, the darker the shade. The more distance between the dots, the lighter the shade. Stippling can produce highly realistic drawing due to its robustness

of shading value. but it may be time consuming for Betty.

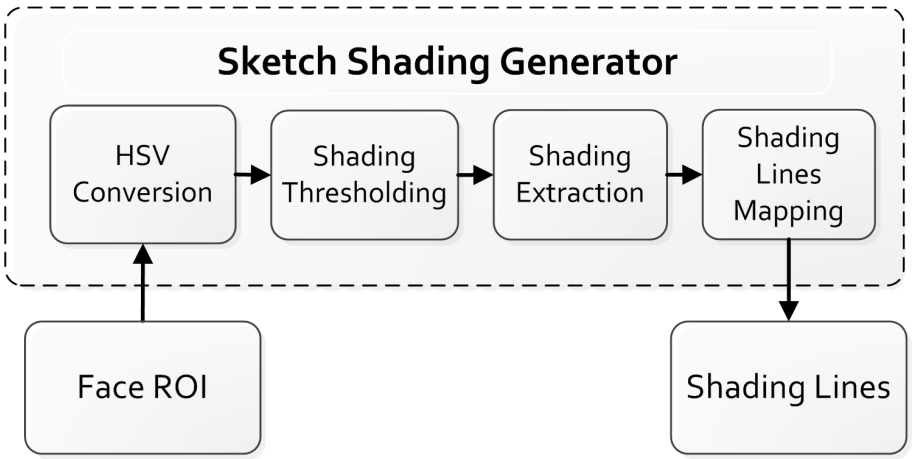


Figure 3.23: Flowchart for Shading generator.

Shading is the process of darkening to illustrate varying levels of darkness on a sketch. Figure 3.23 shows the flowchart of the sketch shading extraction process implemented to extract shading lines (e.g. hatching or cross-hatching).

First, the image or extracted portrait is saved as a single channel grayscale PGM (Portable Gray Map) format and its binary image is created by applying a fixed-level thresholding to extract pixels shading as seen in Figure 3.24(b). After thresholding, closing (dilation and erosion) and opening (erosion and dilation) operators [32; 77] are applied to the image. Closing fills small holes (white) and opening removes small objects (black) as seen in Figure 3.24(c). Then I applied an exclusive OR operator (XOR) to this extracted shading with a chosen shading pattern (Figure 3.24(d)) to create shading lines like Figure 3.24(e). Figure 3.25 shows some different shading pattern applied to a checker image.

Last, the algorithm processes the shading lines image to create the line segments

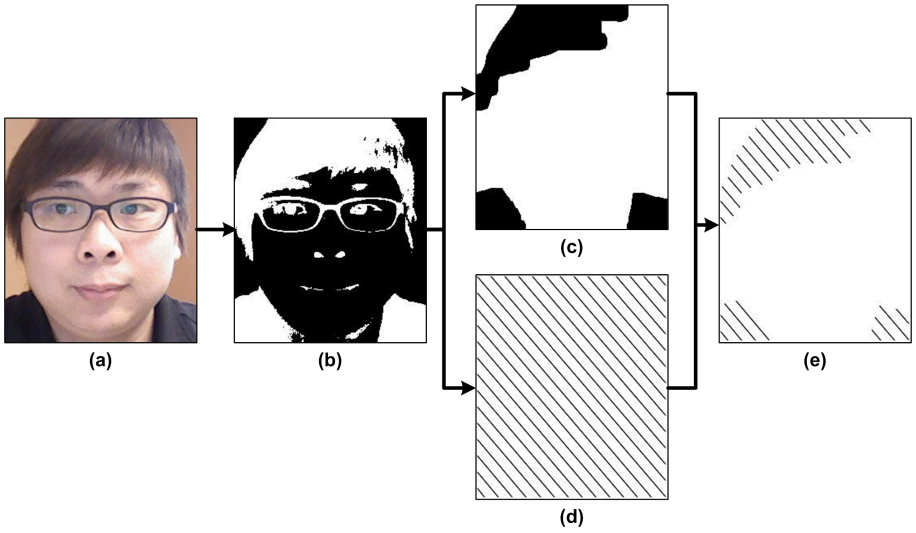


Figure 3.24: Sketch shading (a): input image (b): binary threshold (c): extracted shading (d): shading pattern (e): shading lines.

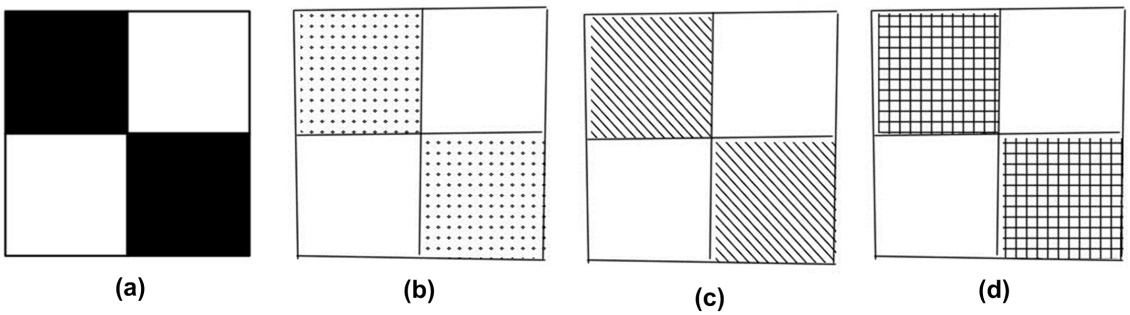


Figure 3.25: Different shading outputs. **(a)**: input checker image **(b)**: stippling **(c)**:hatching **(d)**: cross-hatching.

vectors. To produce a complete line sketch image, outline and shading are overlaid as a single image as shown in Figure 3.26

Figure B.4 in appendix B shows the screen-shot of the implemented automatic sketch generator.

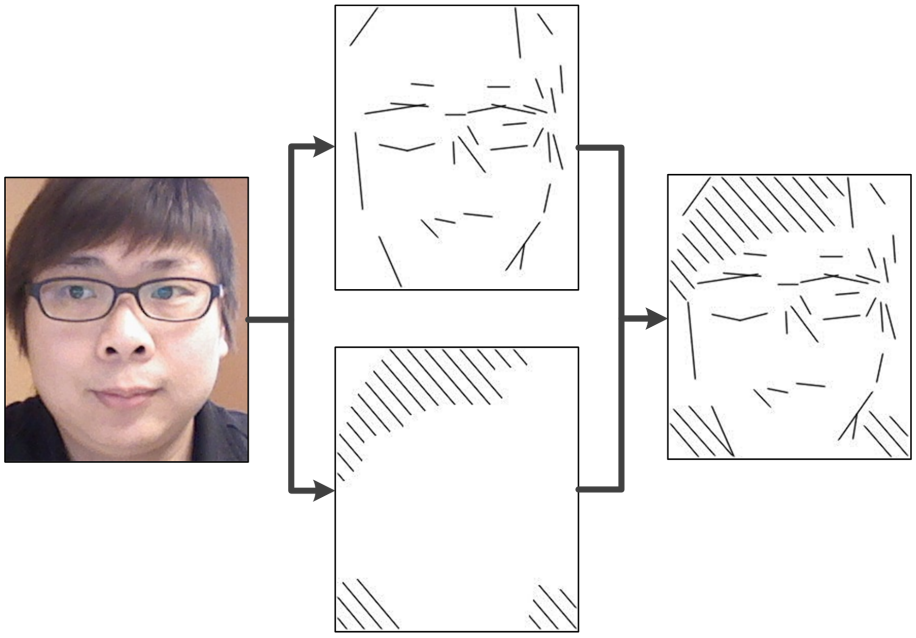


Figure 3.26: Full sketch is computed by overlying both outline and shading lines.

3.5 Compensation of Drawing Deviation

3.5.1 Torque Feedback Control (TFC) Model

For Dynamixel RX-64 servos, torque can be measured on a 0-1023 (0x3FF) scale, based on its maximum holding (stall) torque 64 kgf-cm (6.92 Nm) at 18V. As seen in Tira-Thompson's [90] study, a Dynamixel servo can read back the current position, temperature, load (torque), etc. For instance, position read back can measure to within 0.5 degree of accuracy at full communication speed in every 130ms. However, reading the present load would not provide precise torque measurement. Robotis posted a comment on their official website that the "Present Load" or torque measurement is not a real torque or electrical current measurement. It is actually just based on the difference between current position and goal position. It has a control

loop to make sure the motor actually got to the goal position. Hence, the "Present Load" value is the torque that the servo is *applying*, not the torque that it is *experiencing*. Therefore, it is not possible to perform "Torque Control" of a servo [31].

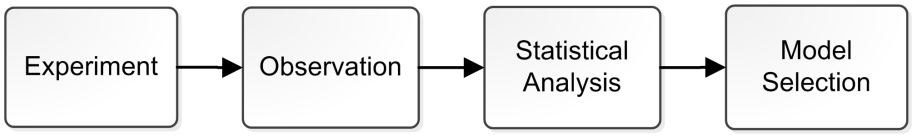


Figure 3.27: Designing of torque feedback control (TFC) model.

However, in the brief tutorial book of Anderson [4], he suggested the reality can be estimated based on the empirical evidence hypotheses and statistical models as shown in Figure 3.27. In my approach, I designed a simple torque feedback control (TFC) model based on the two simplest strokes, horizontal and vertical. Some preliminary experiments are performed to obtain suitable sets of data from the servos' torque feedback (right arm). By using this data, a simple TFC model is created and a suitable set of candidate models is selected based on statistical hypothesis testing to estimate stroke pressure. "Given candidate models of similar predictive or explanatory power, the simplest model is most likely to be correct" [4]. Figure 3.28 shows the design of Betty's TFC model.

The TFC approach is implemented to observe pen pressure during the drawing process to prevent overpressure or no-touch errors. Several experiments were conducted to obtain the statistical model. It provides optimum thresholds to estimate pen pressure. In these experiments, the pressure applied on the Wacom Bamboo tablet (underneath the paper) are recorded.

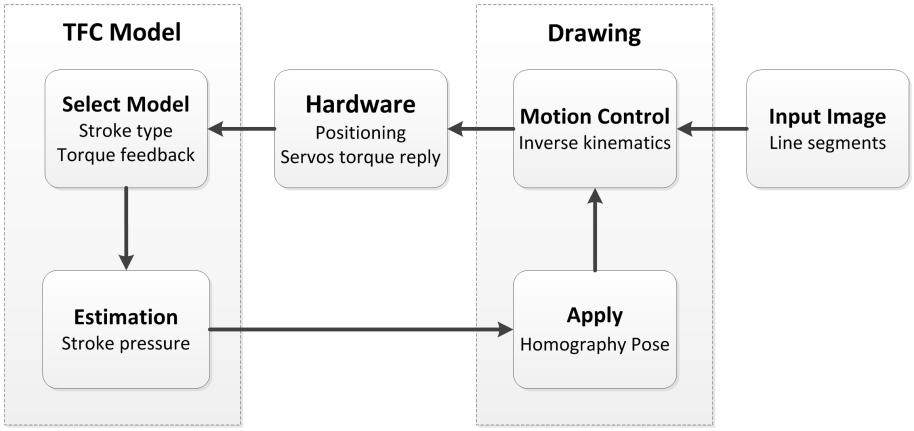


Figure 3.28: Processing flow of torque feedback control (TFC) for drawing task.

Betty's right arm performed two basic stroke types, namely vertical and horizontal strokes. Two directions are drawn for both types of stroke. These are upward-downward and left-right. Figures 3.29 and Figure 3.30 show the average vertical strokes and Figures 3.31 and Figure 3.32 show the average horizontal strokes from 100 recorded strokes respectively in different directions. Medium pressure is the normal pressure read from the Wacom Table in the range of 0.7 ± 0.1 and high pressure represents 1.0 (highest) pressure measurement from the tablet. From these preliminary experiments we observed that high pressure caused significant errors in both horizontal and vertical strokes compares to normal pressure. So, it makes the estimation of stroke pressure essential.

During the experiments, I recorded the torque measurements of each servo on Betty's right arm. By using this torque data as seen in the following tables, a simple TFC model is created. Hence, the suitable set of candidate models can be selected based on statistical hypothesis testing to estimate stroke pressure.

Table 3.3 shows the torque measurements from four servos on the right arm namely

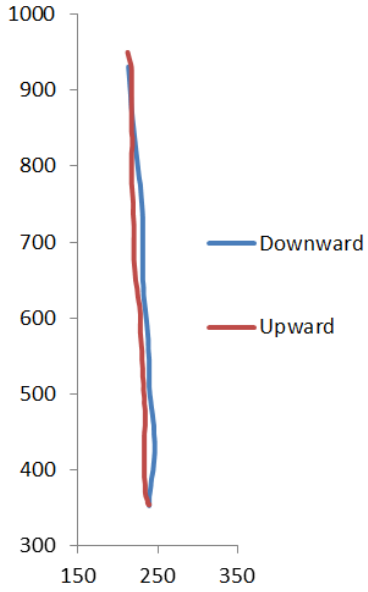


Figure 3.29: Medium pressure vertical strokes.

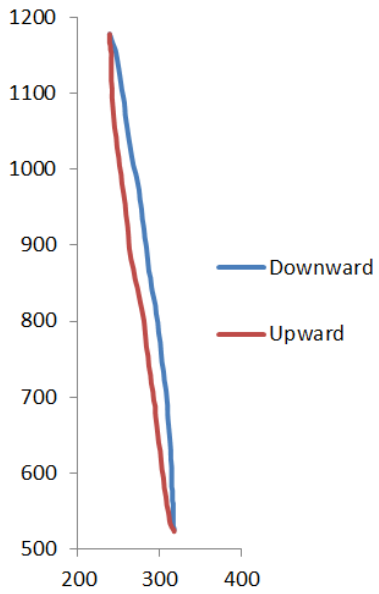


Figure 3.30: High pressure vertical strokes.

ID 1, 2, 3 and 4 for shoulder-frontal, shoulder-lateral, elbow and wrist respectively. Table 3.4 shows the model of vertical downward stroke which suggests how significantly torque measurements are affected by different stroke pressures. As seen in Table 3.4, servo ID 2 is not sufficient to differentiate normal pressure strokes from no-touch conditions. However, servos ID 1 and 3 torques are highly suited for use as the evaluators to estimate the pen's tip pressure for this stroke pattern.

Table 3.5 and Table 3.6 show the model of vertical upward stroke. As seen in Table 3.6, servo ID 2 shows no significant difference between normal pressure stroke and no-touch conditions. But servos ID 1, 3 and 4 torques may be used to estimate the pen's tip pressure for vertical upward stroke.

Table 3.7 and Table 3.8 show the model of horizontal right-left stroke. As seen in Table 3.8, it is not identical to the previous stroke types we discussed. For horizontal

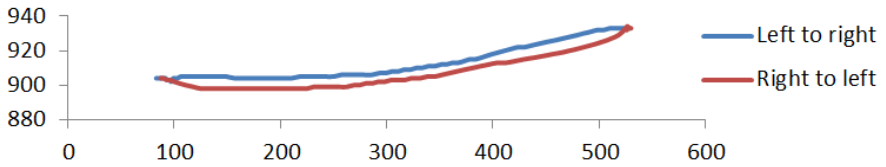


Figure 3.31: Medium pressure horizontal strokes.

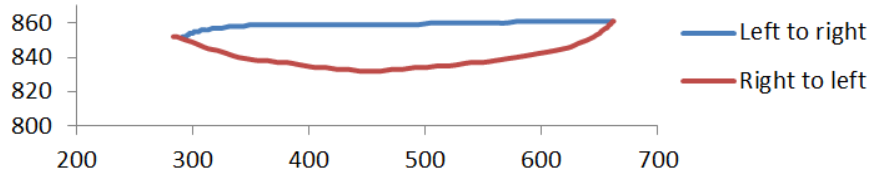


Figure 3.32: High pressure horizontal strokes.

right-left stroke, servo ID 1 is not sufficient to differentiate high pressure strokes from no-touch conditions. However, it is highly suited for differentiate these conditions from normal pressure. Servos ID 3 and 4 torques may be used to estimate the pen's tip pressure.

Table 3.9 and Table 3.10 show the model of horizontal right-left stroke. As seen in Table 3.10, servo ID 2 shows no significant result to differentiate normal pressure stroke from no-touch conditions like the other vertical strokes we have seen. Similar to the vertical upward stroke, servos ID 1, 3 and 4 torques may be used as the evaluators to estimate the pen's tip pressure for horizontal right-left stroke.

Based on these preliminary experiments, Table 3.11 shows the general TFC model to estimate the pen's tip pressure from servos' torque feedback. For simplicity of the implementation Servo ID 1 and 3 (shoulder and elbow) are mostly used. Figure 3.33 illustrates the control loop of the TFC model.

Table 3.3: Results of vertical stroke (downward) tests.

Vertical Strokes (Downward)												
Average Pressure	**No**				**0.67**				**1.00**			
Servo ID	1	2	3	4	1	2	3	4	1	2	3	4
Average	16.88	0.02	52.42	0.12	32.32	0.84	20.7	23.6	200.44	11.82	93.1	1.42
Std. Dev.	12.21	0.14	7.65	0.33	16.24	5.37	8.95	2.83	12.13	7.01	8.92	4.36

Table 3.4: Comparison of vertical (downward) stroke model.

	Servo ID			
Pressure Comparison	**1**	**2**	**3**	**4**
No vs. Normal	**2.96e-07**	*0.1430*[1]	**1.24e-34**	3.59e-48
No vs High	**6.73e-89**	2.26e-16	**2.65e-43**	0.0202
Normal vs. High	**3.27e-74**	4.05e-14	**2.57e-63**	3.41e-47

[1] Not significant to identify different pressure on pen tip.

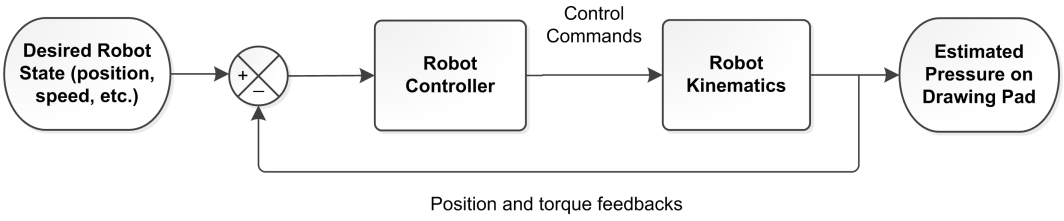


Figure 3.33: Closed-loop control diagram for TFC

Table 3.5: Results of vertical stroke (upward) tests.

Vertical Strokes (Upward)												
Average Pressure	**No**				**0.67**				**1.00**			
Servo ID	1	**2**	3	4	1	**2**	3	4	1	2	3	4
Average	76.04	0.12	33.36	0.2	32.42	1.08	11.96	23.52	40.96	58.58	46.42	18.78
Std. Dev.	11.65	0.33	6.08	0.4	18.22	3.99	10.12	3.39	3.08	18.42	8.34	5.21

Table 3.6: Comparison of vertical stroke (upward) model.

	Servo ID			
Pressure Comparison	**1**	**2**	**3**	**4**
No vs. Normal	**2.25e-24**	*0.048072*[1]	**2.08e-21**	**3.81e-44**
No vs High	**4.90e-28**	**9.74E-28**	**2.32e-14**	**3.67e-30**
Normal vs. High	0.000964	**2.19e-28**	**1.31e-33**	3.20e-07

[1] Not significant to identify different pressure on pen tip.

Table 3.7: Results of horizontal stroke (left to right) tests.

Horizontal Strokes (Left to right)												
Average Pressure	**No**				**0.67**				**1.00**			
Servo ID	**1**	2	3	4	1	2	3	4	**1**	2	3	4
Average	47.26	30.04	66.24	26.52	23.64	36.34	21.86	23.06	47.44	54.74	41.02	11.66
Std. Dev.	7.77	12.87	7.76	4.68	1.79	14.99	4.04	5.41	2.55	12.12	3.18	5.82

Table 3.8: Comparison of horizontal stroke (left to right) model.

	Servo ID			
Pressure Comparison	**1**	**2**	**3**	**4**
No vs. Normal	**5.90E-28**	0.013217	**9.69E-49**	0.00046
No vs High	*0.438413*[1]	**1.15E-16**	**2.76E-31**	**3.77E-25**
Normal vs. High	1.56E-69	6.08E-10	**3.48E-45**	**3.05E-17**

[1] Not significant to identify different pressure on pen tip.

Table 3.9: Results of horizontal stroke (right to left) test.

Horizontal Strokes (Right to left)												
Average Pressure	**No**				**0.67**				**1.00**			
Servo ID	1	**2**	3	4	1	**2**	3	4	1	2	3	4
Average	45.14	35.34	66.24	26.36	18.52	35.56	20.84	17.06	25.72	11.88	88.86	1.06
Std. Dev.	6.55	15.14	7.76	6.62	4.21	13.84	4.1	5.45	14.91	7.07	13.83	3.82

Table 3.10: Comparison of horizontal stroke (right to left) model.

	Servo ID			
Pressure Comparison	**1**	**2**	**3**	**4**
No vs. Normal	**8.33e-40**	*0.469855*[1]	**1.33e-49**	**7.56e-12**
No vs High	**1.94e-12**	2.95e-15	**4.88e-16**	**2.03e-37**
Normal vs. High	0.000871	**4.93e-17**	**1.25e-39**	**7.85e-30**

[1] Not significant to identify different pressure on pen tip.

Table 3.11: Torque feedback control model.

Stroke Type	Torque (τ_i) threshold conditions for normal pressure	
	Low Pressure (No-touch)	High Pressure
Vertical downward	ID1: $\tau_1 < 25$	ID1: $\tau_1 > 50$
Vertical upward	ID1: $\tau_1 > 50$	ID1: $\tau_1 < 50$ and ID3: $\tau_3 > 40$
Horizontal left-right	ID1: $\tau_1 > 30$ and ID3: $20 < \tau_3 < 35$	ID3: $\tau_3 > 20$
Horizontal right-left	ID1: $\tau_1 > 40$ and ID3: $20 < \tau_3 < 35$	ID2: $\tau_2 > 35$ and ID3: $\tau_3 > 20$

3.5.2 Image-based Visual Servoing (IBVS) control

As discussed earlier, a visual servoing approaches is classified based on the type of feedback it perceived. In my implementation, I extracted the line segments of the drawing pad's pen-and-ink sketch. Based on this information, it guides Betty during the drawing process to correct the drawn errors (e.g. line segments pose (two end-points) and completeness). Hence, an imaged-based visual servoing (IBVS) approach is implemented in my thesis to accomplish this task.

