Autonomous Vision-based Control for Within-hand Manipulation using Variable Friction Fingers

by

Joshua Amrith Raj

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Robotics Engineering Department

August 2020

APPROVED:

______________________________
Professor Berk Calli, Thesis Advisor

______________________________
Professor Michael Genner

______________________________
Professor Zhi Li
Abstract

Robotic within-hand manipulation is challenging to implement, even for highly articulated, sensorized and expensive robotic hands, mainly due to the lack of accurate hand-object and contact models. In this thesis, automatic manipulation planning and control schemes are investigated for a robotic hand with variable friction finger surfaces. Three strategies, 1) an offline motion planner that is utilized in a feedforward manner, 2) an online vision-based control algorithm, 3) a hybrid approach that combines the offline and online algorithms, are proposed to re-position and reorient an object within the workspace of the hand. These methods are designed considering the non-holonomic and switching nature of the system. The experimental results show that each method provides different advantages in terms of efficiency, accuracy and path trajectory smoothness. A method to track the object without the presence of any marker is also presented.
Acknowledgements

I would like to express my gratitude to my advisor, Prof. Berk Calli for his guidance throughout the course of this work. It has been an immense learning experience not only in robotics but about doing research. His encouragement always kept me motivated. I enjoyed working under him and never felt any pressure. I would also like to thank Adam Spiers from Max Planck Institute for Intelligent Systems for his valuable inputs during the course of this project.

I would also like to thank Gokul Narayanan, Abhinav Gandhi and Aditya Gupte who worked with me in this project. Without them this work would not have been completed. The discussions with them gave me a lot of ideas and helped understand more about the field.

I am extremely grateful for all my friends who helped me and supported me throughout my Masters. Most importantly, I would like to thank my family. I owe all the success I have had to them. I will never forget their sacrifices which has brought me this far.
Contents

1 Introduction 1

2 Related Work 6
  2.1 Within-Hand Manipulation ................................. 6
  2.2 Within-Hand Manipulation approaches .................... 7
  2.3 Control and Planning for Within-hand manipulation ..... 10
  2.4 Object Tracking .......................................... 12

3 System Description and Model 14
  3.1 Description of the VF Gripper .............................. 14
  3.2 Kinematics Model .......................................... 16
    3.2.1 Within-hand sliding ................................... 17
    3.2.2 Within-hand rotation ................................. 21

4 Planning and Control 24
  4.1 Motion Planning .......................................... 24
    4.1.1 State Space, Action Space and Cost Functions ....... 25
    4.1.2 A* Search Algorithm ................................. 26
    4.1.3 Modified Heuristic Cost Function ...................... 27
    4.1.4 Weighted A* Search ................................. 28
4.2 Visual Servoing for WIHM ........................................... 30
4.3 Hybrid Approach ......................................................... 32

5 Object Tracking ......................................................... 34
5.1 Object tracking using ArUco Markers ................................. 34
5.2 Object tracking using Mean shift ..................................... 35

6 Experiments and Results ............................................... 38
6.1 Experiments with Objects of different shapes ....................... 38
6.2 Experiments with Modelling inaccuracies ............................ 41
6.3 Experiments without the use of ArUco markers ...................... 45

7 Conclusion ............................................................... 47
7.1 Future work ............................................................. 47
List of Figures

1.1 Within-Hand Manipulation of a cubic object from an arbitrary start pose (position and orientation) to a desired end pose using an automatically determined sequence of sliding and rolling actions based on a weighted A* search algorithm with modified cost function. The 2-DOF hand platform utilizes variable friction (VF) finger surfaces. 3

3.1 Friction finger with high and low friction surfaces [1] 15

3.2 Figures 1-3 show a case of the object sliding on the right hand where the left finger is set to high friction (HF) and the right finger is set to low friction (LF). Figures 4-6 shows the object rotating within the hand. Here both the fingers are set to high friction 16

3.3 Parameters of the kinematic model for the hand-object system 17

3.4 The vectors $\vec{p}_1, \vec{p}_2, \vec{w}_o$ and $\vec{w}_p$ form a closed loop while sliding occurs. 19

3.5 Within hand anti-clockwise rotation of a square 22

3.6 Within hand rotation of a square by $\pi/2$. The square is kept rotating until the side comes in contact with the finger i.e. $\phi$ becomes zero 22
4.1 Planned path using motion planner from start state (X, Y, α = 7.0, 7.5, -90) to goal state (12, 12, 0) (a) Conventional A* search (not smooth). (b) A* search with modified cost function (smoother) (d) Weighed A* search (not smooth but fast solution). (d) Weighed A* search with modified cost function (smooth and fast solution). Axis units are cm. .......................................................... 29

4.2 The course of action in the hybrid approach ........................................... 33

6.1 The experimental setup used in this work. ................................. 39

6.2 Objects used in the experiments ................................................ 39

6.3 Goal positions for the object to reach (Refer Table 6.3 ................. 40

6.4 Trajectory tracked by the object from start pose (7,7,0) to goal pose (12,12,0) for different methods a) Visual servoing b) Offline motion planning c) Hybrid method ................................. 43

6.5 Visual servoing path while using mean shift algorithm to track the object ......................................................... 45

6.6 Comparison of mean shift algorithm with ArUco markers ............. 46
List of Tables

4.1 State Action Transition Table, for Moving the Object from State $s$
to State $s'$ ............................................................................... 25

6.1 Goal positions for each tasks ...................................................... 40
6.2 Performance of each method done with objects of different shapes . 42
6.3 Summary of Experiments with Square Prism focusing on Modelling
inaccuracies .................................................................................. 44
6.4 Summary of Experiments using mean shift algorithm to track the object 46
Chapter 1

Introduction

Within-hand manipulation refers to changing the configuration of a given object within the workspace of the hand. Humans heavily rely on their Within-Hand Manipulation (WIHM) capabilities for performing daily activities[2]. The subconsciously conducted tasks of re-orienting and re-positioning objects within the hand help people to streamline manipulation operations. This can be done by changing the initial grasp pose into task-oriented hand-object configurations without the need for regrasping (e.g. re-positioning a pen, scissors or keys after picking-up to make the objects ready for use).

Current robotics technology lacks similar dexterous manipulation capabilities, which are essential for the service robots that are expected to efficiently operate in daily human environment. WIHM capabilities can further extend the use of industrial robots to less structured environments by relaxing the constraints of their initial grasps and allowing them to re-position grasped objects within the gripper, even in confined spaces that restrict gross arm motions.

The work in this thesis was supported in part by the National Science Foundation under grant IIS-1900953.
Successful WIHM warrants controlled translation, reorientation and fine positioning of an object within the hand workspace. These actions are greatly challenging to achieve even with sensorized, high-DOF, expensive robotic hands operating in structured settings with detailed hand-object models [3, 4, 5, 6, 7].

One of the main challenges in performing these tasks is the mechanical complexity involved in the design of the hand to accomplish them. Complex tasks require dexterous hands with multiple actuators and transmissions. With highly articulated hands, it becomes difficult to model the system with accuracy. Another challenge is the control aspect of it. When the design becomes complex, we will require highly precise sensors to know information about the points of contact, position of the object, and the force outputs.

This research focuses on enabling robotic WIHM by facilitating essential WIHM actions of sliding and rotating of objects with practical and effective combinations of mechanical design, planning and control strategies, and minimal system model requirements. Spiers et al. [1] proposed a simple 2-finger, 2-DOF robot gripper with variable friction (VF) surfaces Fig. 1.1. The actuated contact surfaces of each finger (which consist of interlaced high and low friction materials) allows us to easily switch between sliding and pivoting motions of the object. Controlled sliding and rotation is performed by leveraging the friction properties of the surface. The gripper can achieve large sliding and reorientation actions despite its minimal articulation. This dexterity allows operations such as rotating objects by 180 degrees within-hand, and sliding-gating the object from the center of the grasp to the tip of the fingers.