IBVS involved manipulation of a set of image features (object) from an acquired image based on their coordinates by computation of the *image Jacobian* [55]. In general, the Jacobian matrix (J) is defined as a matrix based on the motion of the image feature. J is used to set the desired pose of manipulator as seen in equation 3.21 [47; 55].

$$\dot{f} = J\dot{q} \tag{3.21}$$

Where $\dot{f}$ represents the motion of the image feature in consecutive images and $\dot{q}$ denotes the six dimensions velocities, $[\dot{x}, \dot{y}, \dot{z}, \dot{w}_x, \dot{w}_y, \dot{w}_z]$: three translations and three rotations of the feature with respect to the camera frame [21]. Based on the unit focal length perspective projection model, the relationship between the robot arm and the image feature is shown as equation 3.22 [47; 55].

$$\begin{bmatrix} \dot{x'} \\ \dot{y'} \end{bmatrix} = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} & xy & (1+x^2) & -y \\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -(1+y^2) & xy & x \end{bmatrix} \tag{3.22}$$

During the drawing task, the pen's tip is orthogonal to the surface of the drawing

pad. Hence, no orientation adjustment is required during the servoing. The image features only need to be translated to the desired pose with the three translation $[\dot{x}, \dot{y}, \dot{z}]$ workspace. Therefore, the Jacobian can be simplified as equation 3.23.

$$J = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} \\ 0 & \frac{1}{Z} & -\frac{y}{Z} \end{bmatrix} \tag{3.23}$$

IBVS expresses the control error function directly in 2D image space [55]. The aim of all vision based control schemes is to minimize the error $e(t)$, which is typically defined by equation 3.24 as we have seen in Figure 2.6.

$$e(t) = f - f' \tag{3.24}$$

f is a vector of n visual features based on image measurements (e.g., the image coordinates of interest points, or the parameters of a set of image segments) and a set of parameters that represent partial knowledge about the system (e.g. camera intrinsic parameters or three-dimensional object models) [82]. The vector f' is the desired values of these features. In the drawing tasks, the simplest model for the controller is its drawing deviation. And there must be a relationship between the time variation of each variable in equation (3.24).

Based on Kragic et al.'s [55] and Chu et al.'s research [21], the most common proportional control approach is applied to generate the control signal for the robot.

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = K_p J^{-1} \cdot e(t) \tag{3.25}$$

J^{-1} is the pseudoinverse of the image Jacobian and K_p is the proportional gain. The visual servoing scheme continues until the evaluated error is less than the given threshold pixels limit on the image.

Figure 3.34 illustrates the proposed framework of the visual processing pipeline for the IBVS system.

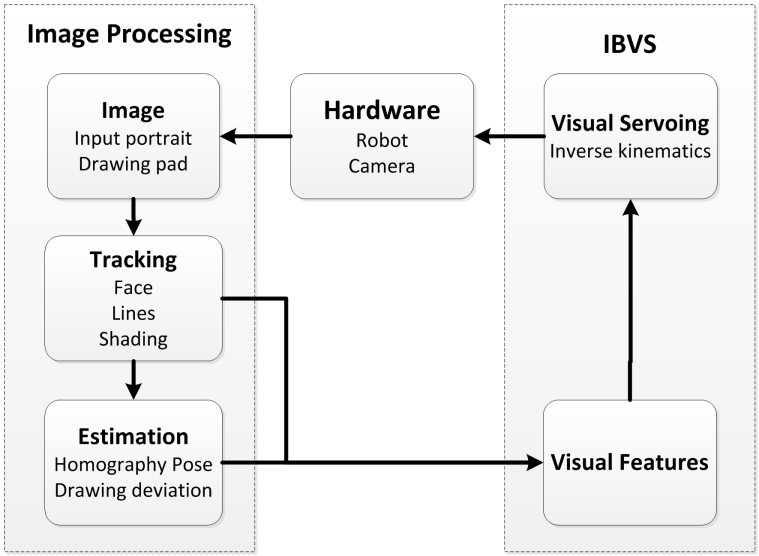


Figure 3.34: General framework of IBVS for drawing task.

Let us assume the camera position is static and the height of the drawing pad is known, as shown in Figure 3.35. A number of feature lines are tracked and used to generate a vector of current measurements, $f_1 f_2$. The vector of desired reference

measurements is denoted $f_1' f_2'$. The error function for the drawing deviation is defined as a function of the distance between these measurements, $e = f - f'$. This error function is then updated in each frame and used together with the inverse kinematics to estimate the control input to Betty.

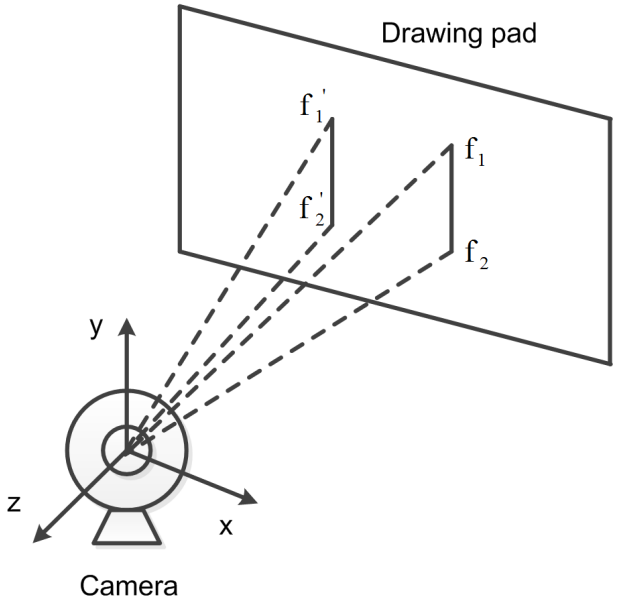


Figure 3.35: A schematic overview of image based visual servoing.

Drawing pad detection and sketch extraction

Drawing pad location is a crucial information in the IBVS implementation. Based on this information, it constructs the image plane on a Cartesian space to extracts content on the drawing pad.

Figure 3.36 shows the overall process to detect the drawing pad location and to extract its content within the drawing area. First, I applied the Canny edge detector to the input image to obtain the line image as seen in Figure 3.38(b). With the wide

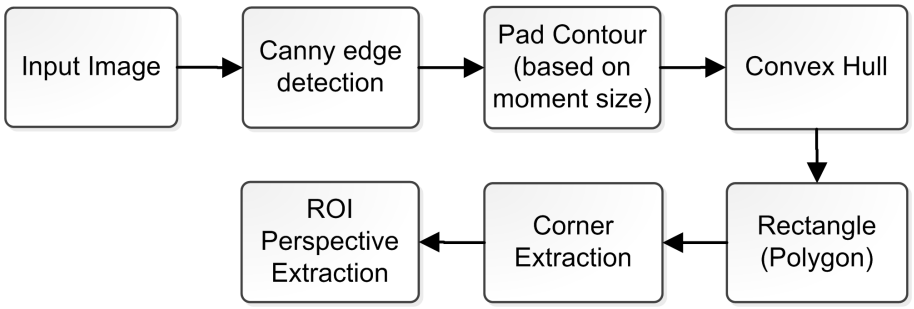


Figure 3.36: Flowchart for drawing pad detection.

range of edges in the image, drawing pad contour is extracted based on its moments size by using the *moments* function in OpenCV. It allows computation of all moments as seen in equation 3.26 [87].

$$m_{ij} = \sum_{x,y} (array(x, y) \cdot x^j \cdot y^i) \tag{3.26}$$

Then I used a rectangular convex hull algorithm as the minimum bounding box of the rectangle drawing area to compute its four corners' coordinates in Figure 3.38(d). Last, based on the detected corners vertices, a perspective transformation is performed to map the deformed drawing pad (region of interest) to its destination image. The mapping is done by *getPerspectiveTransform* and *warpPerspective* in OpenCV library [86]. Equation 3.27 [86] shows the calculation of the 3×3 perspective transform matrix from four pairs of the corresponding points $i = 0, 1, 2, 3$. The destination image vertices are represent as $dst(i) = (x'_i, y'_i)$ and the source image vertices are referred as $src(i) = (x_i, y_i)$.

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = map\_matrix \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.27}$$

Then I used the *warpPerspective* function to perform a perspective transformation of a source image to a destination image using the specified 3×3 matrix, M. Mapping of the drawing pad dimension is described in equation 3.28 [86]. Algorithm 3.4 is the implementation of drawing pad detection. Figure 3.38 shows an example of extracted drawing pad area. Figure 3.37 shows the perspective transformation of drawing area to correct distortion and due to viewing angle of Betty right eye.

$$dst(x, y) = src(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}) \tag{3.28}$$

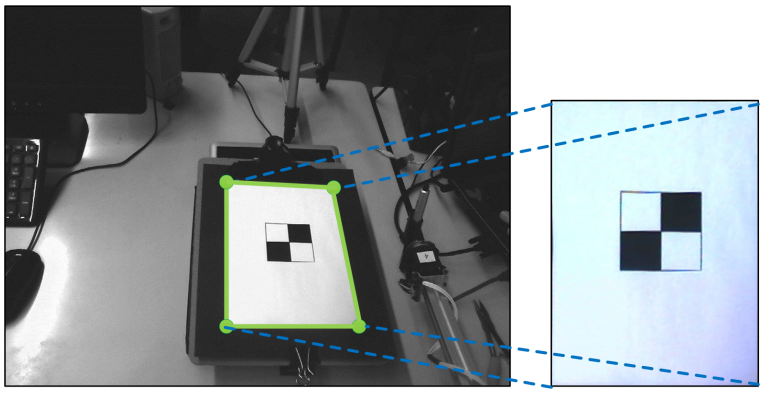


Figure 3.37: Drawing area perspective transformation: perspective transformation is applied to the drawing area (region of interest) to extract drawing content.

Algorithm 3.4 Extract sketch from drawing pad

Require: input image

1: Define contour vectors
2: *cvtColor()*: convert color space to grayscale
3: *Canny()*: compute Canny edge detector
4: *findContours()*: find contours of edges
5: Define image moment and convex hull vectors based on contour's size
6: **for each** contours **do**
7: Compute image moments of the contours with *moments()*
8: Compute convex hull vectors with *convexHull()*
9: Approximate a polygon with *approxPolyDP()* from the convex hull vectors
10: **if** Number of corners = 4 **then**
11: Arrange four corners in counterclockwise order: top-left, bottom-left, bottom-right, top-right
12: Verify the rectangularity of polygon based on parallelity and equivalent of opposite edges
13: **end if**
14: **break**
15: **end for**
16: Define region of interest (ROI) of the drawing pad
17: Calculate the perspective transform matrix from four pairs of the corresponding corners and ROI using *getPerspectiveTransform()*
18: Apply a perspective transformation to the image using *warpPerspective()*
19: **return** Drawing pad ROI

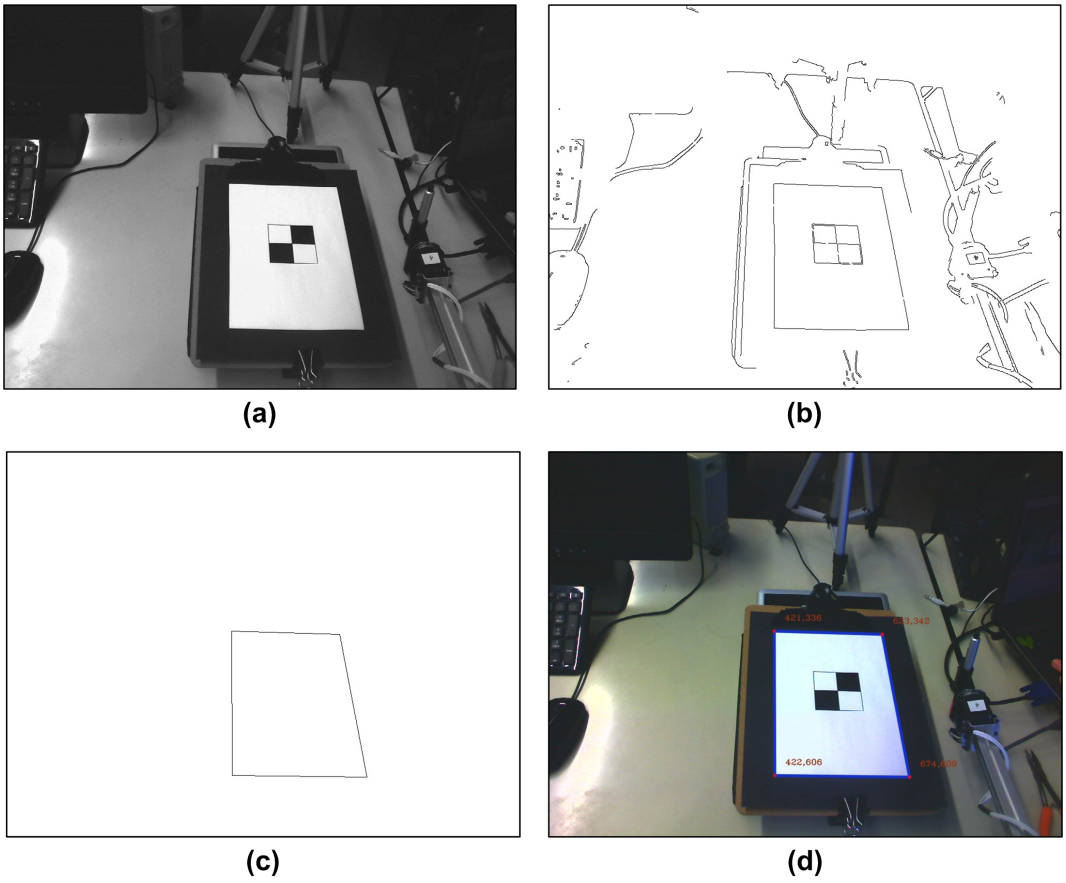


Figure 3.38: Image pipeline for drawing pad detection. (a): input image; (b): Canny line detection output; (c): pad contour of drawing area is extracted; (d): four corners are estimated based on rectangular convex hull.

Pen detection

Pen detection is important for the implementation of drawing task with the IBVS approach. Due to the limited feature of a pen in the drawing scene, pen position (x- and y-coordinates) on the image plane is estimated by using a pink colour marker on its tip. Figure 3.39 shows the flowchart of this implementation.

RGBA (referring to red, green, blue and alpha for opacity) colour space is not easy to perform interpretation and segmentation for what human perceive as a constant colour. The first step of the implementation is colour space conversion from the default four-channel RGBA to a three-channel HSV (referring to hue, saturation and value) and it also called HSB (B refer as brightness) which is more commonly used for image segmentation [74]. Figure 3.40(b) shows the output in HSV space. Colour is defined by hue and saturation and the value (V) is used to describe the brightness. If the brightness is low then the colour can be considered as black [65]. Therefore, HSV is better at capturing the colour intensity for the segmentation compared to the RGBA colour space.

Next, any pixels in range of the pink colour are filtered and converted to a binary image. Then the algorithm calculates all moments' sizes of the dark pixels. It represents the contour of the pink coloured marker (pen's tip) as shown in Figure 3.40(c). Figure 3.40(d) shows the final output where the central moments (based on equation 3.29 [87])of the selected tip's contour; with its vector of points is returned as the estimated pen position.

$$mu_{ij} = \sum_{x,y}(array(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i) \tag{3.29}$$

$$\text{where } \bar{x} = \frac{m_{10}}{m_{00}} \text{ , } \bar{y} = \frac{m_{01}}{m_{00}}$$

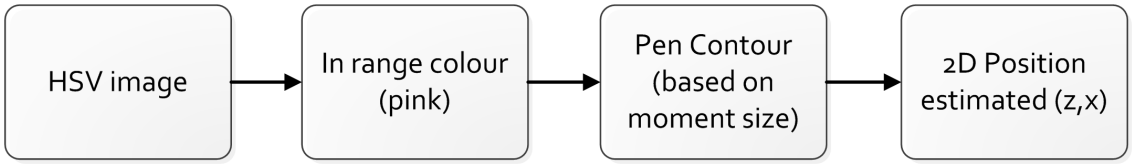


Figure 3.39: Flowchart for pen detection

Algorithm 3.5 shows the detail of the pen's tip detection process and usage of OpenCV functions to find contours of the filtered image and compute their image moments.

Algorithm 3.5 Estimate pen X-Y position

Require: input image

1: Define line segment and image moment vectors
2: *cvtColor()*: convert color space to HSV
3: *inRange()*: find in range coloured segments
4: *findContours()*: find contours of filtered segments
5: **for each** contours **do**
6: Compute image moments of the contours with *moments()* function
7: **if** image moment in range of limits **then**
8: Compute pen centre as suggested in equation 3.29
9: **break**
10: **end if**
11: **end for**
12: **return** Pen centre coordinate

Figure 3.41 shows the example images of occlusion detection that applied in the implementation. The occlusion detection is based on the pen position and a wrong

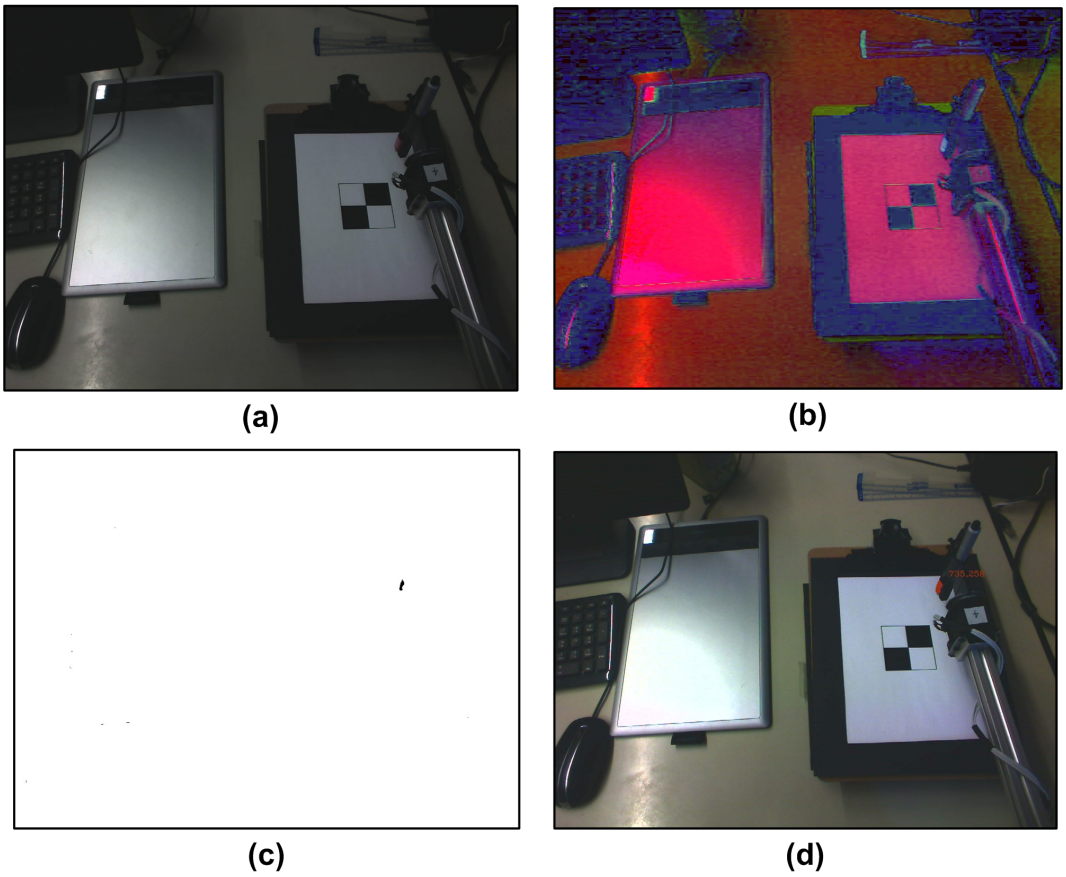


Figure 3.40: Image pipeline for pen detection. **(a)**: input image;**(b)**: HSV image; **(c)**: extracted pen contour based on colour marker on the pen; **(d)**: estimated position on the drawing pad plane.

corner or incompleteness of the perceived drawing area. It avoids unnecessary features extraction in the drawn sketch.

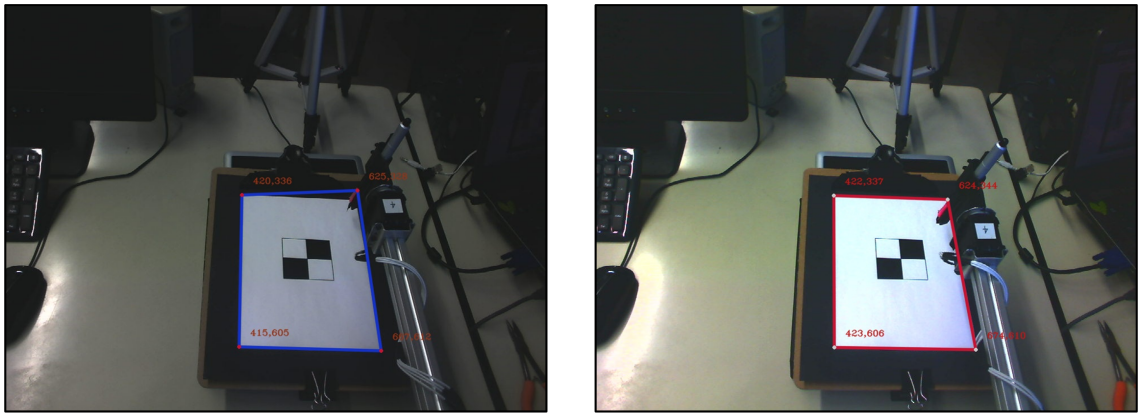


Figure 3.41: Pad (corners) errors detection. **(a)**: fail to avoid wrong corner; **(b)**: wrong corner is avoided and drawn sketch is not extracted to avoid inclusive of Betty's hand in the sketch while occlusion occur.

3.5.3 Sketch lines matching

Sketch line matching is the process to match line segments based on geometrically possible pairs of generated sketch and actual drawn lines segments to compensate for drawing errors for the IBVS approach. This technique is adapted from the eEDLines approach. The geometry of the drawing pad is mapped to the generated sketch dimension in 2D space (x- and y-coordinates) as shown in Figure 3.42 with directional scaling from Figure 3.42(a) to Figure 3.42(b). Its projective transformation matrix is shown in equation 3.30, where $s_x = W_{sketch}/W_{image}$ and $s_y = H_{sketch}/H_{image}$. Then the extracted ROI image is sharpened with a Gaussian blur filter (*cv::GaussianBlur*) to enhanced the line features on the image.