Spiers et al. [1] demonstrated the capabilities of the gripper design with manually determined, pre-programmed actions and also presented position-level hand-object kinematics models for manipulating rectangular prisms. In this thesis, we expand on these abilities with the following additions:
Figure 1.1: Within-Hand Manipulation of a cubic object from an arbitrary start pose (position and orientation) to a desired end pose using an automatically determined sequence of sliding and rolling actions based on a weighted A* search algorithm with modified cost function. The 2-DOF hand platform utilizes variable friction (VF) finger surfaces.
1. A generalized position and velocity-level kinematics model of the hand-object system that can work with any arbitrary shaped object,

2. Three novel approaches for conducting automatic WIHM:

   (a) A feedforward control method based on a WIHM planner,
   
   (b) A feedback control method using a visual servoing scheme,
   
   (c) A hybrid method that combines the feedforward and feedback methods,

3. Experimental results using four objects of different shapes and sizes,

The algorithms conduct WIHM to move an object from an initial grasp pose to a given target pose. The VF mechanism allows us to design simple controllers, which can operate with (often inaccurate) quasi-static models and achieve accurate and efficient WIHM. In the lack of such a mechanism in the finger design, manipulation actions would require accurate kinematics and dynamics models and complicated controllers, which would try to operate within narrow stability margins (especially for sliding and gross rotation motions that we focus on). On the other hand, while the design simplifies the mechanics of WIHM, it is a non-holonomic, switching system with uneven action outcomes, all of which need to be considered for developing an automatic planning and control scheme.

The experimental results show various advantages for each of the three control methods described above: The feedforward controller results in fast executions, but require offline planning time and is sensitive to modelling inaccuracies. The visual servoing controller does not require offline planning, can achieve accurate results, but can get stuck in local minima. The hybrid method provides both fast and accurate execution, while still needing offline planning. We believe that algorithms and analysis presented in this thesis will also be valid for recently developed variable-
friction mechanisms that were inspired by design of the variable friction finger [8, 9],
or similar systems such as [10].

The following are the clarifications about the assumption and system properties

1. Only planar WIHM (object position and orientation in the grasp plane) are considered. While the VF mechanism can enable various 6D manipulation strategies, these out of the scope of this thesis.

2. The system can operate with or without a support plane (i.e. when the object is lifted).

3. The geometric model of the object is provided. This model does not need to be accurate since the feedback methods can compensate for modelling imperfections.

4. It is assumed that the target object has at least one flat surface to allow within-hand sliding.
Chapter 2

Related Work

2.1 Within-Hand Manipulation

Within-hand manipulation is to change the configuration of the object with respect to the hand, without regrasping [11, 12]. The configuration of the object with respect to the hand is also known as grasp pose. While moving the object from one grasp pose to another, the grasp stability must be ensured at the initial and final grasp configuration [11]. There are a number of approaches available to change the grasp configuration, like using anthropomorphic hands [13, 14, 15], in-grasp manipulation [16, 17], finger gaiting [18, 19, 20, 21, 22, 17, 23], finger pivoting [24], rolling [25, 26] and sliding [5, 6, 7, 27] etc. Some of these approaches are discussed in more detail in Section 2.2.

Although, within-hand manipulation offers advantages like, accurate and efficient manipulation of objects, it poses some challenges as well. It is desirable that the hands be as dexterous as possible mechanically. However, this leads to more complexity in the design, planning and control of the fingers [28]. Secondly, in order to know the accurate position of the object with respect to the hand, the sensors
have to be of high resolution and robust to noise [29]. One way of achieving this is via tactile sensors, but are often expensive and hard to integrate. In the present work, we use visual feedback in order to know the position of the object. In Section 2.2, we discuss a few approaches to perform within-hand manipulation, their advantages and disadvantages.

2.2 Within-Hand Manipulation approaches

As mentioned in the previous section, there are a number of approaches to within-hand manipulation. One of the early attempts to perform within-hand manipulation was using the anthropomorphic hands. Anthropomorphic hands are robot hands which are designed like a human hand having multiple fingers and joints. The NASA Robonaut hand [13] has five fingers along with a wrist. This hand, in total, this hand has fourteen degrees of freedom. The DLR’s articulated hand [14] is a four fingered hand with twelve degrees of freedom. This hand also contains sensors which helps us to know the position of the object and the force being exerted. The Utah hand [15] has twenty five degrees of freedom consisting of tactile sensors. J. K. Salisbury et al.[30] presented a three fingered Stanford-JPL hand.

Although anthropomorphic hands are highly dexterous, they pose a number of disadvantages [29]. Anthropomorphic hands have a complex mechanical design and require large number of actuators and transmissions, making the hands heavy. Increasing the dexterity also makes modelling of the system difficult. Similar hands such as those in [31, 32, 33] are based on detailed modelling of the hand-object system and the friction forces between the object and robotic hand. Increased complexity in the hand-object model results in difficulties in developing planning and control schemes. Another problem one encounters is to maintain a stable grasp
of the object during manipulation. Ensuring a stable grasp also results in limiting the motion of the fingers and hence the movement of the object is also constrained.

Finger gaiting is another technique which is used for within-hand manipulation. This involves hands with at least three fingers. In this approach, some of the fingers grasp the object while the remaining are free. A stable grasp on the object is maintained by the grasp fingers while the free fingers break contact and then establishes contacts at other points on the object. L. Han et al. [22], proposed a method to perform within-hand manipulation using rolling and finger gaiting. This hand contains three fingers with each finger having six degrees of freedom. B. Sundaralingam et al.[23] presented a method that involves alternating between finger gaiting and in-grasp manipulation. In-grasp manipulation is when the object is manipulated to new location without changing the point of contact between the fingers and the object. This work involves a hand with four fingers. Unlike anthropomorphic hands, finger gaiting ensures grasp stability. However, finger gaiting methods still involve a certain degree of mechanical complexity and requires accurate coordinated control of the fingers.

Dafle et al. [34] presented a way to perform within-hand manipulation by using extrinsic dexterity, where they rely on the resources external to the hand to manipulate the object. These resources could be gravity, dynamic arm motions or pushing the object against a surface. Some of the advantages extrinsic dexterity offers is avoiding the complexity of dexterous hands. The motion of the fingers are often not complex, therefore the hands can be of a simple design and cheaper. Some of the tasks they are able to achieve are sliding the object using contact, rotating the object with respect to the hand by using the fingers or by rolling it on the ground. Later some of the within-hand manipulation tasks, namely straight sliding, pivoting and rolling by pushing the grasped object were performed by pushing the grasped
object against a contact surface[35]. These approaches utilizes environmental constraints for in-hand manipulation using an extrinsic dexterity-based method. The variable friction finger and the planning and control algorithms do not rely on the use of environmental constraints, and boost intrinsic dexterity of the system.

Shi et al. [6] proposed a way to perform controlled sliding by using contact with the environment and inertial forces. Vina B. et al. [5] followed a similar approach by using gravitational force and controlled slip. They were also able to achieve rotation of the object within-the hand. The point at which the object is grasped acts as a pivot about which rotation occurs. Sliding is done by regulating friction between the finger surface and the object. To regulate friction, the opening between the fingers is increased to allow slipping of the object. Recent work by Shi et al.[7] carry out controlled sliding at the finger tips by pushing the object against a surface in the environment. These methods, in addition to relying on the environment, also require accurate hand-object models. The hand-object models have to be modelled dynamically, thereby requiring different values of friction coefficients. In contrast, the variable friction finger can work with simple kinematic models with quasi-static assumptions, which greatly simplifies the control approach and boosts system accuracy.