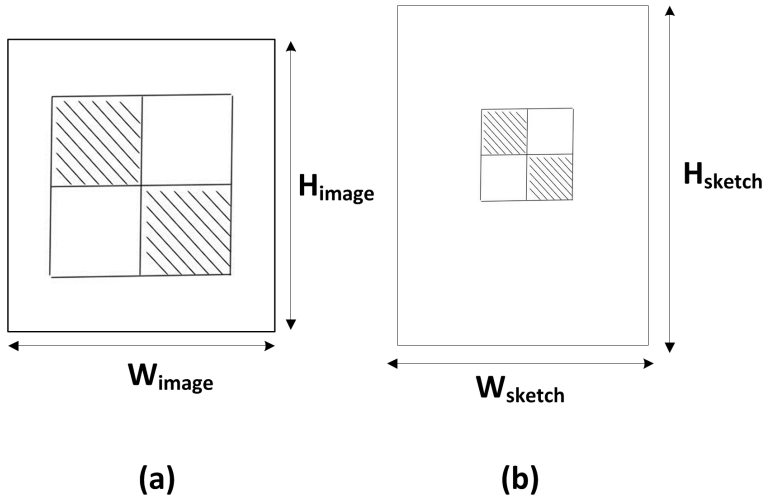


Figure 3.42: Image mapping from input image to sketch for drawing pad. (a): input image (b): mapped sketch based on drawing pad dimension

$$\begin{bmatrix} x_{image} \\ y_{image} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{sketch} \\ y_{sketch} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x_{sketch} \\ s_y y_{sketch} \\ 1 \end{bmatrix} \tag{3.30}$$

Each of the line segments will be redrawn based on the correction of its position error. Errors of two end points of a line segment conveys the errors of angle and length of the matching line segment to its desired baseline. It provides sufficient information to redraw the unmatched line segment. However, if the matching is unsuccessful after several trials, the drawing process will move on to the next line segment to avoid infinite loop of single line segment matching. I use five trials as the termination condition for each line segment.

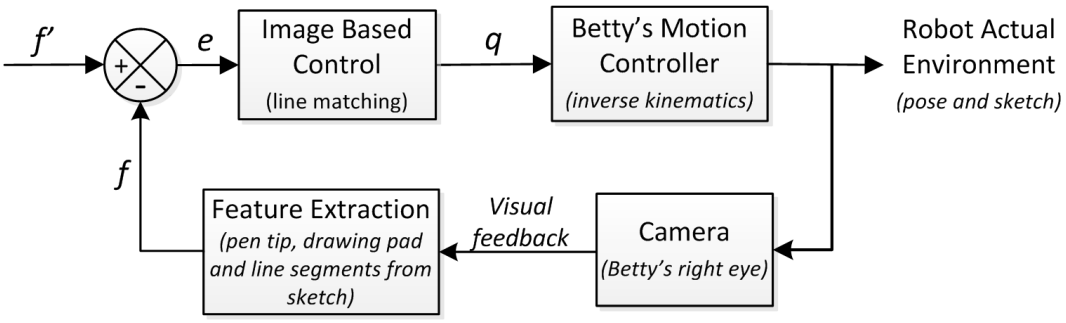


Figure 3.43: Closed-loop control diagram for IBVS

As discussed in the previous chapter, the error function is defined simply as the difference between the current (or the closest line segment) and the desired line segment feature $e(t) = f - f'$. Based on their two end points, as seen in Figure 3.44; f is the pose of the current line segment ab and f' is its desired pose which is $a'b'$. Hence, the current errors $e(t)$ of endpoints a and b in 2D space could be defined as the Manhanttan distance between them:

$$e_a(t) = dist(a, a') = [e_a^x, e_a^y] = [(x_a - x'_a), (y_a - y'_a)] \tag{3.31}$$
$$e_b(t) = dist(b, b') = [e_b^x, e_b^y] = [(x_b - x'_b), (y_b - y'_b)]$$

Then these errors are passed to the inverse kinematics routine to change the joint angles to desired pose with a P controller. Equation 3.32 shows the proportional control function where K_p is a empirically tuned gain. The third dimension z could only be estimated based on no-touch events. The process of the line segments matching is shown in algorithm 3.6. No transfer of errors from one stroke to the next. It avoids

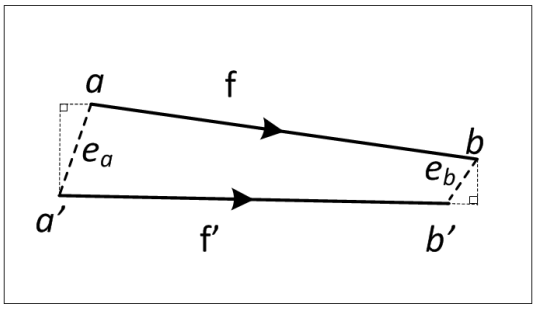


Figure 3.44: IBVS based on line segment feature.

false corrections due to shifting of drawing pad during the drawing task.

$$a^x_{IK}(t) = a^x_{IK}(t-1) + K_p * e^x_a \tag{3.32}$$

$$a^y_{IK}(t) = a^y_{IK}(t-1) + K_p * e^y_a$$

$$b^x_{IK}(t) = b^x_{IK}(t-1) + K_p * e^x_b$$

$$b^y_{IK}(t) = b^y_{IK}(t-1) + K_p * e^y_b$$

Algorithm 3.6 Match line segments

Require: line segments from drawing pad, S_{pad} and line segments from sketch, S_{sketch}

```
Read and draw each S_sketch
for each line segments from S_sketch do
  if S_sketch and S_pad are approximately collinear then
    Compute the closest distance between S_sketch and S_pad with algorithm 3.2
    if S_sketch and S_pad distance < threshold then
      Calculate two end points distances between S_sketch and S_pad, p1_dist and p2_dist
      if p1_dist and p2_dist < lower threshold then
        Flag the S_sketch as match
        Return line match
      else
        if p1_dist and p2_dist < within lower and upper thresholds then
          Return the errors of end points, e1_x, e1_y, e2_x and e2_y
        else
          Return no line match
        end if
      end if
    end if
  end if
end for
```

3.5.4 Hybrid Control

Hybrid control is an integrated approach based on torque feedback control and visual servoing. However, most researchers use force sensors mounted on the robot's end-effector to estimate its pose [5; 20; 45; 59; 75]. Here, I define hybrid control as the combination of the torque feedback control and image-based visual servoing. It estimates the pressure errors on the pen's tip and the 2D pose of the line segment features on the drawing pad. The main goal of this combination setup or hybrid control is to combine their advantages, while overcoming their shortcomings in a 3D workspace. Where depth error is estimated by the pressure and paper shifts is corrected by drawn line segment features. Both the task quality (in the sense of

increased robustness), the accuracy, and the range of feasible tasks have to increase reasonably with integrated vision and torque feedback control. The results of the implementation is elaborated in Chapter 4.

3.6 Summary

This chapter described the core of my research and how it was implemented in both hardware and software. It explained the kinematic model I have designed to manipulate Betty's arm. I introduced the TFC model to estimate the pen's tip pressure based on the servos' basic torque feedback. I also developed the algorithms to the represent a sketch as a series of line segment vectors which serve as visual features for visual servoing in the compensation of drawing deviation section.

Chapter 4

Evaluation

4.1 Introduction

In this chapter, I designed various experiments to evaluate different aspects of my research. I tested efficiency of two automatic sketch generator methods, namely FNTG and eEDLines. Next, I evaluated the drawing feasibility based on three implementation: TFC, IBVS and their hybrid control. The data collected from a Bamboo tablet was analysed and was discussed in this chapter. Finally, I evaluated the drawn outputs of open-loop (feedforward) and closed-loop (hybrid feedback) based on the similarity test to their inputs. SURF and pixel matching evaluation methods were used in this experiment.

4.2 Automatic Sketch Generator

The ability to generate a sketch using FNTG was tested by comparing different parameter settings of the algorithm. Then I compared the number of generated edges based on these settings. Next, I evaluated my eEDLines algorithm based on two aspects, its processing time and the number of generated line segments. Its performance was compared to established algorithms, e.g. LSD and EDLines.

4.2.1 Furthest Neighbour Theta-graphs (FNTG)

In this experiment, several settings of different parameters were tested e.g. number of cones; to see the capability of FNTG in sketch generating. It is clear that the accuracy of estimated edges is greatly affected by the bounded perimeter of the discs. It could be too fine (too many insignificant edges) or too coarse (loss of detail, such as the eyeglass frames in Figure 4.1) to correctly represent the portrait. Figure 4.1 shows several sketches generated by different parameter configurations of the furthest neighbour theta-graph.

Table 4.1 shows the results of different inner and outer discs' radii and their respective total number of edges based on various numbers of cones, k_θ. These results were obtained from an input image with a 25:100 Canny's thresholds ratio, resulting in 5450 pixels before thinning and 1607 pixels after thinning. It showed that the number of edges could be reduced according to the inner and outer radii ratio, where a smaller range of disc perimeters produced fewer edges.

The comparison chart shown in Figure 4.2 illustrates the effectiveness of the fur-

Figure 4.1: Different sketch outputs based of discs' radii ratio r_{inner}:r_{outer}. **(a)**: 0:10, **(b)**: 9:10 and **(c)** 14:15 with four cones, $k_\theta = 4$.

thest neighbour theta-graph to reduce the total number of output edges with different values of k_θ and discs' radii. For each k_θ, the number of edges is reduced significantly for smaller distance of a inner and a outer discs, e.g. 9:10 ratio. Although the number of edges grew approximate linearly when the number of cones increased, its setting was chosen so that a visually reasonable output was obtained. Removing insignificant edges from the sketches reduced the total number of drawing motions required by Betty, allowing it to complete the sketching in less time.

Based on a visual evaluation of the output (e.g. portrait **(b)** in Figure 4.1), I found that with $k_\theta = 4$ and a discs' radii ratio, r_{inner}:r_{outer}, of 9:10 produced reasonable output that faithfully captured and presented the portrait's detail (e.g. the glasses frame). Figure 4.3 shows several outputs from a few input images; namely, checker, *molecules* [98] and *chairs* [98] under the chosen parameters. I choose these images because they are established in the literatures, have good representation, and complexity of graphic that is suitable for evaluation. In these examples, FNTG

Table 4.1: FNTG Output edges of k_θ-cone comparison based on Canny's threshold-25:100, input pixels- 5450 and output pixels- 1607.

k_θ-cone	Inner radius (Pixels)	Outer radius (Pixels)	Output edges
2	0	10	3427
2	5	10	2096
2	9	10	1264
4	0	10	4137
4	5	10	3323
4	9	10	1567
4	14	15	1889
6	0	10	5519
6	5	10	4149
6	9	10	1709
8	0	10	6718
8	5	10	4785
8	9	10	1786

showed the capability to represent line sketches in most cases, but not the checker input due to the simple single line pattern in the image. Furthermore, thousands of edges were too many for Betty to draw in a short period of time. Therefore, another sketch generator, namely eEDLines was tested in the next subsection.

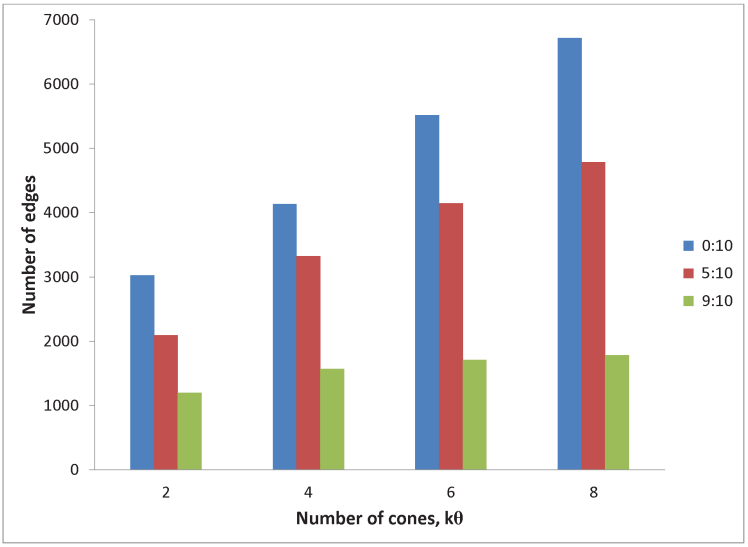


Figure 4.2: Comparison of number of edges in the output images.

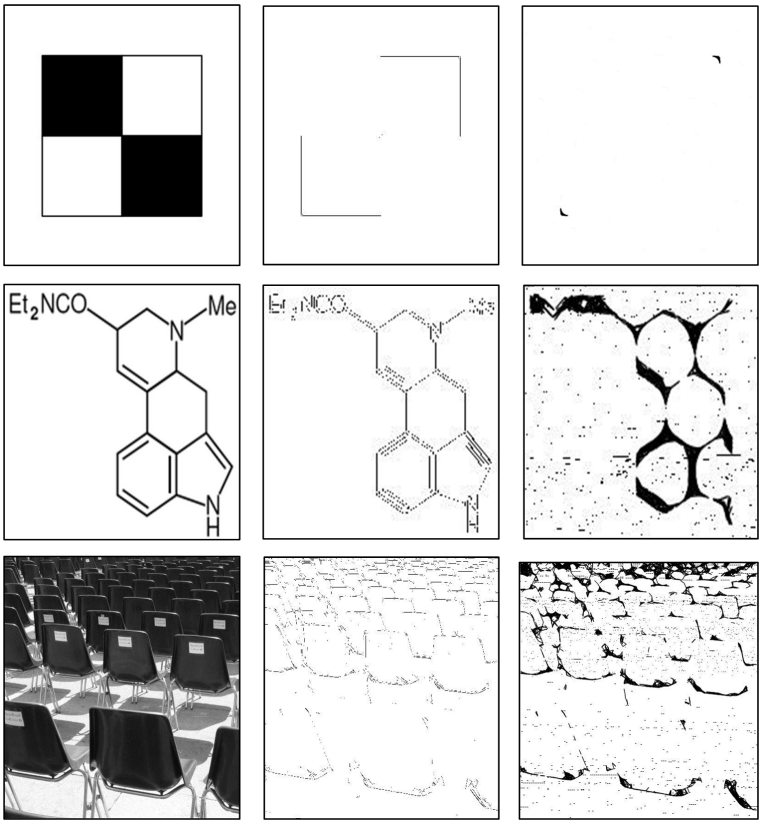


Figure 4.3: Outputs from different images. Output edges from top to bottom (checker, molecules and chairs): 26, 759 and 3295. Left to right: input image, Canny edge, and FNTG 9:10 ratio

4.2.2 Extended Edge Drawing Lines (eEDLines)

In this subsection, I compared the performance of the proposed line segment detector algorithm, extended Edge Drawing Lines (eEDLines), to Progressive Probabilistic Hough Transform (PPHT) [35; 70], Line Segment Detector (LSD) and the original Edge Drawing (EDLines) algorithm. I implemented the PPHT with OpenCV *HoughLinesP* function as descried in [70] with *rho*=1, *theta* = $\pi/180$, *threshold* = 40, *minLineLength* = 10 and *maxLineGap* = 10. *rho* is the distance resolution of the accumulator in pixel. *theta* is the angle resolution of the accumulator in radians and *threshold* is the accumulator threshold parameter for votes returned from those lines. *minLineLength* is the minimum rejected line length and *maxLineGap* is the maximum allowed gap between points on the same line to link between segments.

Figure 4.1 shows the outline of several sketches generated by different line detection algorithms and their input images namely *checker*, *portrait*, *molecules* [98] and *chairs* [98] with various complexity. Based on visual evaluation, the proposed eEDLines algorithm (as seen in the last column) produced a good representation of the sketches outline and successfully reduced the near-collinear line segments accurately in comparison to the other three algorithms.

Figure 4.5 shows the comparison of a number of line segments extracted based on these algorithms. PPHT performed poorly in most cases and was unable to represent the input images. LSD and EDLines produced good detection but as mentioned in Chapter 3, they both suffered from the co-linear lines issue. My eEDLines algorithm performed better compared to these algorithms. It successfully reduced the number of line segments by nearly 50% when compared to PPHT, LSD, and EDLines across

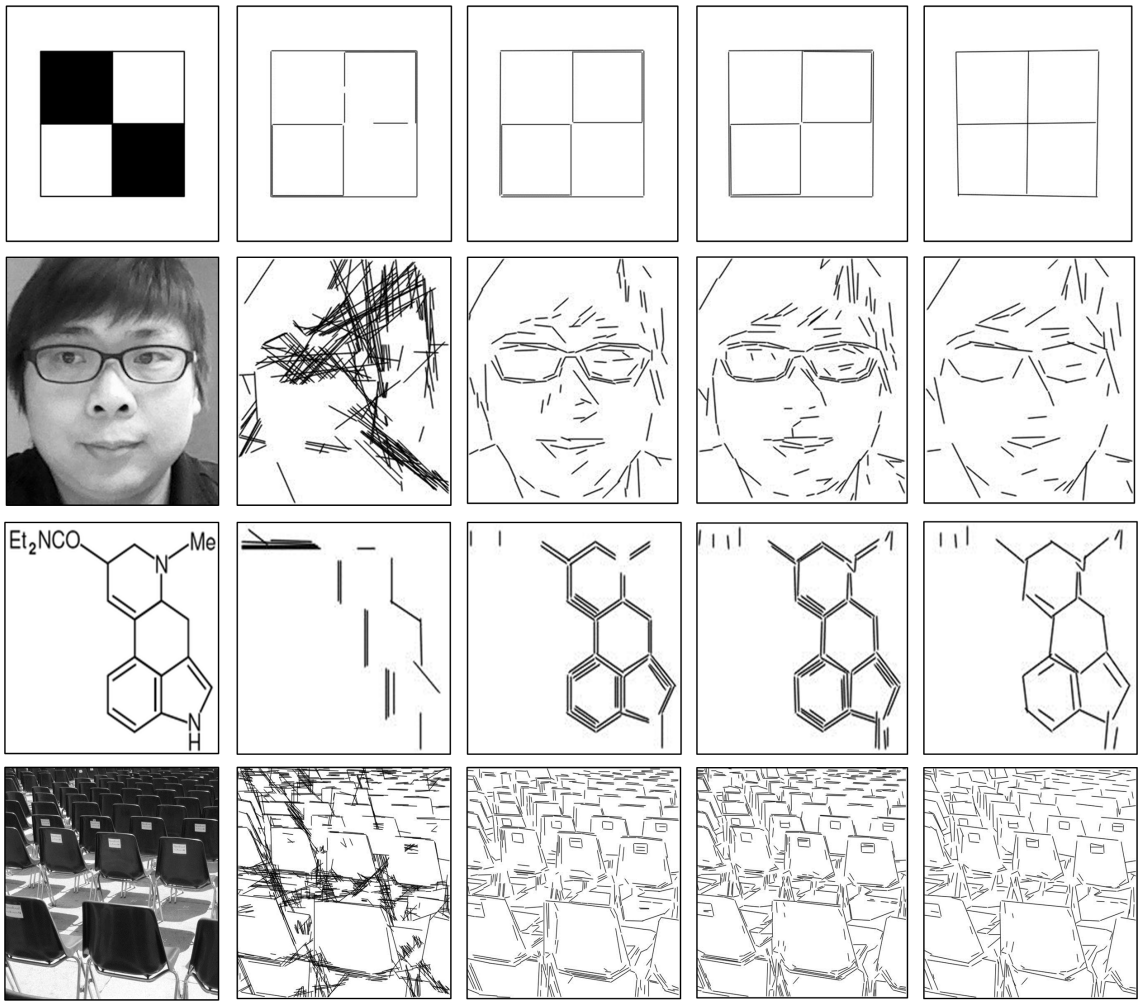


Figure 4.4: Comparison of generated sketch outline based on different algorithms. From left to right: Input Image, Probabilistic Hough Transform (PPHT), Line Segment Detector (LSD), Edge Drawing Lines (EDLines)and Extended Edge Drawing Lines (eEDLines). From top to bottom: *checker*, *portrait*, *molecules* and *chairs*.

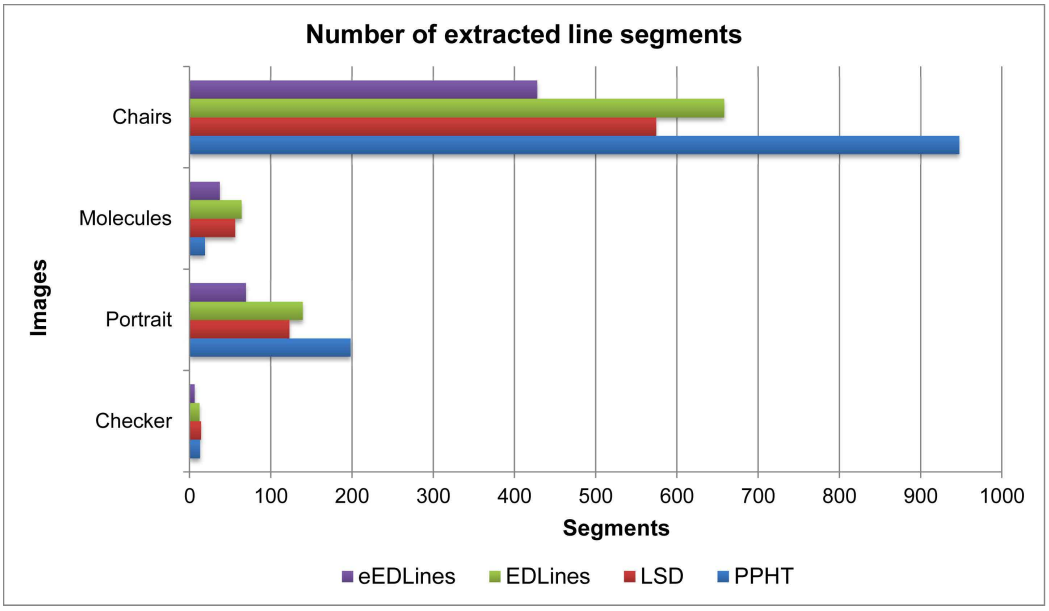


Figure 4.5: Number of line segments extracted based on different algorithms.

four test images while still retaining the representation of the original input images. The full result of this experiment is shown in appendix C, Table C.1.

Figure 4.6 shows the comparison of processing time between different line segments extraction algorithms. As seen in Akinlar and Topal's [2; 3] publications, EDLines outperforms both the PPHT and the LSD algorithms. As seen in Figure 4.4, despite the fact that eEDLines extended EDLines algorithm to remove the near-collinear line segments. It also increased the total processing time by the factor of $O(n \log n)$ since the line segments need to be sorted [78]. But, generally eEDLines retained its advantage over LSD and PPHT. The full result of this experiment is shown in appendix C, Table C.2.