Recent researches have proved that underactuated hands [36] can successfully perform few of the within-hand manipulation tasks. Some of these tasks include controlled sliding without relying on the environment. Unlike anthropomorphic hands, underactuated hands have a simple design with simple actuations, making it easier to model the system and have simple control schemes.

R. Ma et al [10] uses an underactuated system with two fingers. One a finger has an active conveyor surface and the other is underactuated with one joint, with the surface having rollers. Some of the primitive tasks performed using this system are
controlled sliding, surface constrained sliding and out-of-plane alignment. Caging manipulation [37] is also another approach that can perform finger gaiting and sliding ensuring that the object always remains in the hand workspace and has a stable grasp. The variable friction finger [1] performs within-hand sliding and in-hand rotation of the object by leveraging the friction surfaces of the fingers. The simplicity of the system enables us to model the system and come up with planning and control strategies to manipulate the object to the desired position and orientation.

Inspired by the friction finger, Q. Lu et al. [8] proposed a way to switch from one friction surface to another using an origami inspired surface. The planning control schemes presented here, can be used for similar systems [9, 37, 38].

2.3 Control and Planning for Within-hand manipulation

Within-hand manipulation leads to accurate and efficient manipulation of the object. Therefore it is important to use optimal planning and control schemes to perform tasks. For multi fingered hands Li et al. [11] presented an approach to perform coordinated manipulation, rolling, sliding and finger gaiting while constraints are involved. Sundaramalingam et al. [23] proposes a planning strategy that alternates between finger gaiting and in-grasp manipulation to accomplish tasks while avoiding collisions. As discussed in the earlier section, multi fingered hands have a disadvantage of requiring detailed modelling of the hand-object system and friction forces between the object and robotic hand. Complex models lead to complex planning and control schemes.

Various recent methods relax the dependency on the accurate models by relying
on mechanical compliance [39] or joint-level impedance control [17]. Calli et al. [39] used visual servoing based on model predictive control to precisely manipulate the object using an underactuated Model T42 hand. In [17] the authors use finger gaiting for within-hand manipulation where geometric model of the object is required. In none of these works controlled sliding could be achieved, and rotational workspace is limited compared to the variable friction finger.

Dafle et al. has presented a few algorithms for systems that use extrinsic dexterity based on T-RRT* [40] and sampling based planning [41]. Sampling based planners does not guarantee a complete and an optimal solution. In addition, the plan returned by the sampling based methods needs post processing. The approach followed in this thesis, however, guarantees an optimal path without any need for post processing.

The offline planner developed in this thesis, is inspired by search based planning techniques [42, 43] for high dimensional spaces. The authors in [42] have generated a graph with nodes consisting of motion primitives and have used anytime heuristic search (ARA*) on the graph to find set of actions which needs to be executed to connect start and end pose. Our approach, is similar to this method, where we generate a graph of motion primitives offline and perform weighed A* search on it, to generate sequence of actions to be performed.

Learning based approaches have been proposed for the within-hand manipulation task demonstrating the generalization to unknown objects. These works [44, 45] utilizes model free reinforcement learning techniques to generate control policies, to perform in-hand manipulation. Even though these methods generalize well to unknown objects, they are inefficient and needs complex hyper-parameter tuning to make it run on a real system. The authors [46] have proposed a model based learning-controller which learns control policies for using low-level motion primitives
such as sliding, reposing and flipping of object to perform in-hand manipulation. In [47], the authors learn the dynamic model of the system. Later, they perform online planning to select optimal actions and execute it with a model predictive controller. The variable friction finger uses a simple kinematic model of the gripper system and hand with feedback loop control to perform within-hand manipulation task efficiently, accurately and also generalize well to objects of different sizes and shapes.

2.4 Object Tracking

Sensors play a vital role in within-hand manipulation[29]. It is important to know the position of the object with respect to the hand and to know the amount of force being exerted on the object to ensure grasp stability. Tactile sensors which are the most commonly used sensors in within-hand manipulation literature are expensive and hard to integrate to the mechanical design.