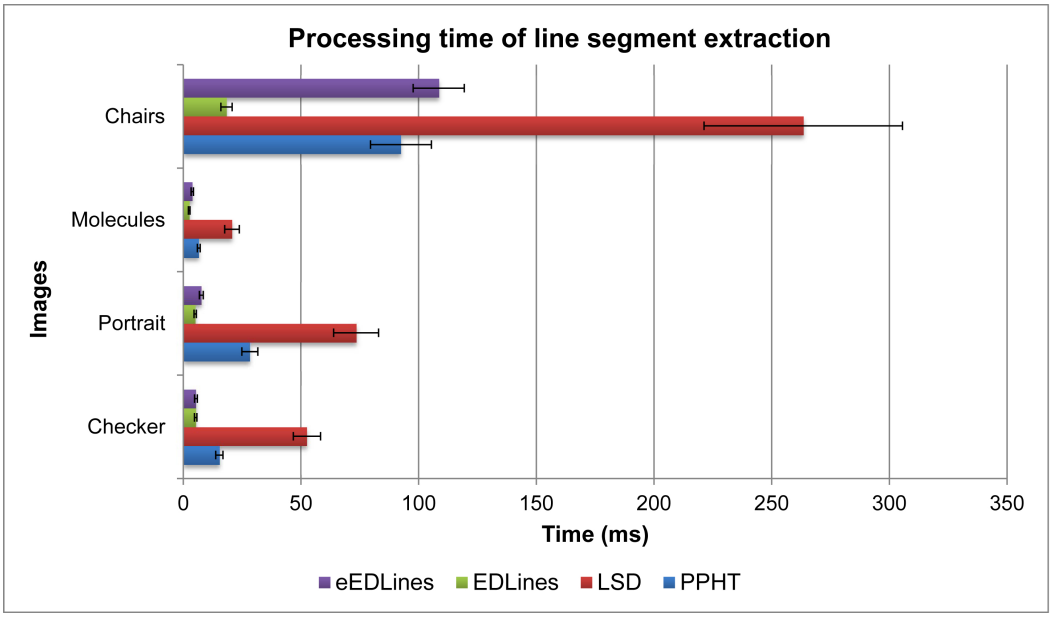


Figure 4.6: Processing time of different line segments extraction algorithms. Error bars indicate the standard deviation.

4.3 Drawing Experiments

In this experiment, I evaluated the feasibility of my implementation of feedback control based on their drawn outputs. Figure 4.7 shows the experimental setup of Betty. During the experiments, Betty used its right arm to draw. I used sinusoidal interpolation for its motion trajectory. Betty used its right eye (Logitech webcam) for visualisation. I placed a Wacom Bamboo tablet (CTH670M) under the paper to measure position and pressure on the drawing pad. The measures of Bamboo tablet were only used to established the reliability of the experiments but did not provide any feedback control to Betty. The drawing area is based on the specification provided by the manufacturer, the accuracy of a Bamboo tablet is ±0.5 mm on a 217 mm x 137 mm active area [34].

Figure 4.7: Experimental setup using a Wacom Bamboo tablet (CTH640M), Betty, camera and pen.

4.3.1 Position and torque experiments

In these experiments, I established the tests for position repeatability and torque reliability as seen in Figure 4.8. In the repeatability test, Betty drew four precise points repeatedly (30 times for each point) to show that it can visually reposition the pen accurately as seen in Figure 4.9. Table 4.2 shows that Betty successfully kept a small distance error (< 3 dot units) in this experiment where 3 dot units is approximately 1.5 mm based on the Bamboo tablet's accuracy specification. In the torque reliability test, Betty drew two types of strokes, vertical strokes (upward and downward) and horizontal strokes (left-to-right and right-to-left) as seen in previous chapter. The experimental results showed that by way of different stroke patterns, a significant difference in torque feedback can be measured. It allowed for a significant

estimate of the pressure of the pen's tip.

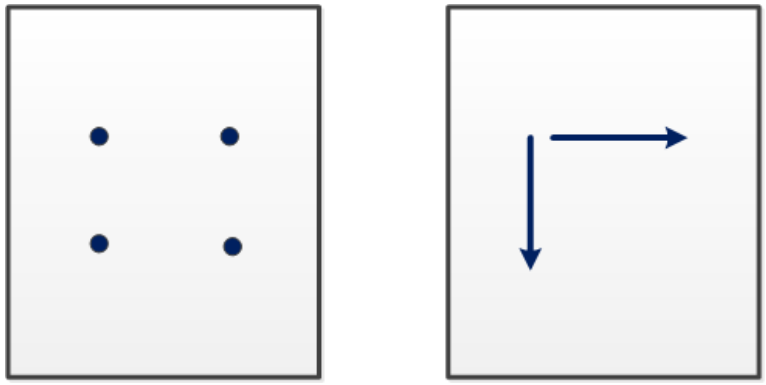


Figure 4.8: ***Left***: Position repeatability test and ***Right***: Torque reliability test.

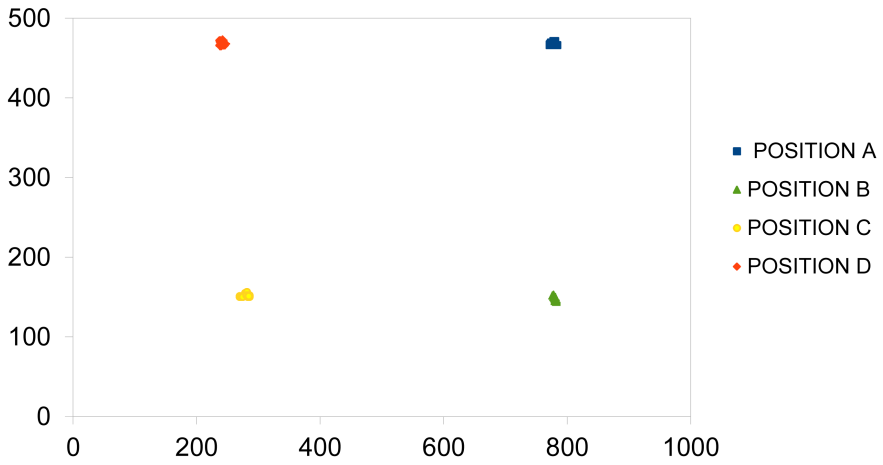


Figure 4.9: Repeatability test

After the position and torque tests, I evaluated three implemented feedback approaches that we have seen, namely TFC, IBVS and hybrid control. The efficiency of each approach was tested with a three-stroke task. Then I examined the overall

Table 4.2: Position repeatability test result.

	Average Position		Error from mean	Std. Dev.	t-test
	x	y			(a = 0.05, 3-pixel)
Position A	776.46	467.21	2.40	1.4	0.000088
Position B	241.89	467.62	2.75	1.36	0.036366
Position C	282.90	151.41	2.20	1.48	0.000187
Position D	780.10	148.47	1.59	1.21	0.000187

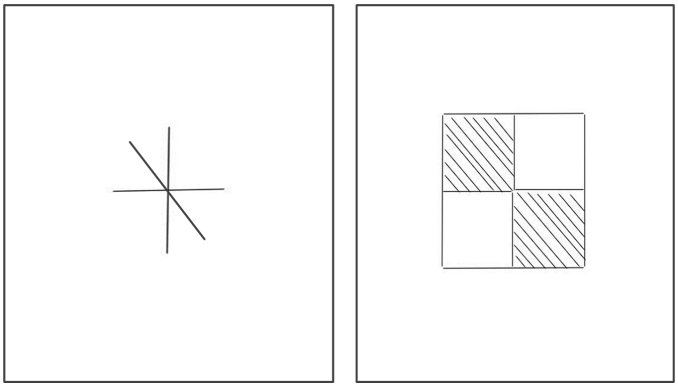


Figure 4.10: Generic drawing tasks. **Left**: vertical, horizontal and diagonal lines and **right**: checkers grid for drawing experiments in torque feedback control, image-based visual servoing and hybrid control

performance of the accuracy relative to the prior feedforward (open-loop) approach in more complex experiments, e.g. draw a checker grid as shown in Figure 4.10.

For each implementation, two measures were employed to evaluate the drawing deviation correction which were: the error in the drawing pressure and the position error of the pen's tip. For each experimental trial, I recorded the average error per-stroke in pixels. To compute the errors, I compared the line segments on the drawn result to the actual input sketch (e.g. outlining and shading) by the deviation of each pixel (endpoints) that map into the Bamboo tablet. In each of the measures, their values were then summed up and divided by the number of strokes to give the average error in the drawing process for each stroke. The level of error gave a good indication of how well each of the feedback techniques improved the final

drawing result. Meanwhile, I measured the total number of required correcting strokes to establish a drawing task to determine the efficiency of the deviation correction approaches.

For pressure measures, I recorded the 0 to 1024 pressure level on the Bamboo tablet within the scale of 0.00 to 1.00. The total number of required correcting strokes to finish a drawing task and the changes in error over time were sampled. However, due to different implementations and set up of the hardware, the experimental designs were varied. Such as, in visual feedback experiments, the pen's tip prevented the tablet touch-pen from touching the tablet. Hence, no pressure measures in this kind of experiments.

4.3.2 Torque Feedback Control Experiment

In this experiment, I used the generic drawing tasks to evaluate efficiency of torque feedback control (TFC). I used two initial conditions, no-touch (0.0) and high pressure (1.0) on the Bamboo tablet to justify the feasibility of TFC to correct these errors. Betty repeated 30 times for each vertical (upward and downward), horizontal (left-right and right-left) and diagonal lines (left-right with downward) based on these two initial conditions for a total of 150 trails. For these trials, the average pressure on the Bamboo tablet and measured torque reply for each stroke were recorded. I measured the error in pressure for each stroke based on the TFC model using initial errors of "high pressure", "low pressure" (no-touch) and desired "normal pressure" as seen in the previous chapter. The ideal average "normal pressure" experience on the Bamboo tablet should be at the level of 0.7 ± 0.1. A correction of 10 mm was applied when the

measured torque was too low or too high.

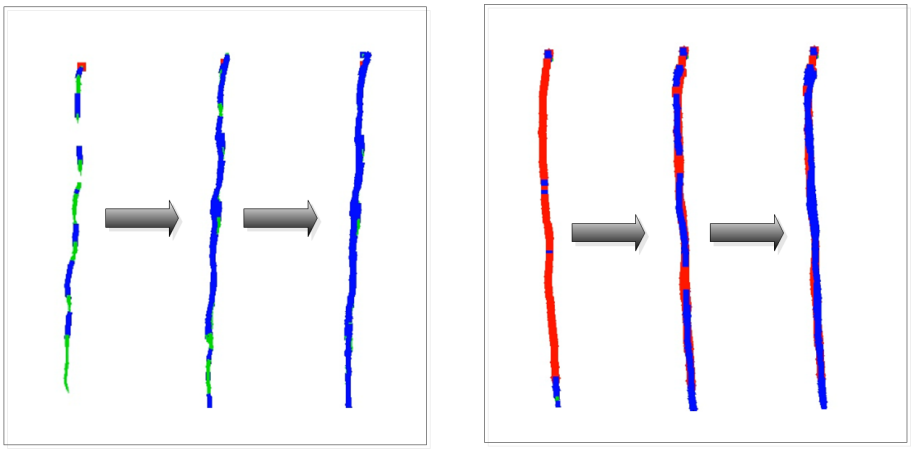


Figure 4.11: Stroke pressure correction based on torque feedback control. **(a)**: low to normal **(b)**: high to normal. Green, blue and red were used to indicate low, normal and high pressure measurements respectively.

Figure 4.11 shows the different pressure applied to the tablet. In Figure 4.11(a), it started with initial lower pressure strokes until it was corrected by the TFC model in two strokes. Similarly, the system corrected the initial high pressure error in two strokes as seen in Figure 4.11(b). An real world example of the pressure correction is illustrated in Figure 4.12

Figures 4.13 and 4.14 show how different stroke types corrected the high pressure error with their average pressures for each stroke. These figures clearly show that the high pressure error could be corrected at average of three strokes in all tested stroke types. From the 1.0 pressure recorded in the first stroke, the average of pressure detected was reduced to the 0.7±0.1 range.

Figures 4.15 and 4.16 show the no-touch error correction results and their average

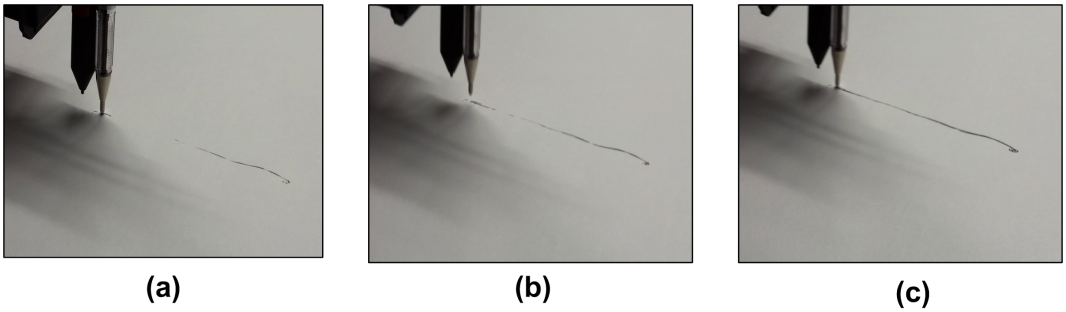
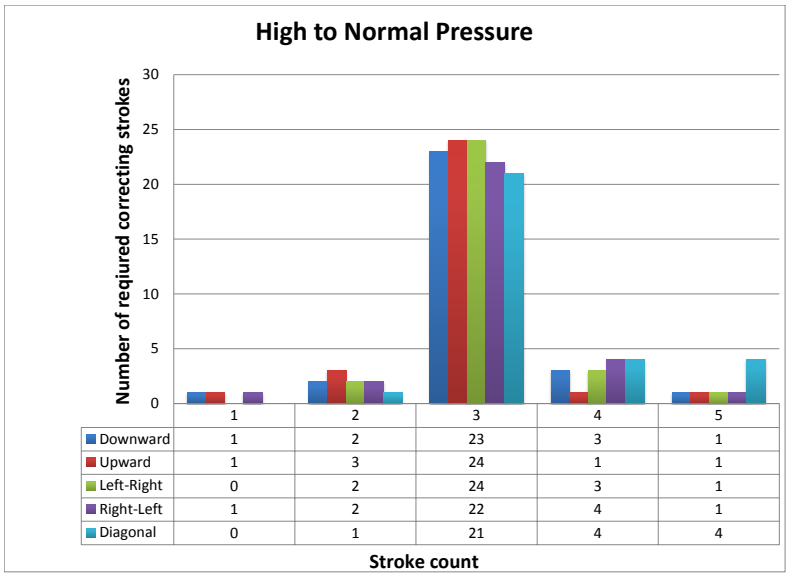


Figure 4.12: Real world example of stroke pressure correction. A line segment is completed in few trial after depth correction based on torque feedback from servos.

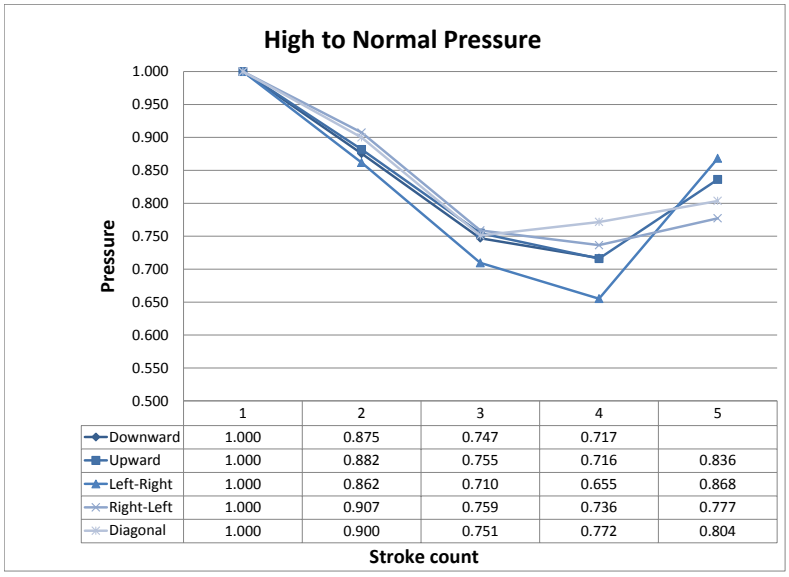
pressures for each stroke. As we have seen in the high pressure error results, the no-touch error could also be corrected at average of three strokes in those tested stroke types. From the initial no-touch stroke (no pressure is read from the tablet) the average of pressure detected increased to the 0.7 ± 0.1 range. In this experiment, I labelled the trail as a failure if correction was unsuccessful after the first stroke and the last stroke (fifth stroke). Hence, the TFC model provides 92.7% and 94.0% of successful detection of high pressure and no-touch respectively.

Figures 4.17 and 4.18 show the accumulated strokes for the corrections which compare difficulty of different stroke types to make their corrections. Figure 4.19 shows the averages of trails needed to make the correction of each stroke type in both the high pressure and the no-touch conditions. Table C.6 in appendix C shows its full experimental result.



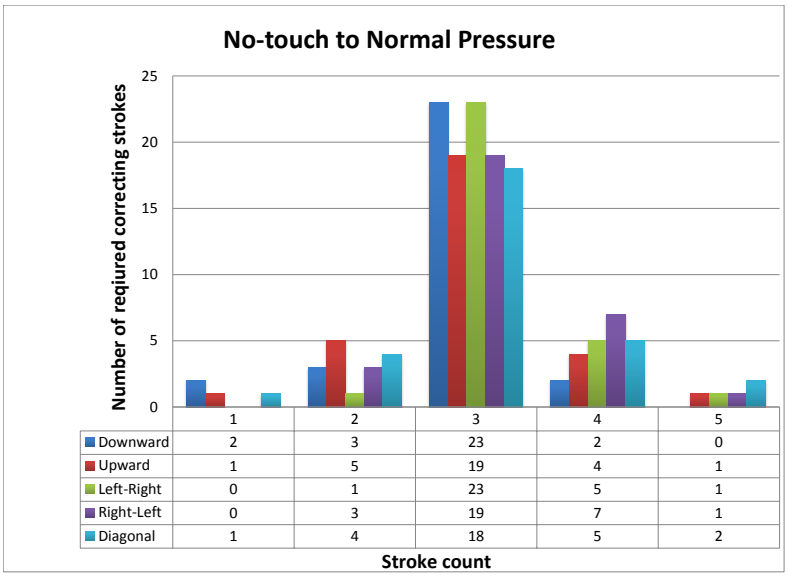
	1	2	3	4	5
Downward	1	2	23	3	1
Upward	1	3	24	1	1
Left-Right	0	2	24	3	1
Right-Left	1	2	22	4	1
Diagonal	0	1	21	4	4

Figure 4.13: Number of stroke occurrence: High to normal pressure.



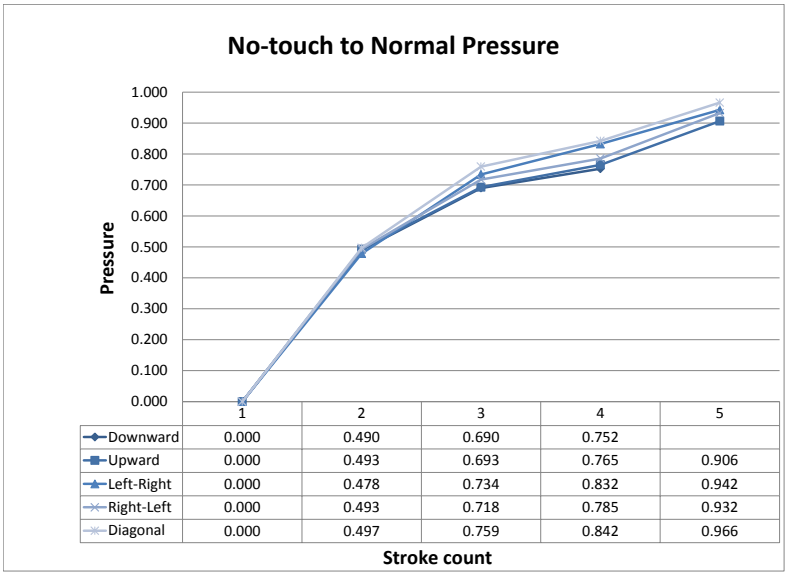
	1	2	3	4	5
Downward	1.000	0.875	0.747	0.717	
Upward	1.000	0.882	0.755	0.716	0.836
Left-Right	1.000	0.862	0.710	0.655	0.868
Right-Left	1.000	0.907	0.759	0.736	0.777
Diagonal	1.000	0.900	0.751	0.772	0.804

Figure 4.14: Average pressure for each strokes: High to normal pressure.



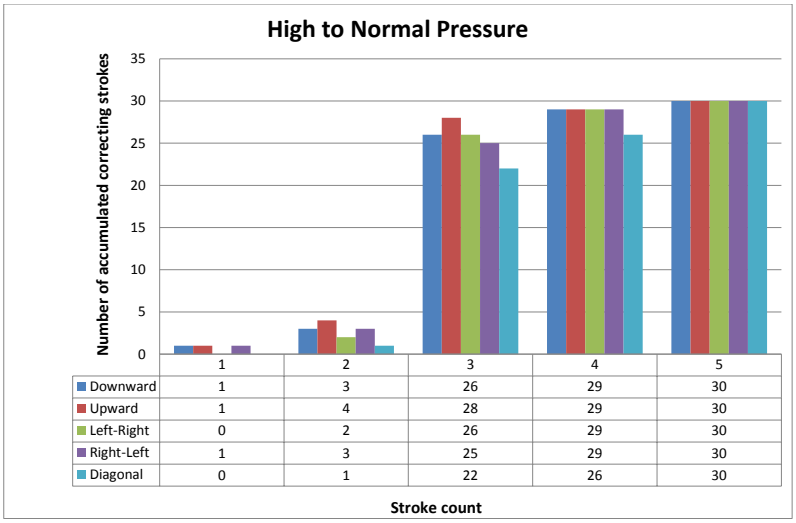
	1	2	3	4	5
Downward	2	3	23	2	0
Upward	1	5	19	4	1
Left-Right	0	1	23	5	1
Right-Left	0	3	19	7	1
Diagonal	1	4	18	5	2

Figure 4.15: Number of stroke occurrence: No-touch to normal pressure.



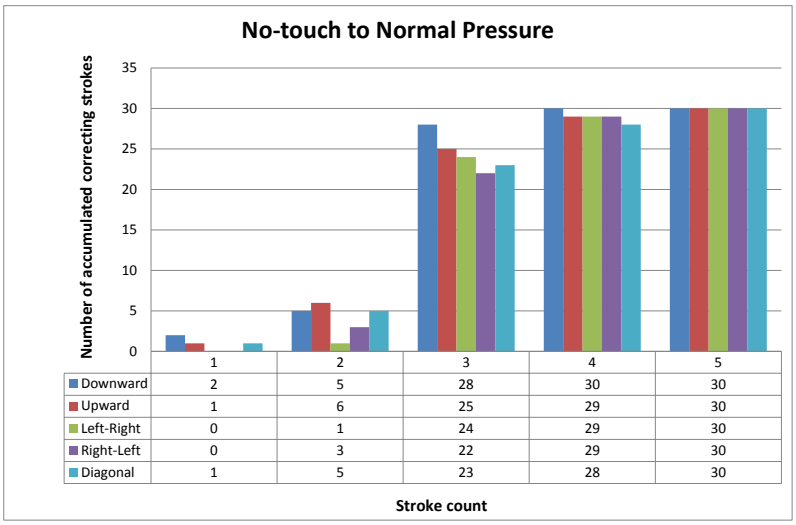
	1	2	3	4	5
Downward	0.000	0.490	0.690	0.752	
Upward	0.000	0.493	0.693	0.765	0.906
Left-Right	0.000	0.478	0.734	0.832	0.942
Right-Left	0.000	0.493	0.718	0.785	0.932
Diagonal	0.000	0.497	0.759	0.842	0.966

Figure 4.16: Average pressure for each strokes: No-touch to normal pressure.



	1	2	3	4	5
Downward	1	3	26	29	30
Upward	1	4	28	29	30
Left-Right	0	2	26	29	30
Right-Left	1	3	25	29	30
Diagonal	0	1	22	26	30

Figure 4.17: Number of accumulated stroke: High to normal pressure.



	1	2	3	4	5
Downward	2	5	28	30	30
Upward	1	6	25	29	30
Left-Right	0	1	24	29	30
Right-Left	0	3	22	29	30
Diagonal	1	5	23	28	30

Figure 4.18: Number of accumulated stroke: No-touch to normal pressure.

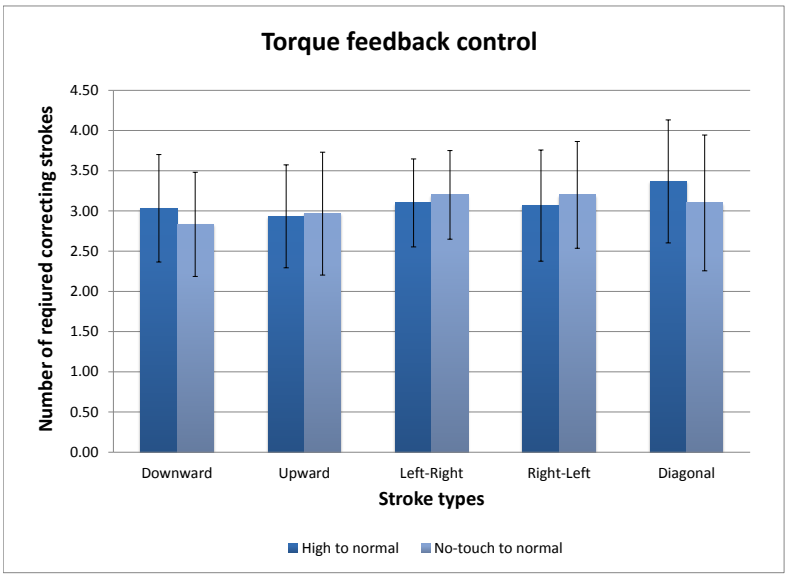


Figure 4.19: Number of strokes for different stroke type based on TFC model. Error bars indicate the standard deviation.

4.3.3 Visual Servoing Control Experiment

In this experiment, I tested the efficiency of the IBVS control approach based on two given x- and y- position errors of 10 mm and 20 mm offset (0.5 mm is equivalent to 1 Bamboo tablet's unit). Betty repeated 30 times for each stroke type of vertical, horizontal and diagonal lines based on these two offsets for a total of 90 trails in each initial condition. Only one direction of strokes (downward and left-right) were examined in this experiment because it had no effect on visual information. For these trials, the average errors in the position for each type of stroke were measured. Next, I calculated the absolute differences of the current position of two endpoints, p_1 and p_2; and the desired position, $p_1^{'}$ and $p_2^{'}$ as mapped by the Bamboo tablet. This calculation is shown in equations (4.1) and (4.2), *dist* is the Euclidean distance between two end points of p_1 and $p_1^{'}$. I selected the proportional gain, $K_p = 0.4$ empirically during the experiment.

$$e_{p1} = dist(p_1, p_1^{'}) \tag{4.1}$$

$$e_{p2} = dist(p_2, p_2^{'}) \tag{4.2}$$

Figure 4.20 shows the result if the drawing is moved during the experiment. In Figure 4.21 shows how line segments were perceived from the camera based on the algorithms discussed in previous chapters.

Figures 4.22 and 4.23 show the number of required correcting strokes to the initial 10 mm offset error and how it converged to its desired distance from a baseline (line

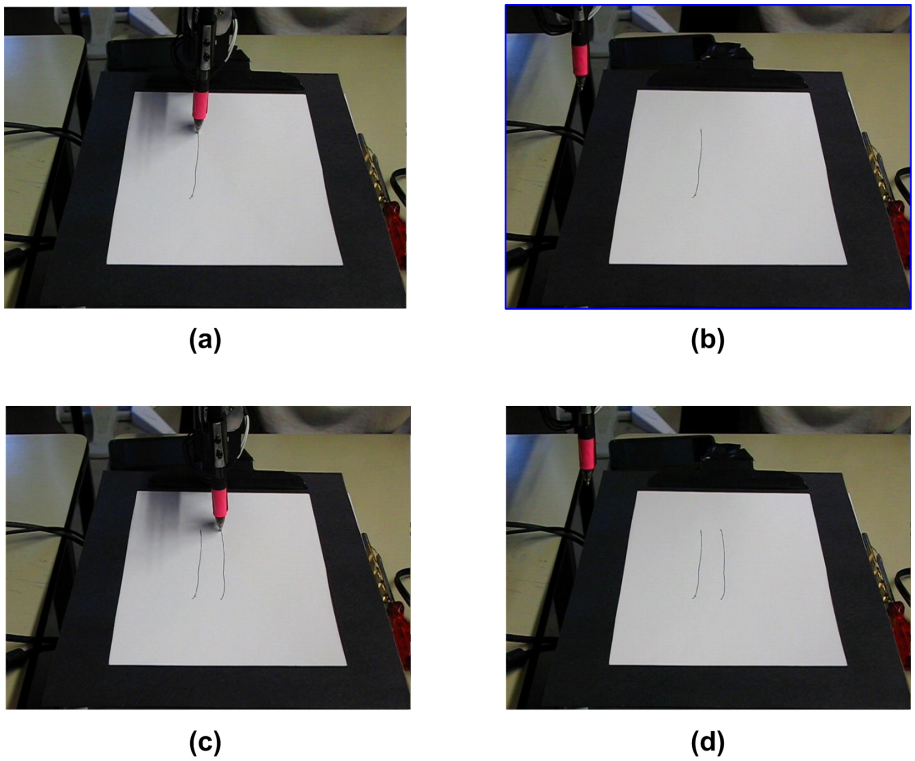


Figure 4.20: Real world example of stroke error correction with IBVS control. A correct line segment was drawn in second trial after error correction based on its line segment's feature.

segment). These figures clearly show that this 10 mm offset error was successfully corrected within two to three strokes for vertical and horizontal strokes. It reduced the error to approximately 5 mm from the desired baseline. However, diagonal strokes did not show good convergence on average. Based on my observation, it was caused by the sinusoidal drawing motion at the wrist that is likely to create curvy lines.

Therefore, I used linear interpolation in this matter but small vibration at the pen's tip caused by the non-smooth trajectory created jerking lines. This problem introduced significant noise to the existing line segment detection algorithm and affected the result as seen in Figure 4.23.

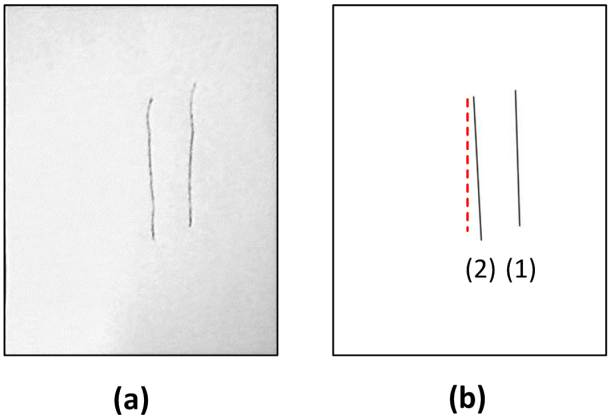


Figure 4.21: Line segments perceived from the camera. **(a)**: input ROI image as seen from camera, **(b)**: line segments detected by eEDLines approach *(1)* is the first stroke, *(2)* is the corrected stroke and the dotted line is the desired line segment.

Similarly to 10 mm offset experiments, Figures 4.24 and 4.25 show consistent convergence of vertical and horizontal strokes. However, due to the drawing motion trajectory; diagonal strokes remained harder to correct, often requiring 4-stroke and 5-stroke.

In this experiment, I labelled five-stroke trails as correction failures. IBVS control approach had relatively higher failure rate of 13% and 17% in diagonal strokes compare to lower failure rate of nearly 3% in vertical and horizontal strokes for 10 mm and 20 mm offsets respectively.

Figures 4.26 and 4.27 show the accumulated strokes for the corrections which compare difficulty of different stroke types to make their corrections. Figure 4.28 shows the averages of trails needed to make the correction of each stroke type in both 10 mm and 20 mm offset conditions. Table C.9 in appendix C shows its full experimental result.

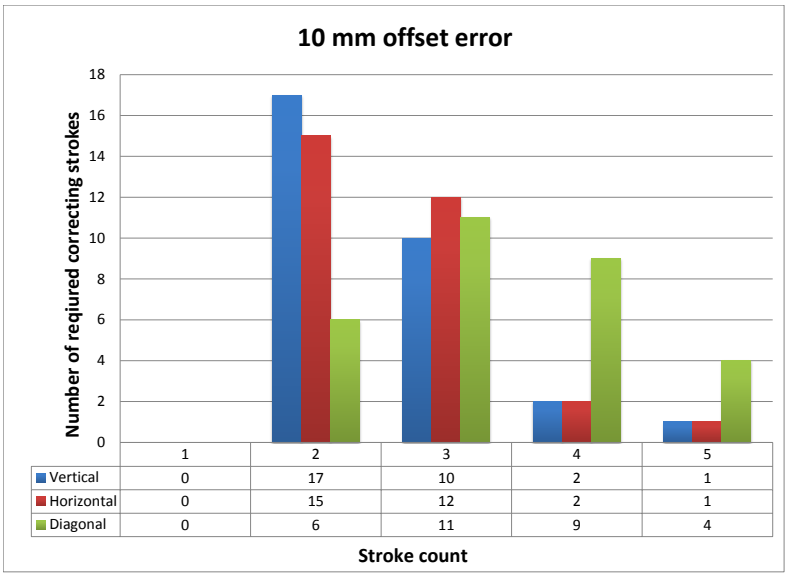


Figure 4.22: Number of stroke occurrence for each trial with initial 10 mm offset

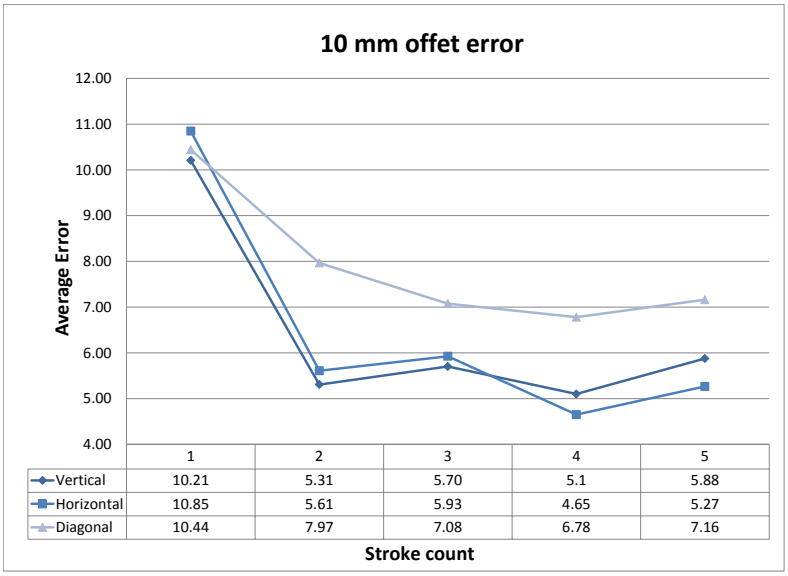


Figure 4.23: Average error of p1 and p2 for each strokes with initial 10 mm offset

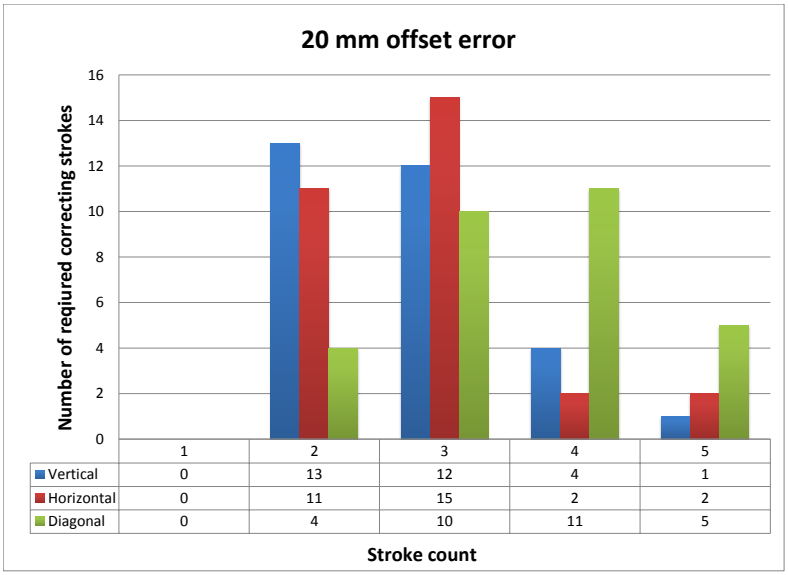


Figure 4.24: Number of stroke occurrence for each trial with initial 20 mm offset.

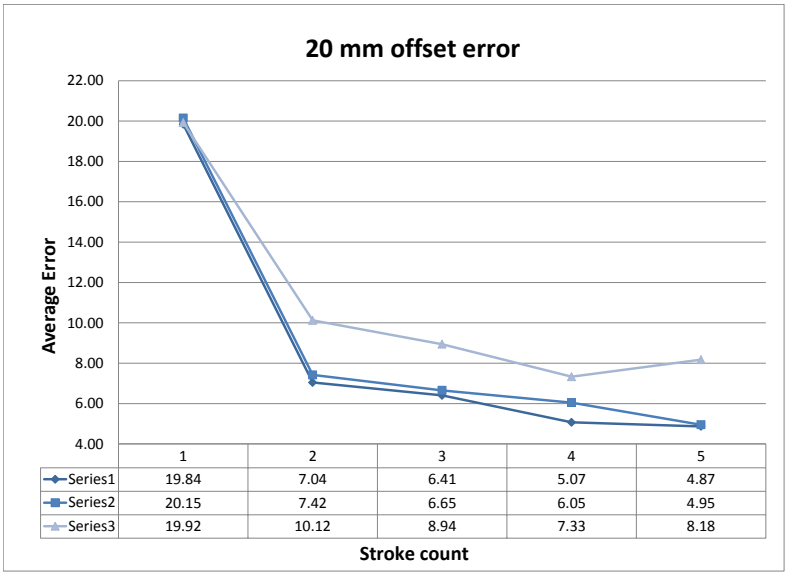


Figure 4.25: Average error of p1 and p2 for each strokes with initial 20 mm offset.

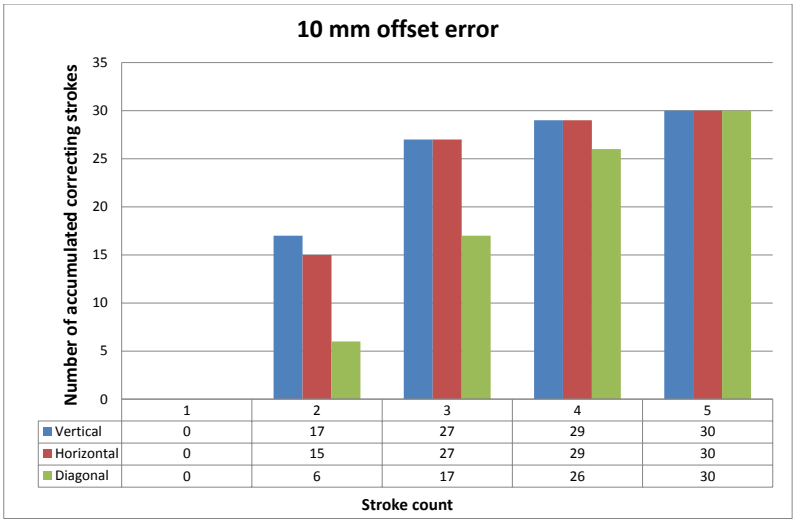


Figure 4.26: Number of accumulated stroke for each trial with initial 10 mm offset.

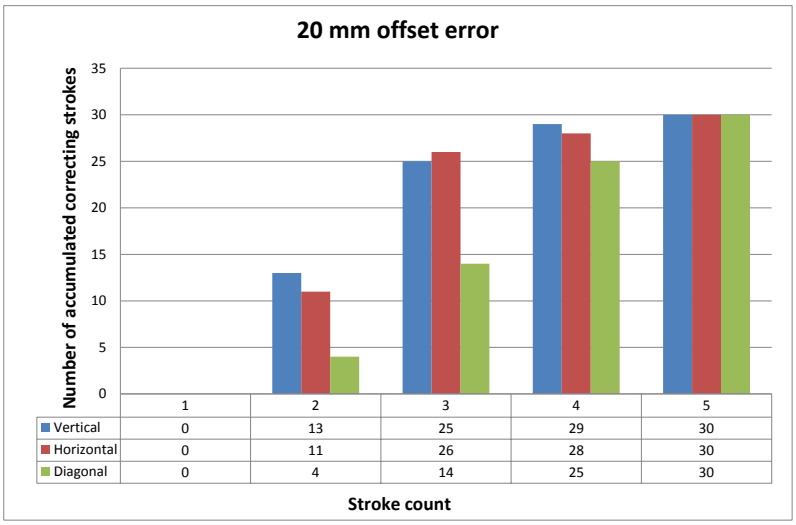


Figure 4.27: Number of accumulated stroke for each trial with initial 20 mm offset.

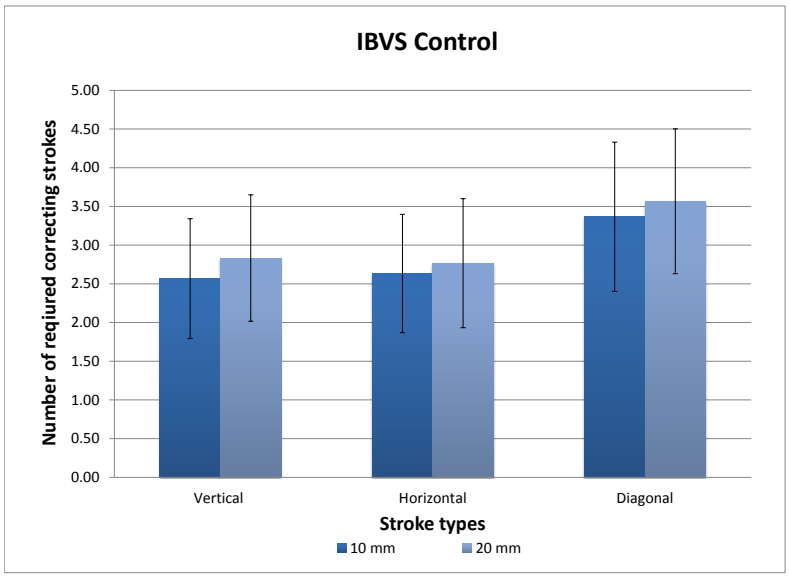


Figure 4.28: Number of strokes for different stroke type based on IBVS control. Error bars indicate the standard deviation.

4.3.4 Hybrid Control Experiment

In this experiment, TFC and IBVS approaches were combined. The pressure and position errors were measured as we have seen in two previous experiments to investigate the superiority compared to TFC and IBVS. However, due to the high number of physical experiments; only 20 mm offset error condition was used in this experimental setting. Figure 4.29 shows the example of the no-touch and 20 mm offset correction in this experiment.

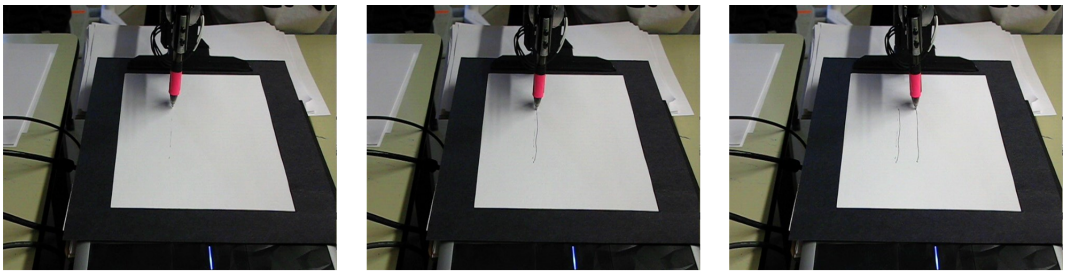


Figure 4.29: Hybrid control vertical line experiment. **(a)**: no line is drawn in first trail due to no-touch error, **(b)**: line is draw in second trial with 20 mm offset error, **(c)**: offset error is corrected in third trial

Figure 4.31 and 4.30 show the number of required correcting strokes to correct initial high pressure error in the 20 mm offset error experiment. I observed no significant difference between the hybrid approach compare to IBVS control. This indicates that the IBVS control can successfully corrected the drawing deviation independently even in high pressure condition. As we have seen previously, the increased error of diagonal stroke is likely the result of drawing motion trajectory.

For the no-touch scenario, the initial correction (second stroke) of position error was indeterminate due to the lack of visual reference for the line segment feature in

the first stroke. When a line segment was drawn on the second stroke, it corrected the position error based IBVS control. Since the initial no-touch error could be corrected by either TFC or IBVS (detects the absent of line segment), it raised an important question of the necessity of using torque feedback. However, to draw a more complex sketch, IBVS could be insufficient to detect the error due to multiple line segments from the drawing area. Therefore TFC is necessary to identify the initial no-touch error where the hybrid control is superior in this matter.