In this work, computer vision is used to determine the object’s position and orientation. One of the way to find the object’s pose is making use of ArUco markers[48]. Another approach used is to track the object with computer vision techniques alone without the use of any markers on the object. In this section, we will discuss a few methods used to track objects.

Optical flow [49] method tracks the object by computing the change in location of points belonging to the object solving the brightness constancy constraint [50, 51]. This equation is solved iteratively [52]. The points that are tracked are chosen based on feature detectors [53, 54, 55, 56]. Most of the objects manipulated are plain and has less features to track. Therefore, this method would not work well for the variable friction finger system.
Kalman filter [57] is another technique used for object tracking [58, 59]. It is a model based object tracking method which predicts the likely position of the object and corrects the equation of motion using the observations made. Extended Kalman filter (EKF) [60, 61] is also used for object tracking.

Object tracking can also be performed using learning methods like SVM [62], Neural Networks [63, 64], Adaptive boosting [65], Multiple Hypothesis tracking [66], Hidden Markov Models [67, 68].

In this present work, mean shift algorithm is used [69, 70] for object tracking. Mean shift algorithm is a non-parametric tracking method, which uses the distribution of HSV values to track the object. Continuously Adaptive Mean Shift (CAM shift) [71] is an adapted version of mean shift where the distribution of the HSV values are updated after every frame. Therefore, the distribution keeps changing after every frame whereas in mean shift the distribution of the HSV values remains the same since it is not being updated. In our case, the HSV distribution of the region of interest does not change during tracking, therefore we choose to use mean shift algorithm.
Chapter 3

System Description and Model

In this chapter, we summarize the design and functionalities of the VF gripper. For a more detailed description about the working of the finger [1] can be referred. A kinematic model of the system is presented, which will later be utilized in the planning and control schemes in Sections 4.1 and 4.2.

3.1 Description of the VF Gripper

Friction plays an important role in WIHM. Effective frictional forces between the finger and object’s surfaces can change the outcome of a particular finger motion on the object pose. The VF gripper is designed to allow modification of the effective friction at the contacts on demand. The VF finger hardware is open-source as part of the Yale OpenHand project [72]. CAD files and assembly instructions may be downloaded from https://www.eng.yale.edu/grablab/openhand.

Each finger’s body and low friction surface were 3D-printed in ABS (using a Stratsys Fortus 3D printer), whereas the high friction surfaces were molded using urethane rubber (Vytafelx 30, manufactured by Smooth-On). The ABS inserts offer lower friction and can be retracted using servo motors to expose the high friction
urethane rubber. This capability of changing friction allows switching between sliding and rolling of the object. Fig. 3.1 shows the design of each finger with high and low friction surfaces.

By setting the friction configuration appropriately, different WIHM behaviours can be achieved. As such, moving the fingers during the following friction conditions 1) \( \{ \text{left finger, right finger} \} = \{ \text{high, low} \} \), 2) \( \{ \text{low, high} \} \) and 3) \( \{ \text{high, high} \} \) leads to 1) sliding of the object along the right finger, 2) sliding of the object along the left finger and 3) rotation of the object, as shown in Fig.3.2.

Dynamixel Model-X actuators are used to control the motion of the fingers while miniature hobby servos (HiTec HS-85MG) toggle the variable friction surfaces between high and low conditions. As previously described in [1], at any moment, one of the Dynamixels is in velocity mode to lead the motion, while the other Dynamixel is in torque mode, to maintain finger contact with the object. The role of each Dynamixel actuator changes depending on the friction state of the gripper and direction of object motion, as also detailed in [1].
Figure 3.2: Figures 1-3 show a case of the object sliding on the right hand where the left finger is set to high friction (HF) and the right finger is set to low friction (LF). Figures 4-6 shows the object rotating within the hand. Here both the fingers are set to high friction

3.2 Kinematics Model

In this section we present a kinematics model of our hand-object system, which expands on the model presented in [1] to achieve sufficient detail for object pose control and planning. The kinematics provide a mapping between actuator space \((\theta_L, \theta_R)\), finger space \((d_L, d_R)\) and Cartesian space \((x, y)\). As also mentioned in Section 3.1 this mechanism is a switching system; the motion of the object depends on the finger friction configuration, and a kinematic model is derived for each of these configuration.