Figures 4.34 and 4.35 show the accumulated strokes for the corrections which compare difficulty of different stroke types to make their corrections. Figure 4.36 shows the averages of trails needed to make the correction of 20 mm offset under both no-touch and high pressure conditions. Table C.12 in appendix C shows its full experimental result.

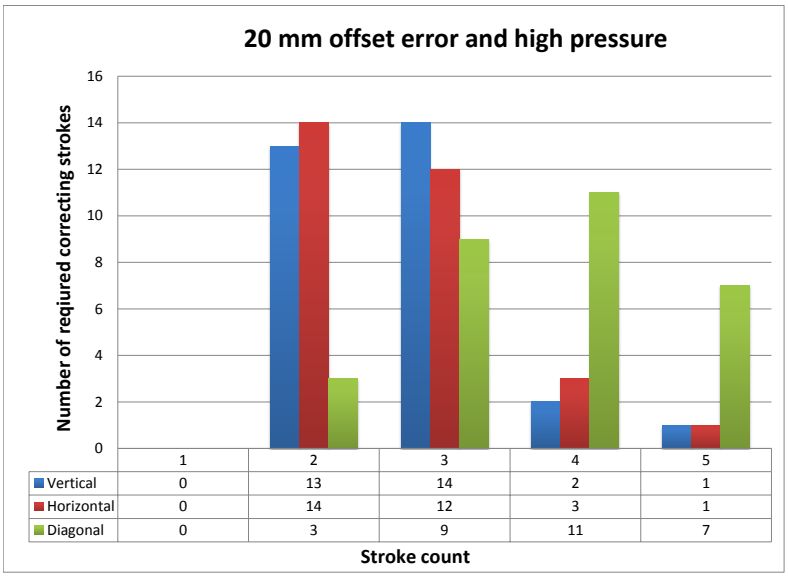


Figure 4.30: Number of stroke occurrence: High to normal pressure with 20 mm offset.

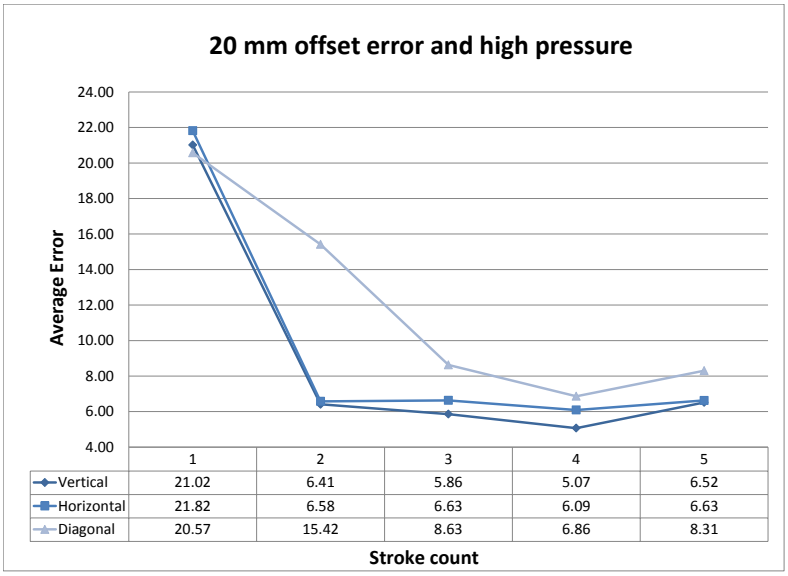


Figure 4.31: Average pressure for each strokes: High to normal pressure with 20 mm offset.

Figure 4.32: Number of stroke occurrence: No-touch to normal pressure with 20 mm offset.

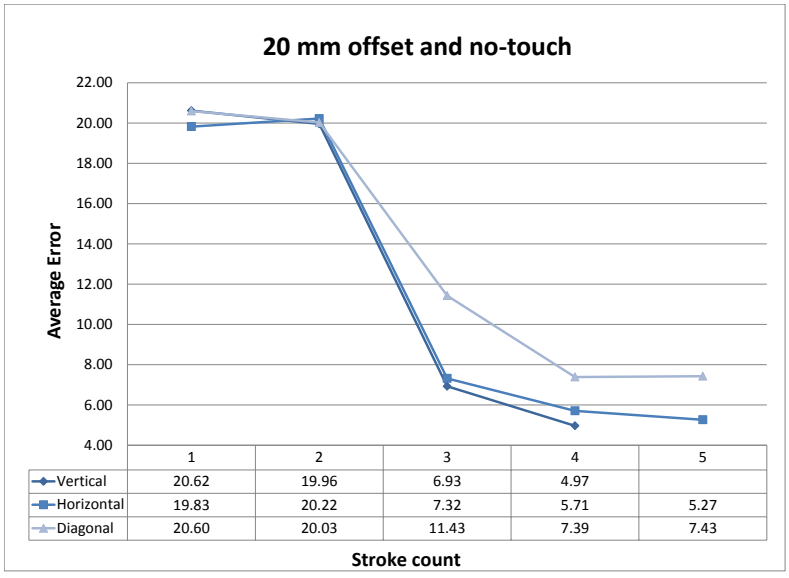


Figure 4.33: Average pressure for each strokes: No-touch to normal pressure with 20 mm offset.

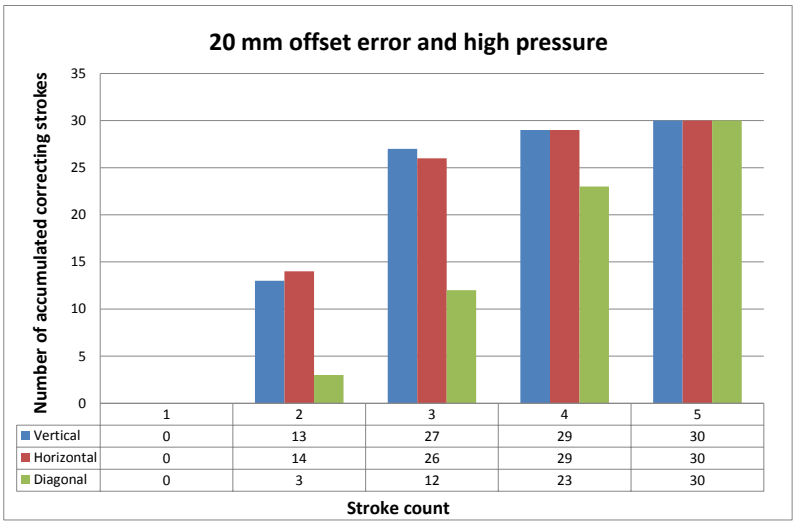


Figure 4.34: Number of accumulated stroke: High to normal pressure with 20 mm offset.

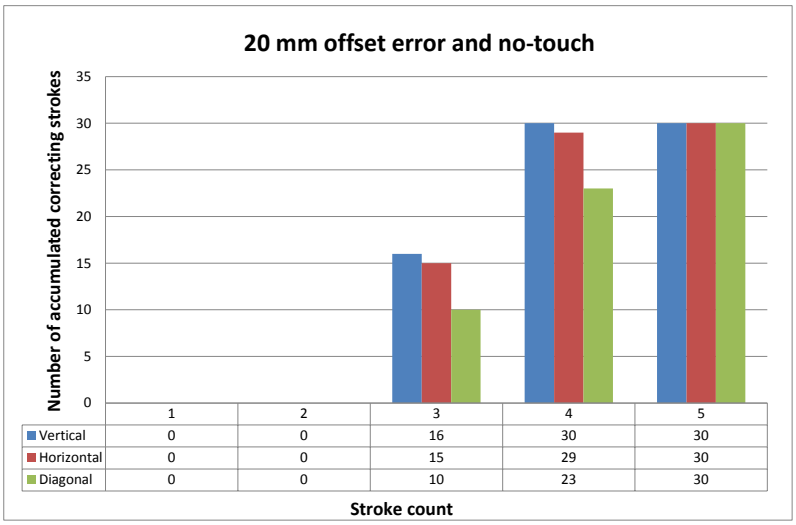


Figure 4.35: Number of accumulated stroke: High to normal pressure with 20 mm offset.

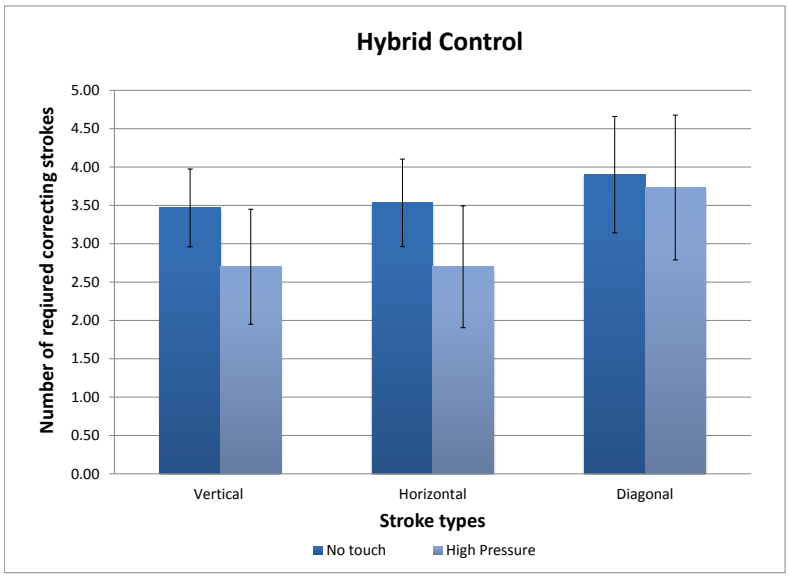


Figure 4.36: Number of strokes for different stroke type based on Hybrid model. Error bars indicate the standard deviation.

4.3.5 Drawing Accuracy Experiment

This segment compares the performance of the non-feedback and the feedback controls by matching between their baseline image (as seen in Figure 4.10). In this evaluation, only hybrid control was used as a feedback control because TFC and IBVS are not powerful enough for a complicated sketch. I evaluated the accuracy of the output by using their dark pixels and SURF descriptors. In the SURF evaluation, I used OpenCV's *SurfFeatureDetector*, *SurfDescriptorExtractor* and *BruteForceMatcher* to run the feature matching between input baseline image and drawn sketch. Figure 4.37 illustrates an example of output from the SURF evaluation.

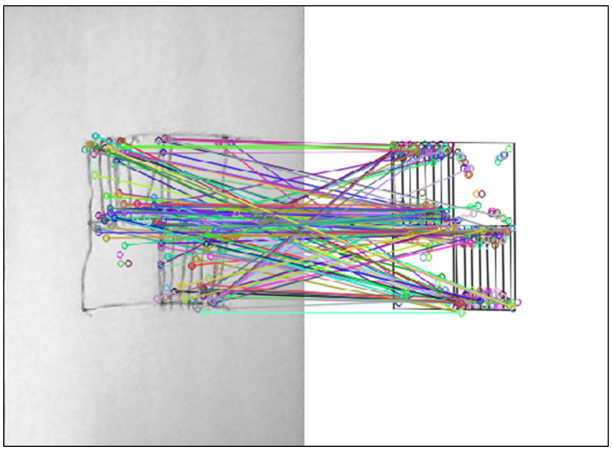


Figure 4.37: SURF features comparison between the baseline checker and the drawn output.

Figure 4.38 is the comparison of drawn checkers of different drawing conditions. Figure 4.38(b) and (c) were drawn without inverse kinematics (IK) noise; and Figure 4.38(d) and (e) were the drawn sketches with noise of 10 mm for x, y, and z-dimensions. These noises were randomly applied during the drawing process.

On average, the number of strokes were increased for hybrid control on both with

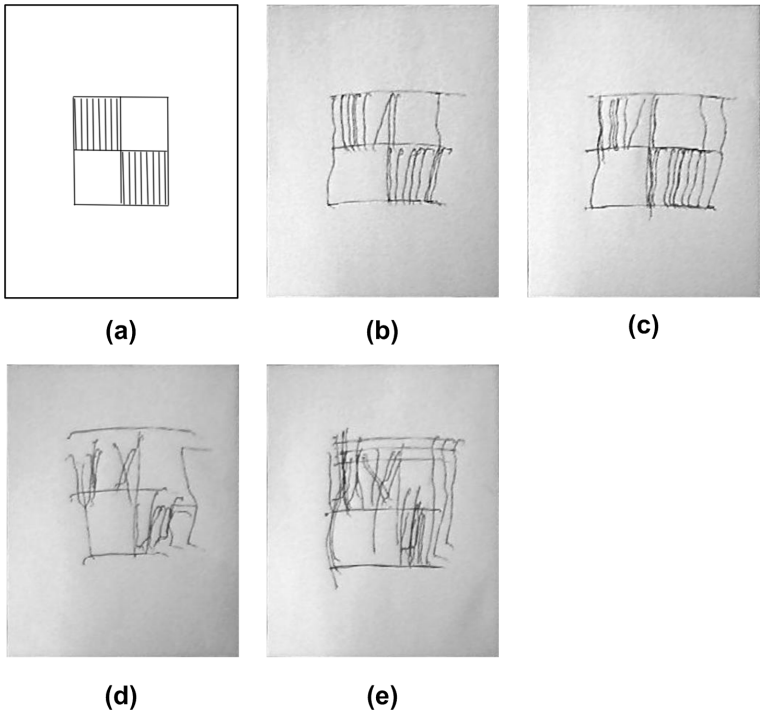


Figure 4.38: Drawn checker from different configuration. **(a)**: checker sketch **(b)**: output without feedback **(c)**: output with hybrid control **(d)**: output with IK noise and without feedback **(e)**: output with IK noise and hybrid control. I used a cropped image of chairs, due to a high number of line segments in the original image.

and without IK noise as seen in Table 4.3. Based on my observation and visual evaluation of Figure 4.38(e), the hybrid control improved its drawn output significantly against the IK noise as compared to Figure 4.38(d). Figure 4.39 illustrates other comparisons of the drawn outputs of open-loop and hybrid control.

Table 4.3: Average stroke count per line segment

	Non-feedback	Feedback	Average stroke increse
No IK noise	22	48	2.18
With IK noise	22	63	2.86

Table 4.4 shows the SURF keypoints comparison of the baseline checker (345 keypoints) and pixel-to-pixel to the drawn checker with no IK noise. The results did

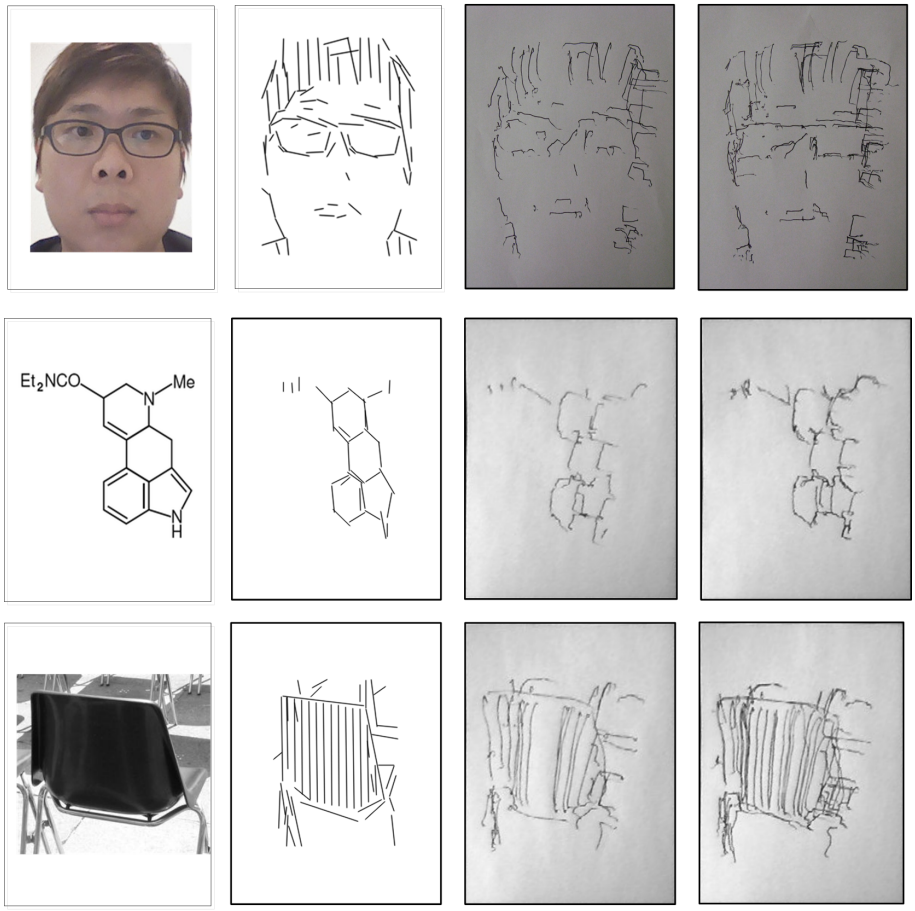


Figure 4.39: Drawn outputs comparison. From left to right: input image, generated sketch, drawn output without feedback, drawn output with hybrid control

not show the improvement of the feedback control in the sense of feature matching percentages of dark pixels due to simplicity of sketch. But it shows increase of SURF keypoints detected in feedback approach due to the greater number of drawn lines. In the other drawn outputs (portrait, molecules and chair with SURF keypoints of 633, 419 and 690 respectively), Tables 4.5, 4.6 and 4.7 clearly shows an increase of percentages for matched dark pixels and SURF keypoints when using feedback control.

Table 4.4: Feedback and non-feedback comparison without IK noise (checker).

	No IK noise (Checker)			
	Non-feedback	%	Feedback	%
SURF (354)	152/187	81.28	225/259	86.87
Pixel-to-pixel	3512/10015	35.07	3701/10015	36.95

Table 4.5: Feedback and non-feedback comparison without IK noise (portrait).

	No IK noise (Portrait)			
	Non-feedback	%	Feedback	%
SURF (633)	287/316	90.82	409/425	96.24
Pixel-to-pixel	2284/12909	17.69	3870/12909	29.98

Table 4.6: Feedback and non-feedback comparison without IK noise (molecules).

	No IK noise (Molecules)			
	Non-feedback	%	Feedback	%
SURF (419)	111/125	88.80	279/294	94.90
Pixel-to-pixel	1566/5188	30.19	1762/5188	33.96

Table 4.7: Feedback and non-feedback comparison without IK noise (chair).

	No IK noise (Chair)			
	Non-feedback	%	Feedback	%
SURF (690)	290/312	92.95	567/589	96.26
Pixel-to-pixel	4234/16984	24.93	6559/16984	38.62

Under random IK noise, I only used checker in the experiment to see how my hybrid control method performed against the noise due to the complexity of noisy output of other inputs. Table 4.8 clearly shows an increase of percentage of SURF keypoints when using feedback control. However, the pixel matching method is difficult to justify due to the error line segments created by IK noise. Therfore, the SURF method is a better estimator of drawing accuracy.

Table 4.8: Feedback and non-feedback comparison with IK noise (checker).

	With IK noise (Checker)			
	Non-feedback	%	Feedback	%
SURF (354)	159/201	79.10	330/384	85.94
Pixel-to-pixel	3019/10015	30.14	5170/10015	51.62

4.4 Summary

In this chapter, I have described my methods and hardware that I used to evaluate my implementations of drawing robot. Based on the experimental results, it shows that eEDLines successfully reduced the number of line segments compare to other tested algorithms and retained the representation of the input images. I established that TFC, IBVS and their combination in the drawing task are reliable to drawing deviation of line segments in sketch nearly 85% of the time. I also established superiority and robustness of hybrid control against drawing deviation in the drawing task. It shows that the hybrid control is nearly 5% better than non-feedback control with or without IK noise in SURF keypoint matching experiments. Therefore, it is proven that a feedback control has it advantage over a non-feedback control by improving the quality of its drawn output.

Chapter 5

Conclusion

5.1 Contribution

This thesis makes many contributions to the state of the art in modern robotics. The major contribution is the development of a humanoid robot with drawing deviation correction capability.

In this thesis work, I developed a real-time embedded control system (Betty's Control Program) to enable communication between the Motion Editor and hardware (e.g. CM2+, RX-64). This embedded system deploys a pre-emptive multi-tasking kernel to handle several tasks. The main advantage of a pre-emptive scheduling is its real-time response on the task level. Where the time to tick a task is mainly depends on the interrupt latency. Therefore, I use a PID controller to optimise the latency and jitter by controlling the context switching frequency. On the other hand Betty's embedded system uses the Absolute Position protocol and a queue data structure to improve the reliability of its data communication and makes it easier to handle.

For the drawing problem, I implemented a new approach for the drawing robot to visual servo the drawing task, using an image-based visual servoing technique. The goal of this approach is to translate the drawing deviation into control actions within the drawing task.

Furthermore, the main contribution of my work is I implement the combination of the torque feedback control with the IBVS framework which combined vision/-torque control. This framework is based on the task frame formalism in a hybrid control setting and uses both the torque feedback equipped in the Dynamixel RX-64 servo motor at each joint, and vision feedback of the current drawing image. This hybrid approach improves Betty's drawing capability in a 3D workspace, where XY and Z(depth) errors could be corrected by the drawn line segment features and the estimated pressure respectively.

As we seen in Chapter 2, there are many drawing robots. Unfortunately, the cost of the hardware of these robot systems is too expensive. In my work, I used an affordable upper body humanoid robot, which has the downside of error prone drawing. The benefit of this approach is that it provides Betty with the capability of observational drawing by way of torque and visual feedbacks which can overcome the drawbacks of the hardware limitations.

In order to reduce the total number of strokes required to compute a drawing task, I developed a unique modified furthest neighbour theta graphs approach to generate sketch lines that can reduce the time required for a drawing task. However, this approach did not reduce the number of line segments significantly. Therefore, an improved EDLines approach, namely the extended EDLines or eEDLines approach is

introduced to significantly reduce the line segments in the sketch generator.

The capability to evaluate the efficiency of different drawing implementations is very important. Unfortunately, none of the current research done in the field has provided a conclusive platform for drawing output evaluation. In my thesis, I used a Bamboo tablet and a series of experiments to provide the capability of drawing evaluation. This may lead to a better understanding of the various components, such as the correlation between pressure and drawing (pixels) deviation in different implementations.

5.2 Future Research

As future work, I would like to improve the visual feedback of the drawing task by investigating better line segment detection algorithms. Such as, a line detection algorithm proposed by Liu et al. [62] which is based on steerable filters for edge ridge detection and line fitting algorithm. Based on their experimental results, this method outperforms LSD and EDline in the evaluation of their work.

Since there is a limitation in the area of diagonal line drawing motions, another extension to my work would be developing a better interpolation motion e.g. 3rd order spline fit to smooth the IK trajectory of the drawing motion. It would potentially prevent jerking and vibrations when a line is drawn as suggested in Calinon et al. work [16].

I would also like to consider machine learning techniques to automatically select and tune the existing parameters. They would pre-set thresholds instead of manually specifying them as I did in some of my implementations, e.g. eEDline. This extension

would improve the robustness of the sketch generator to create a realistic portrait when used with my existing approach.

One last interesting future work would be to integrate other control methods with the IBVS approach, e.g. stereo vision and fuzzy control as proposed by Hanh and Lin in their 6-DOF manipulator object grasping research [39]. This would improve the accuracy and robustness of drawing deviation correction throughout the drawing process.

5.3 Conclusion

In this thesis, three approaches are implemented for drawing deviation control, torque feedback, image-based visual servoing and hybrid control. These implementations correct the errors during the drawing task when no accurate force sensor feedback at the end effector is used. This image-based visual servoing utilises an image-based structure, using image moments as a set of general features representative of an image object if possible. The control action of the image-based visual servoing has been used as a function of error in the visual space. This approach leads to a significant reduction of the computational burden as compared to existing model based approaches, as well as compared to existing learning approaches that model inverse kinematics.

Appendix A

Robot Motion Controller

Data Structure

In order to optimise the embedded system, I measured the latencies and jitters of 100 runs to analyse the efficiency of those different implementation discussed in section 3.3. Baud rate of 1 Mbps is used in USART0 for serial communication. In these experiments, 58-byte of instruction packet is sent to the Control Program; then the averages of latencies and jitters in different queue data structures were compared.

Table A.1: Results of circular and linear queue data structures

	Queue Data Structures	
	Linear	Circular
Latency (μs)	581	1103
Jitter (μs)	2.0	1.4

Based on the experimental results in Table A.1, generally both queue data structures produce acceptable latency and jitter. Circular queue needs 1103μs compare to 581μs in regular queue which perform better in this implementation where it takes

10μs to send a byte at 1 Mbps baud rate. However, circular queue provide an advantage over linear queue where it uses less memory in an embedded system for real-time transmission.

PID Controller

Then the PID controller is extended with a Optimisation Quality Function to minimize the latency. This function returns its control output in the factor of 75% latency and 25% jitter to control the context switching frequency. The optimization phase is similar to the general PID controller, except it will find the context switching time which has the lowest control output from the quality function. It will be adjusted to reach the desired set point based on the latencies and jitters measurements. Performance of the PID controller and Optimisation Quality Function are tested based on three difference protocols discussed in the previous work of motion controller implementation [58]. Figure A.1 shows the test results of PID controller with 15000μs as the set point (SP) of latency. Figure A.2 shows the optimization result of 100 trials at 5000Hz with $\pm$ 50 of context switching frequency in different protocols. The averages of latencies are 15320μs, 15105μs and 15411μs for Absolute Position, Difference and Sliding Resolution respectively.

Although the Difference protocol has shown better performance in both experiments but it is not significant and only if the differences between new and current positions are less then 28 for each servo movement as seen in A.1. In contrast, the Absolute Position protocol is simpler and easier to implement and it has most reliable performance compare to the other two tested protocols.

Due to the fluctuation of latency along the desired set point, I implement control limit approach to acknowledge the steady state after a few occurrences of control outputs fall between the control limits are perceived.

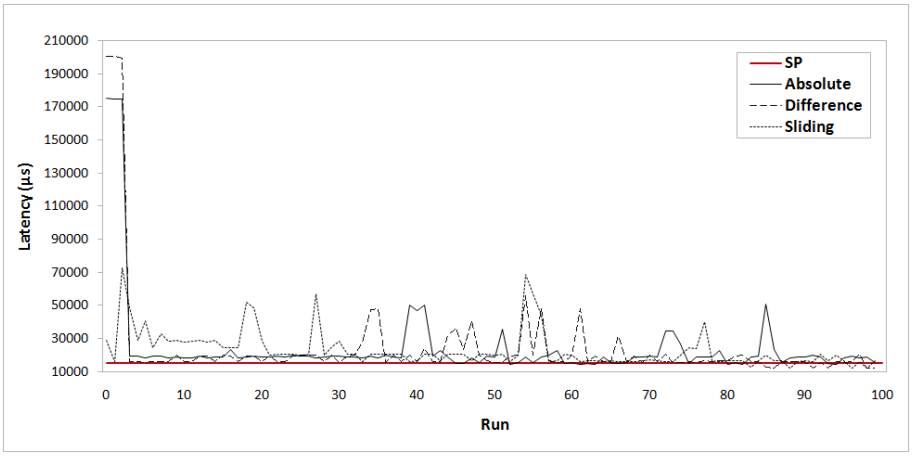


Figure A.1: Performance of PID controller in different communication protocols with $K_P = 0.1$, $K_I = 0$ and $K_D = 0.02$

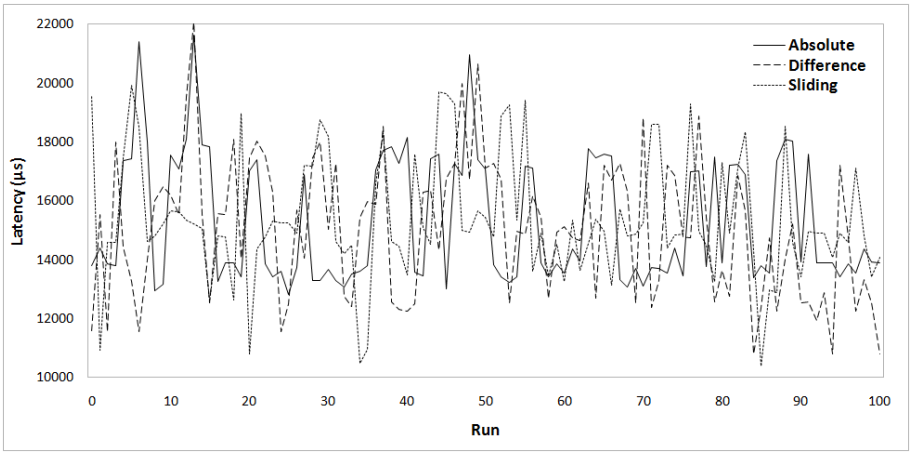


Figure A.2: Performance of Optimisation Quality Function in different communication protocol

Appendix B

Graphical User Interface

Following are some GUI screenshots of the implementation of Betty.

Motion Controller

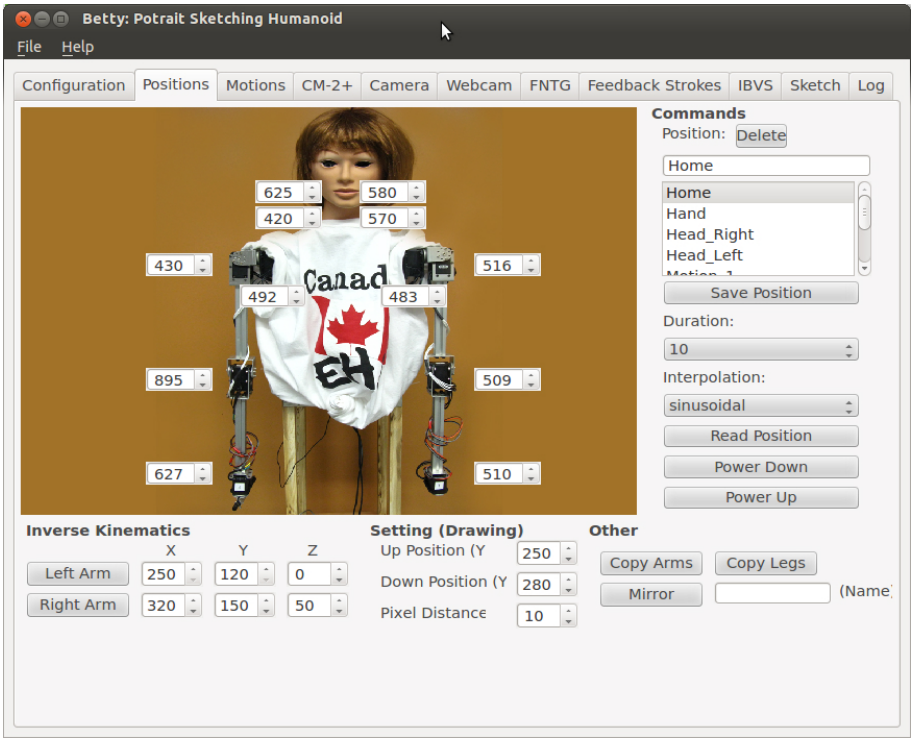


Figure B.1: Motion controller screen-shot

Furthest Neighbour Theta-graph

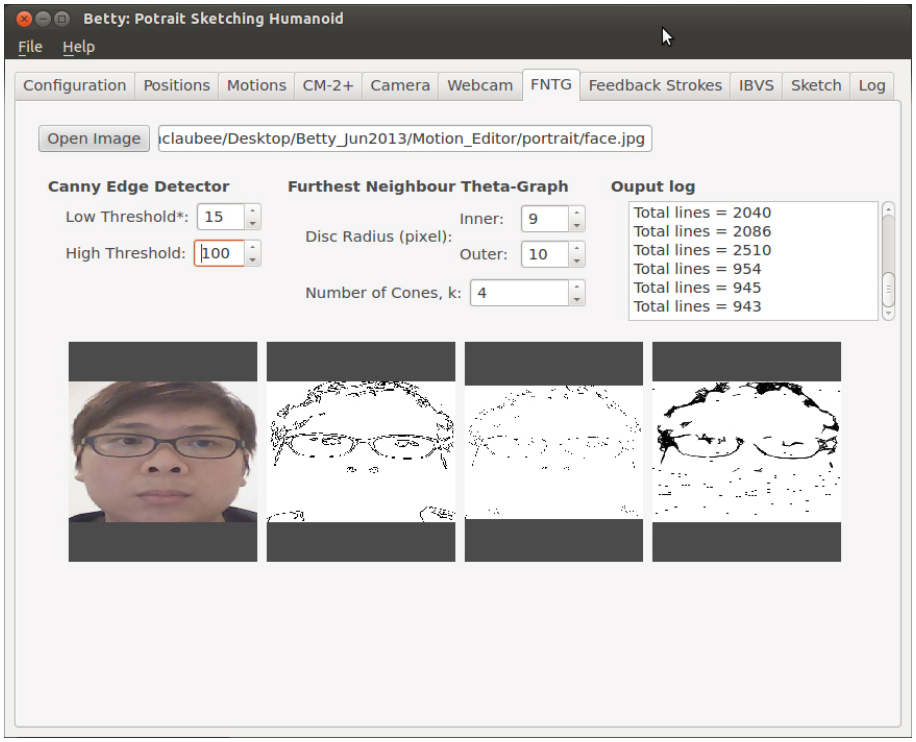


Figure B.2: Furthest Neighbour Theta-graph screen-shot.

Face Detection

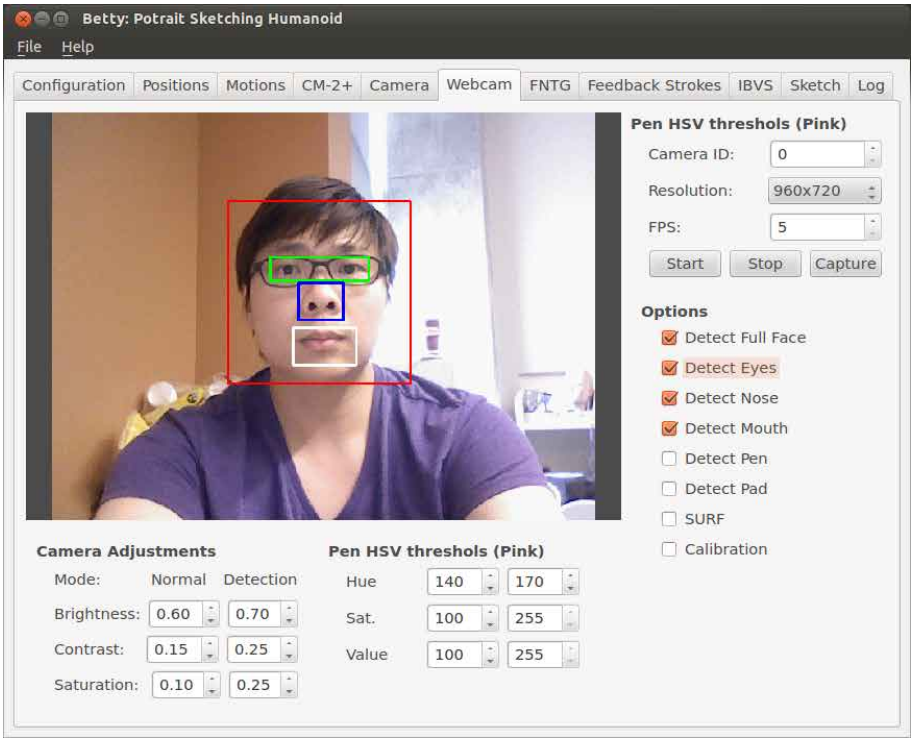


Figure B.3: Face detection.

Outlining: extended Edge Drawing Lines (eEDLines)

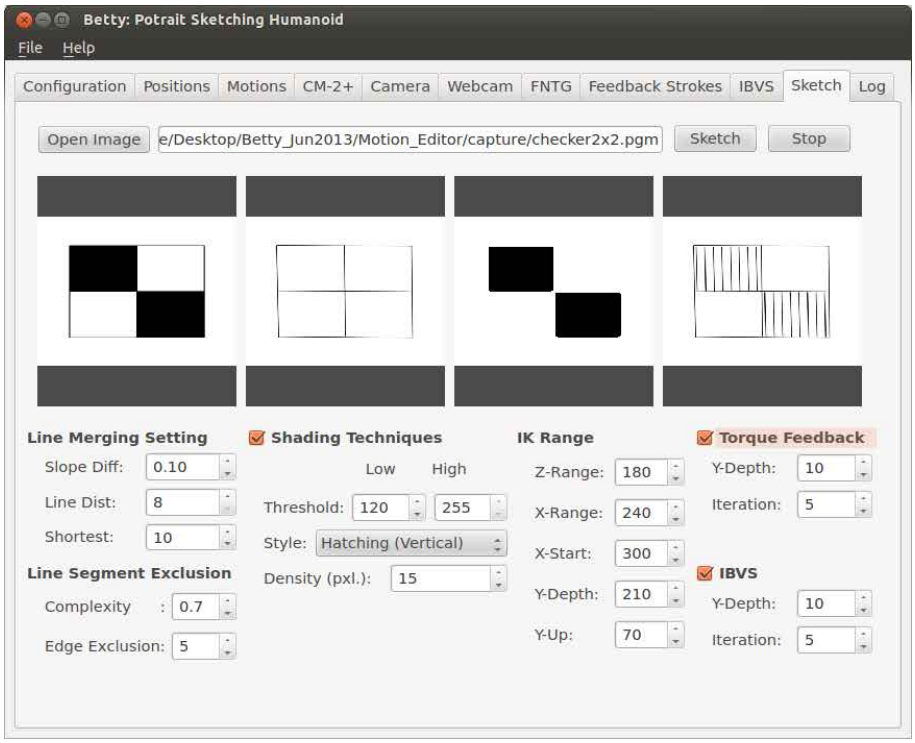


Figure B.4: Sketch generator: eEDLine.

Pad and Pen Detection

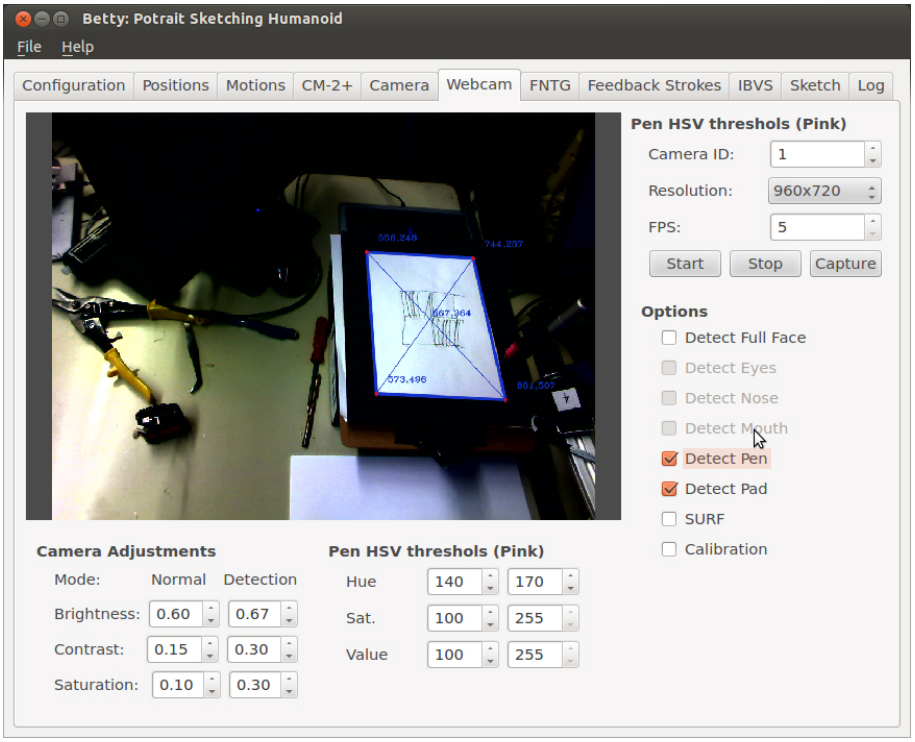


Figure B.5: Pen and pad detection.

Drawn sketch extraction

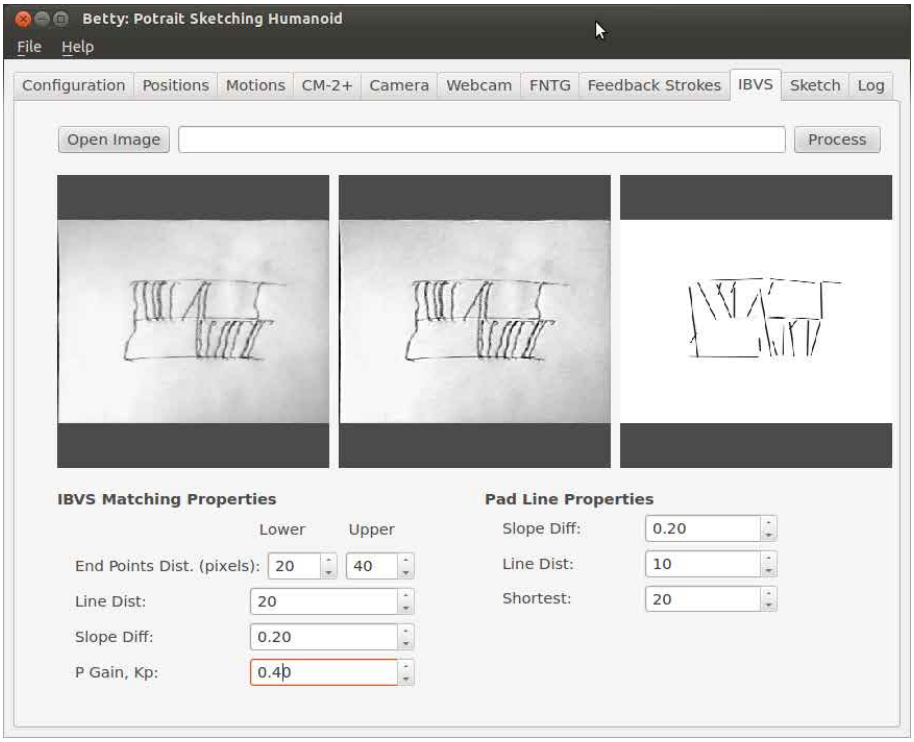


Figure B.6: Drawn sketch extraction.

Appendix C

Full Evaluation Results

Following tables are full evaluation results of the experiments as described in Chapter 4.

Extended Edge Drawing Lines

Table C.1: Number of line segments extracted based on different algorithms.

Input image	resolution	Number of Extracted Line Segments			
		PPHT	LSD	EDLines	eEDLines
Checker	349 x 383	13	14	12	6
Portrait	236 x 276	198	123	139	69
Molecules	192 x 172	19	56	64	37
Chairs	512 x 512	947	574	658	428

Table C.2: Processing time of different line segments extraction algorithms.

Input image	resolution	Processing Time (ms)			
		PPHT	LSD	EDLines	eEDLines
Checker	349 x 383	15.31	52.53	5.26	5.39
Portrait	236 x 276	28.29	73.42	5.04	7.65
Molecules	192 x 172	6.58	20.71	2.50	3.83
Chairs	512 x 512	92.48	263.41	18.37	108.53

Torque feedback control

Table C.3: Torque feedback control with high pressure

	Stroke 1			Stoke 2		
	Average	Std. Dev.	Occurrence	Average	Std. Dev.	Occurrence
Vertical Downward	1.000	0.0000	1	0.875	0.0618	2
Vertical Upward	1.000	0.0000	1	0.882	0.0399	3
Horizontal Left-Right	1.000	0.0000	0	0.862	0.0404	2
Horizontal Right-Left	1.000	0.0000	1	0.907	0.0631	2
Diagonal	1.000	0.0000	0	0.900	0.0623	1

Table C.4: Torque feedback control with high pressure (cont.)

	Stroke 3			Stroke 4		
	Average	Std. Dev.	Occurrence	Average	Std. Dev.	Occurrence
Vertical Downward	0.747	0.0484	23	0.717	0.0403	3
Vertical Upward	0.755	0.0457	24	0.716	0.0308	1
Horizontal Left-Right	0.710	0.0507	24	0.655	0.0189	3
Horizontal Right-Left	0.759	0.0591	22	0.736	0.0516	4
Diagonal	0.751	0.0500	21	0.772	0.0694	4

	Stroke 5		
	Average	Std. Dev.	Occurrence
Vertical Downward			1
Vertical Upward	0.836		1
Horizontal Left-Right	0.868		1
Horizontal Right-Left	0.777		1
Diagonal	0.804	0.0113	4

Table C.5: Torque feedback control with no-touch

	Stroke 1			Stoke 2		
	Average	Std. Dev.	Occurrence	Average	Std. Dev.	Occurrence
Vertical Downward	0.000	0.0000	2	0.490	0.0533	3
Vertical Upward	0.000	0.0000	1	0.493	0.0604	5
Horizontal Left-Right	0.000	0.0000	0	0.478	0.0433	1
Horizontal Right-Left	0.000	0.0000	0	0.493	0.0551	3
Diagonal	0.000	0.0000	1	0.497	0.0677	4

	Stroke 3			Stroke 4		
	Average	Std. Dev.	Occurrence	Average	Std. Dev.	Occurrence
Vertical Downward	0.690	0.0555	23	0.752	0.0304	2
Vertical Upward	0.693	0.0352	19	0.765	0.0528	4
Horizontal Left-Right	0.734	0.0495	23	0.832	0.0682	5
Horizontal Right-Left	0.718	0.0549	19	0.785	0.0570	7
Diagonal	0.759	0.0511	18	0.842	0.0836	5

	Stroke 5		
	Average	Std. Dev.	Occurrence
Vertical Downward			0
Vertical Upward	0.906		1
Horizontal Left-Right	0.942		1
Horizontal Right-Left	0.932		1
Diagonal	0.966	0.0479	2

Table C.6: Number of strokes for different stroke type based on TFC model.