The model has four variables, \(d_L, d_R, \theta_L\) and \(\theta_R\), as illustrated in Fig. 3.3. Here, \(d_L\) is the distance from the left finger joint to the object’s bottom corner and \(d_R\) is the distance from the right finger joint to the object’s bottom corner. \(\theta_L\) and \(\theta_R\) are the angles of the fingers with respect to the positive X-axis in the gripper frame.
3.2.1 Within-hand sliding

Forward Kinematics

When the object slides, one of its surfaces maintains contact with the high-friction surface while the opposite surface slides against the low-friction finger. This constraint, of the object being effectively stationary against one finger, is used to find the position of the center of the object (or any other reference point in the object) in terms of the variables $d_L, d_R, \theta_L, \theta_R$. Only one finger is responsible for controlling the position of the object due to the torque-control contact-maintaining function on the other finger (as was outlined in Section 3.1). The hand-object system forms a closed kinematic chain that bridges both fingers with the object. Therefore, the position of the object can be expressed in terms either $[d_L, \theta_L]$ or $[d_R, \theta_R]$. We use $[d_L, \theta_L]$ or $[d_R, \theta_R]$ to formulate the object position depending on which finger the object is sliding against. The position of the center of the object while it slides...
against the right finger is given as follows:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \begin{bmatrix}
  (d_L + b)\cos(\theta_L) + (a + f_w)\sin(\theta_L) \\
  (d_L + b)\sin(\theta_L) - (a + f_w)\cos(\theta_L)
\end{bmatrix}
\] (3.1)

Similarly the equations for sliding on left finger is given in equation 3.2.

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \begin{bmatrix}
  w_p + (d_R + b)\cos(\theta_R) - (a + f_w)\sin(\theta_R) \\
  (d_R + b)\sin(\theta_R) + (a + f_w)\cos(\theta_R)
\end{bmatrix}
\] (3.2)

Here, \(a\) is the length of the perpendicular line drawn from the side in contact with the finger to the center of the object, \(b\) is the distance between the object’s bottom corner and the point of intersection of the perpendicular with the finger. The geometric parameters associated with the hand is the palm width (the distance between the finger joints), \(w_p\), and the finger width, \(f_w\), which is the distance between the line passing through the finger joint to the fingerpad, as illustrated in Fig. 3.3.

**Inverse Kinematics**

We can use the two equations in the \(X\) and \(Y\) directions given in equation (3.1), to find \(d_L, \theta_L\) from the objects position. The relation between the variables \(d_L, \theta_L\) and \(d_R, \theta_R\) is derived using the constraint that the vectors \(\vec{p}_1, \vec{p}_2, \vec{w}_o\) and \(\vec{w}_p\) form a closed loop during motion. This is illustrated in Fig. 3.4. The equation (3.3) shows the constraint used to compute \(d_R\) and \(\theta_R\) for given values of \(d_L\) and \(\theta_L\).

\[
\vec{p}_1 + \vec{w}_o - \vec{p}_2 - \vec{w}_p = 0
\] (3.3)
Figure 3.4: The vectors $\vec{p}_1, \vec{p}_2, \vec{w}_o$ and $\vec{w}_p$ form a closed loop while sliding occurs.

where,

$$\vec{p}_1 = \begin{bmatrix} (d_L + w_o)\cos(\theta_L) + f_w\sin(\theta_L) \\ (d_L + w_o)\sin(\theta_L) - f_w\cos(\theta_L) \end{bmatrix} \quad \vec{p}_2 = \begin{bmatrix} (d_R + w_o)\cos(\theta_R) - f_w\sin(\theta_R) \\ (d_R + w_o)\sin(\theta_R) + f_w\cos(\theta_R) \end{bmatrix}$$