	High to normal		No-touch to normal	
	Stroke count	Std Dev.	Stroke count	Std Dev.
Vertical Downward	3.10	0.8030	2.97	0.8503
Vertical Upward	3.00	0.7878	3.00	0.8710
Horizontal Left-Right	3.30	0.6513	3.43	0.6261
Horizontal Right-Left	3.13	0.7303	3.33	0.7112
Diagonal	3.50	0.7768	3.00	1.1447

IBVS control

Table C.7: IBVS control with 10 mm offset error

	Stroke 1				Stoke 2			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	10.64	9.77	10.21	0	5.23	5.38	5.31	17
Horizontal	10.69	11.01	10.85	0	5.56	5.65	5.61	15
Diagonal	10.39	10.50	10.44	0	7.98	7.95	7.97	6

	Stroke 3				Stroke 4			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	5.61	5.79	5.70	10	5.74	4.46	5.1	2
Horizontal	5.84	6.01	5.93	12	4.69	4.61	4.65	2
Diagonal	7.04	7.12	7.08	11	6.73	6.82	6.779615	9

	Stroke 5			
	eP1	eP2	Average	Occurrence
Vertical	6.10	5.65	5.88	1
Horizontal	4.67	5.86	5.27	1
Diagonal	6.80	7.52	7.16	4

Table C.8: IBVS control with 20 mm offset error

	Stroke 1				Stoke 2			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	19.94	19.74	19.84	0	8.86	10.28	9.57	13
Horizontal	19.92	20.38	20.15	0	10.40	9.78	10.09	11
Diagonal	20.15	19.69	19.92	0	16.61	16.34	16.48	4

	Stroke 3				Stroke 4			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	6.68	6.25	6.46	12	5.28	4.85	5.07	4
Horizontal	6.67	6.64	6.65	15	6.23	5.86	6.05	2
Diagonal	8.93	8.95	8.94	10	7.31	7.35	7.33	11

	Stroke 5			
	eP1	eP2	Average	Occurrence
Vertical	5.15	4.59	4.87	1
Horizontal	5.24	4.67	4.95	2
Diagonal	8.31	8.04	8.18	5

Table C.9: Number of strokes for different stroke type based on IBVS control.

	Pixel off set			
	10 mm		20 mm	
	stroke count	Std Dev.	stroke count	Std Dev.
Vertical	2.57	0.7739	2.83	0.8172
Horizontal	2.63	0.7649	2.77	0.8339
Diagonal	3.37	0.9643	3.57	0.9353

Hybrid control

Table C.10: Hybrid control with 20 mm offset error and no-touch

	Stroke 1				Stoke 2			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	21.33	19.91	20.62	0	20.15	19.77	19.96	0
Horizontal	19.95	19.70	19.83	0	20.28	20.17	20.22	0
Diagonal	20.44	20.76	20.60	0	20.04	20.01	20.03	0

	Stroke 3				Stroke 4			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	7.06	6.80	6.93	16	5.08	4.86	4.97	14
Horizontal	7.03	7.61	7.32	15	5.68	5.75	5.71	14
Diagonal	11.47	11.39	11.43	10	6.97	7.80	7.39	13

	Stroke 5			
	eP1	eP2	Average	Occurrence
Vertical				0
Horizontal	4.67	5.86	5.27	1
Diagonal	7.28	7.58	7.43	7

Table C.11: Hybrid control with 20 mm offset error and high pressure

	Stroke 1				Stoke 2			
	eP1	eP2	Average	Occurrence	eP1	eP2	Average	Occurrence
Vertical	20.00	22.04	21.02	0	6.57	6.26	6.41	13
Horizontal	21.42	22.21	21.82	0	6.44	6.72	6.58	14
Diagonal	19.62	21.52	20.57	0	15.62	15.21	15.42	3
Vertical	5.92	5.80	5.86	14	4.88	5.26	5.07	2
Horizontal	6.51	6.76	6.63	12	5.91	5.75	5.83	3
Diagonal	8.63	8.63	8.63	9	6.93	6.80	6.86	11
	Stroke 5							
	eP1	eP2	Average	Occurrence				
Vertical	6.51	6.52	6.52	1				
Horizontal	6.76	6.50	6.63	1				
Diagonal	8.12	8.49	8.31	7				

Table C.12: Number of strokes for different stroke type based on hybrid control with 20 mm offset error.

	20mm off set			
	No touch		High Pressure	
	stroke count	Std Dev.	stroke count	Std Dev.
Vertical	3.17	0.3790	2.93	0.8193
Horizontal	3.27	0.5208	2.87	0.8277
Diagonal	3.90	0.7589	3.73	0.9444

Bibliography

[1] Aghili, F., Buehler, M., and Hollerbach, J. M. Dynamics and control of direct-drive robots with positive joint torque feedback. In *IN PROC. IEEE INT. CONF. ROBOTICS AND AUTOMATION* (1997), pp. 2865–2870.

[2] Akinlar, C., and Topal, C. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters 32*, 13 (2011), 1633 – 1642.

[3] Akinlar, C., and Topal, C. Edlines: Real-time line segment detection by edge drawing (ed). In *18th IEEE International Conference on Image Processing (ICIP)* (2011), pp. 2837–2840.

[4] Anderson, D. *Model Based Inference in the Life Sciences: A Primer on Evidence*. Springer Science + Business Media, 2008.

[5] Baeten, J., de Schutter, J., and Schutter, J. *Integrated Visual Servoing and Force Control: The Task Frame Approach*. Engineering online library. Springer, 2004.

[6] Ballard, D. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition 13*, 2 (1981), 111 – 122.

[7] Baltes, J., Cheng, C. T., Lau, M. C., and Anderson, J. E. Cost oriented automation approach to upper body humanoid robot. In *Proceedings of the 18th IFAC World Congress* (Milano, Italy, September 2011).

[8] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (1975), 509–517.

[9] Bentley, J. L. Multidimensional divide-and-conque. *Communications of the ACM 17* (1980), 703–724.

[10] Bonichon, N., Gavoille, C., Hanusse, N., and Ilcinkas, D. Connections between theta-graphs, delaunay triangulations, and orthogonal surfaces. In *Graph Theoretic Concepts in Computer Science*, D. Thilikos, Ed., vol. 6410 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 266–278.

[11] Bose, P., Gudmundsson, J., and Morin, P. Ordered theta graphs. *Comput. Geom. Theory Appl. 28* (May 2004), 11–18.

[12] Bourquardez, O., Mahony, R., Hamel, T., and Chaumette, F. Stability and performance of image based visual servo control using first order spherical image moments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (oct. 2006), pp. 4304 –4309.

[13] Bradsk, G., and Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*, 3rd. ed. O'Reilly Media, 2008.

[14] Brown, P., Bigge, B., Bird, J., and P. Husbands, M. Perris, D. S. The drawbots. `http://www.sussex.ac.uk/Users/philh/pubs/drawbots-muta-final-small.pdf`, 2005. Accessed: 2011-07-25.

[15] Burns, J. B., Hanson, A., and Riseman, E. Extracting straight lines. *Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-8*, 4 (1986), 425–455.

[16] Calinon, S., Epiney, J., and Billard, A. A humanoid robot drawing human portraits. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (HUMANOID 2005)* (Tsukuba, Japan, December 2005), IEEE-RAS.

[17] Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8* (November 1986), 679–698.

[18] Chaumette, F., and Hutchinson, S. Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine 13*, 4 (Dec. 2006), 82–90.

[19] Chaumette, F., and Hutchinson, S. Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine 14*, 1 (Mar. 2007), 109–118.

[20] Cheah, C.-C., Hou, S. P., Zhao, Y., and Slotine, J.-J. Adaptive vision and force tracking control for robots with constraint uncertainty. *Mechatronics, IEEE/ASME Transactions on 15*, 3 (2010), 389–399.

[21] Chu, H. K., Mills, J. K., and Cleghorn, W. L. Image-based visual servoing through micropart reflection for the microassembly process. *Journal of Micromechanics and Microengineering 21*, 6 (2011), 065016.

[22] Clarkson, K. Approximation algorithms for shortest path motion planning. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1987), STOC '87, ACM, pp. 56–65.

[23] Clever, M. Makelangelo 2. `http://www.marginallyclever.com/blog/drawbot`, 2013. [Online: accessed 31-July-2013].

[24] Cohen, H. The further exploits of aaron, paiter. `http://crca.ucsd.edu/~hcohen/cohenpdf/furtherexploits.pdf`, 1994. Accessed: 2011-06-13.

[25] De Luca, A., Oriolo, G., and Giordano, P. R. On-line estimation of feature depth for image-based visual servoing schemes. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April (2007), 2823–2828.

[26] Denavit, J., and Hartenberg, R. S. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanisms* (1955), 215–221.

[27] Desolneux, A., Moisan, L., and Morel, J. *From Gestalt Theory to Image Analysis: A Probabilistic Approach*. Interdisciplinary Applied Mathematics. Springer, 2008.

[28] Desolneux, A., Moisan, L., and Morel, J.-M. Meaningful alignments. *Int. J. Comput. Vision 40*, 1 (Oct. 2000), 7–23.

[29] Duda, R. O., and Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM 15*, 1 (Jan. 1972), 11–15.

[30] Eberly, D. H. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*, 2 ed. CRC Press, 2006, ch. Distance Algorithms.

[31] Emami, S. Dynamixel servo motors. `http://www.shervinemami.info/dynamixels.html`, 2009. Accessed: 2011-11-20.

[32] Florczyk, S. *Robot Vision: Video-based Indoor Exploration with Autonomous and Mobile Robots*. WILEY-VCH Verlag GmbH & Co. KGaA, 2005.

[33] Franklin, G. F., Powell, D. J., and Emami-Naeini, A. *Feedback Control of Dynamic Systems*, 4th ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[34] Futureshop. Wacom bamboo create tablet (cth670m). `http://www.futureshop.ca/en-CA/product/wacom-wacom-bamboo-create-tablet-cth670m-cth670m/10180531.aspx`, 2013. Accessed: 2013-10-20.

[35] Galamhos, C., Matas, J., and Kittler, J. Progressive probabilistic hough transform for line detection. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* (1999), vol. 1, pp. –560 Vol. 1.

[36] Gommel, M., Haitz, M., and Zappe, J. autoportrait project: Portrait drawings with a robotic arm. `http://www.robotlab.de/auto/portrait.htm`, 2004. Accessed: 2011-05-20.

[37] Grompone Von Gioi, R., Jakubowicz, J., Morel, J.-M., and Randall, G. On straight line segment detection. *J. Math. Imaging Vis. 32*, 3 (Nov. 2008), 313–347.

[38] Guptill, A. L. *Drawing with Pen and Ink: And a Word Concerning the Brush*, 4 ed. Reinhold Publishing Corporation, New York, NY, 1946.

[39] Hanh, L. D., and Lin, C.-Y. Combining stereo vision and fuzzy image based visual servoing for autonomous object grasping using a 6-dof manipulator. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on* (2012), pp. 1703–1708.

[40] Hartenberg, R. S., and Denavit, J. *Kinematic Synthesis of Linkages.* McGraw-HIll, New York, NY, 1964.

[41] Hashimoto, M. Robot motion control based on joint torque sensing. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on* (may 1989), pp. 256 –261 vol.1.

[42] Hill, J., and Park, W. Real time control of a robot with a mobile camera. In *in Proceedings of 9th ISIR* (Washington, D.C., Mar 1979), pp. 409–417.

[43] Hodges, S. E., and Hall, T. Looking for a cheaper robot: Visual feedback for automated pcb manufacture, 1996.

[44] Hooper, R. Learn about robots: Robot forward kinematics. `http://www.learnaboutrobots.com/forwardKinematics.htm`, 2010. Accessed: 2010-02-25.

[45] Hosoda, K., Igarashi, K., and Asada, M. Adaptive hybrid visual servoing/force control in unknown environment. In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on* (1996), vol. 3, pp. 1097–1103 vol.3.

[46] Hough, P. V. C. Machine Analysis of Bubble Chamber Pictures. In *International Conference on High Energy Accelerators and Instrumentation* (CERN, 1959).

[47] Hutchinson, S., Hager, G., and Corke, P. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation 12*, 5 (1996), 651–670.

[48] Instruments, N. Control laws. `http://www.ni.com/white-paper/8156/en/`, 2011. [Online: accessed 04-April-2012].

[49] Jeff, F. Data structures and algorithms, part 2. `http://www.javaworld.com/javaworld/jw-06-2003/jw-0613-java101.html?page=7`, 2010. Accessed: 2009-11-15.

[50] Joachim, G., Giri, N., and Michiel, S. Geometric spanners. *Encyclopedia of Algorithms* (2008), 360–364.

[51] Keil, J. M. Approximating the complete euclidean graph. In *No. 318 on SWAT 88: 1st Scandinavian workshop on algorithm theory* (London, UK, 1988), Springer-Verlag, pp. 208–213.

[52] Kennedy, D., and Osuga, R. Calligraphy robot uses a motion copy system to reproduce detailed brushwork. `http://www.diginfo.tv/v/12-0181-r-en.php`, 2012. [Online: accessed 31-July-2013].

[53] Kosuge, K., Takeuchi, H., and Furuta, K. Motion control of a robot arm using joint torque sensors. *Robotics and Automation, IEEE Transactions on 6*, 2 (apr 1990), 258 –263.

[54] Kou, Z., Shi, Z., and Liu, L. Airport detection based on line segment detector. In *Computer Vision in Remote Sensing (CVRS), 2012 International Conference on* (2012), pp. 72–77.

[55] Kragic, D., and Christensen, H. Survey on Visual Servoing for Manipulation. Tech. rep., Centre for Autonomous Systems, Numerical Analysis and Computer Science, 2002.

[56] Kudoh, S., Ogawara, K., Ruchanurucks, M., and Ikeuchi, K. Painting robot with multi-fingered hands and stereo vision. *Robotics and Autonomous Systems 57* (2009), 279–288.

[57] Kurt, M., and Stefan, N. Dynamic fractional cascading. *ALGORITHMICA 5*, 1 (1990), 215–241.

[58] Lau, M. C., and Baltes, J. The real-time embedded system for a humanoid: Betty. In *Proceedings of the 13th FIRA Robot World Congress* (Bangalore, India, September 2010), vol. 103 of *Communications in Computer and Information Science*, Springer-Verlag Berlin Heidelberg, pp. 122–129.

[59] Leite, A. C., Lizarralde, F., and Hsu, L. Hybrid adaptive vision-force control for robot manipulators interacting with unknown surfaces. *Int. J. Rob. Res. 28*, 7 (July 2009), 911–926.

[60] Li, J., An, X., and He, H. Line segments detection with scale analysis: A principal component analysis based approach. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on* (2009), vol. 4, pp. 43–47.

[61] Lin, C. Y., Chuang, L. W., and Mac, T. T. Human portrait generation system for robot arm drawing. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (Singapore, July 2009), IEEE, pp. 1757–1762.

[62] Liu, Y., Mejias, L., and Li, Z. Fast power line detection and localization using steerable filter for active uav guidance. In *ISPRS12*.

[63] Lu, Y., Lam, J. H. M., and Yam, Y. Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator. In *Proceedings of the*

IEEE/ASME International Conference on Advanced Intelligent Mechatronics (Singapore, July 2009), IEEE, pp. 578–583.

[64] Luh, J., Fisher, W., and Paul, R. Joint torque control by a direct feedback for industrial robots. *Automatic Control, IEEE Transactions on 28*, 2 (feb 1983), 153 – 161.

[65] Luijten, H. Basics of color based computer vision implemented in matlab.

[66] Maginnis, C. D. *Pen drawing: an illustrated treatise*, 7 ed. Bates & Guild company, Boston, 1924.

[67] Malis, E., Chaumette, F., and Boudet, S. Positioning a coarse-calibrated camera with respect to an unknown object by 2d 1/2 visual servoing. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on* (May 1998), vol. 2, pp. 1352 –1359 vol.2.

[68] Malis, E., Chaumette, F., and Boudet, S. 2 frac12;d visual servoing. *Robotics and Automation, IEEE Transactions on 15*, 2 (April 1999), 238 –250.

[69] Martin, F. G. *Robotic Explorations: A Hands-on Introduction to Engineering*, 1st ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[70] Matas, J., Galambos, C., and Kittler, J. Progressive probabilistic hough transform, 1998.

[71] McArdle, L. *Portrait Drawing: A Practical Guide for Today's Artist.* Prentice-Hall, Inc., New Jersey, 1984.

[72] Nieto, M., Cuevas, C., Salgado, L., and Garca, N. Line segment detection using weighted mean shift procedures on a 2d slice sampling strategy. *Pattern Analysis and Applications 14*, 2 (2011), 149–163.

[73] Nomura, H., and Naito, T. Integrated visual servoing system to grasp industrial parts moving on conveyer by controlling 6dof arm. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on* (2000), vol. 3, pp. 1768 –1775 vol.3.

[74] Noor A. Ibraheem, Mokhtar M. Hasan, R. Z. K., and Mishra, P. K. Understanding color models: A review. 265–275.

[75] Olsson, T., Bengtsson, J., Johansson, R., and Malm, H. Force control and visual servoing using planar surface identification. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation* (Washington, DC, May 2002), IEEE, pp. 4211–4216.

[76] Pfeffer, L., Khatib, O., and Hake, J. Joint torque sensory feedback in the control of a puma manipulator. *Robotics and Automation, IEEE Transactions on 5*, 4 (aug 1989), 418 –425.

[77] R. Fisher, S. Perkins, A. W., and Wolfart., E. Hipr – the hypermedia image processing reference. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/welcome.htm`, 2003. [Online: accessed 26-March-2013].

[78] Rao, A. A. *Data Structures And Algorithms Using C++*. Pearson Education India, 2010.

[79] Robotis. Robotis: User's manual dynamixel ax-12, 2006.

[80] Robotis. Robotis: User's manual dynamixel rx-64, 2007.

[81] Ruchanurucks, M., Kudoh, S., Ogawara, K., Shiratori, T., and Ikeuchi, K. Humanoid robot painter: Visual perception and high-level planning. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation* (Roma, Italy, April 2007), IEEE, pp. 3028–3033.

[82] Siradjuddin, I., Behera, L., McGinnity, T., and Coleman, S. Image based visual servoing of a 7 dof robot manipulator using a distributed fuzzy proportional controller. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on* (july 2010), pp. 1 –8.

[83] Spong, M. W., and Vidyasagar, M. *Robot Dynamics And Control*. John Wiley Sons, Inc., New York, 2006.

[84] Srikaew, A., Cambron, M. E., Northrup, S., Peters, R. A., II, Ii, R. A. P., Wilkes, D. M., and Kawamura, K. Humanoid drawing robot. In *In Proceedings of the IASTED International Conference on Robotics and Manufacturing* (1998).

[85] Sunday, D. Distance between 3d lines and segments. `http://geomalgorithms.com/a07-_distance.html`, 2012. Accessed: 2012-08-20.

[86] Team, O. D. Geometric image transformations. `http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html#`, 2013. [Online: accessed 18-July-2013].

[87] Team, O. D. Structural analysis and shape descriptors. `http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html`, 2013. [Online: accessed 20-June-2013].

[88] Teller, S. Computational geometric and graphics codes. `http://people.csail.mit.edu/seth/geomlib/`, 2013. Accessed: 2013-02-04.

[89] Tian, L., and Goldenberg, A. Robust adaptive control of flexible joint robots with joint torque feedback. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on* (may 1995), vol. 1, pp. 1229 –1234 vol.1.

[90] Tira-Thompson, E. Digital servo calibration and modeling. Tech. Rep. CMU-RI-TR-09-41, Robotics Institute, Pittsburgh, PA, March 2009.

[91] Tresset, P., and Leymarie, F. F. Generative portrait sketching. In *Proceedings of the 11th Int. Conf. on Virtual Sytems and MultiMedia (VSMM'05)* (Ghent, Belgium, October 2005), VSMM, pp. 739–748.

[92] Tresset, P., and Leymarie, F. F. Aikon: the artistic/automatic ikonograph. In *ACM SIGGRAPH 2006 Research posters* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[93] Tresset P., F. L. F., and N., K. Skediomata: guinea pig and performer. In *Proceedings of the 17th International Symposium on Electronic Art* (Istanbul, Turkey, Septmeber 2011), ISEA Press.

[94] Tsunashima, N., and Katsura, S. Reproduction of human motion using motion-copying system based on coordinate modification. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society* (2010), pp. 1609–1614.

[95] Vahrenkamp, N., Boge, C., Welke, K., Asfour, T., Walter, J., and Dillmann, R. Visual servoing for dual arm motions on a humanoid robot. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on* (2009), pp. 208–214.

[96] Vahrenkamp, N., Wieland, S., Azad, P., Gonzalez, D., Asfour, T., and Dillmann, R. Visual servoing for humanoid grasping and manipulation tasks. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on* (2008), pp. 406–412.

[97] von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 4 (2010), 722–732.

[98] von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. LSD: a Line Segment Detector. *Image Processing On Line 2012* (2012).

[99] Wang, H.-R., Dong, H., Xiao, J.-Z., and Ma, L. Design and implementation of a visual servo system. In *Machine Learning and Cybernetics, 2009 International Conference on* (july 2009), vol. 4, pp. 2404 –2408.

[100] Weiss, L., and Sanderson, A. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation 3*, 5 (1987), 404–417.

[101] Weiss, L., Sanderson, A., and Neuman, C. Dynamic visual servo control of robots: An adaptive image-based approach. In *Proceedings of IEEE International Conference on Robotics and Automation* (March 1985), vol. 2, pp. 662 – 668.

[102] Wilson, W., Williams Hulls, C., and Bell, G. Relative end-effector control using cartesian position based visual servoing. *Robotics and Automation, IEEE Transactions on 12*, 5 (oct 1996), 684 –696.

[103] Wolf, W. *Computers and Components: Principles of Embedded Computing System Design*. The Morgan Kaufmann Series in Computer Architecture and Design Series. Morgan Kaufmann Publishers, 2001.

[104] Yao, A. C.-C. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing 11*, 4 (1982), 721–736.

[105] Zhang, G., and Furusho, J. Control of robot arms using joint torque sensors. *Control Systems, IEEE 18*, 1 (feb 1998), 48 –55.