(3.4)

$$\vec{w}_o = \begin{bmatrix} w_o\sin(\theta_L) \\ -w_o\cos(\theta_L) \end{bmatrix}, \quad \vec{w}_p = \begin{bmatrix} w_p \\ 0 \end{bmatrix}$$

(3.5)

**Jacobian**

In equation (3.1), we see that while the object is sliding along the right finger, the position of the object in the Cartesian space is dependent on $d_L$ and $\theta_L$. It must also be noted that in equation (3.1), the actuator angle, $\theta_L$ alone would vary when the object slides on the right finger, while $d_L$ remains constant throughout the sliding operation. When equation (3.1) is differentiated with respect to time we get the velocity of the object in terms of the velocity of the actuator. The velocity of the
object is related to the finger angles with a Jacobian matrix, \( v = J\dot{\theta} \). We thus determine the Jacobian, which we will use later for visual servoing. The velocity of the object when it is sliding on the right finger can be written as follows:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y}
\end{bmatrix} = \begin{bmatrix}
-(d_L + b)\sin(\theta_L) + (a + f_w)\cos(\theta_L) \\
(d_L + b)\cos(\theta_L) + (a + f_w)\sin(\theta_L)
\end{bmatrix} \dot{\theta}_L
\] (3.6)

The velocity of the object sliding in left finger can be obtained similarly by differentiating equation (3.2) with respect to time. Comparing equation (3.1) with (3.7), we get

\[
\begin{bmatrix}
\dot{x} \\
\dot{y}
\end{bmatrix} = \begin{bmatrix}
-y \\
x
\end{bmatrix} \dot{\theta}_L
\] (3.7)

Similarly the velocity of the object when it is sliding on the right hand is given in equation (3.8).

\[
\begin{bmatrix}
\dot{x} \\
\dot{y}
\end{bmatrix} = \begin{bmatrix}
-y \\
x - w_p
\end{bmatrix} \dot{\theta}_R
\] (3.8)

We observe that the Jacobian for the sliding motion on the left and the right finger is independent of the variables \( d_L, d_R, \theta_L, \theta_R \) and the dimensions of the object. This proves to be of great advantage when we use visual servoing as we shall discuss in Section 4.2.
3.2.2 Within-hand rotation

As shown in Fig. 3.2.2, when object rotation is taking place, the two pivot points about which the object is rotating are always in contact with the same point in the finger. Therefore, the distance between the two pivot points always remains the same. This constraint can be represented in the form of an equation as

\[ \| \vec{p}_1 - \vec{p}_2 \| = d \]  \hspace{1cm} (3.9)

The vectors \( \vec{p}_1 \) and \( \vec{p}_2 \) are represented in terms of the variables \( d_L, d_R, \theta_L \) and \( \theta_R \) and are shown in Equations (3.10) and (3.11) (refer Fig. 3.3).

\[ \vec{p}_1 = \begin{bmatrix} d_L \cos(\theta_L) + f_w \sin(\theta_L) \\ d_L \sin(\theta_L) - f_w \cos(\theta_L) \end{bmatrix} \]  \hspace{1cm} (3.10)

\[ \vec{p}_2 = \begin{bmatrix} d_R \cos(\theta_R) - f_w \sin(\theta_R) \\ d_R \sin(\theta_R) + f_w \cos(\theta_R) \end{bmatrix} \]  \hspace{1cm} (3.11)

\[ \vec{w_p} = \begin{bmatrix} w_p \\ 0 \end{bmatrix} \]  \hspace{1cm} (3.12)

After finding the vectors \( \vec{p}_1 \) and \( \vec{p}_2 \) using the above equations, we can calculate the Cartesian coordinates of the center of the object using (3.13).

\[ \begin{bmatrix} x \\ y \end{bmatrix} = \frac{\vec{p}_1 + \vec{p}_2}{2} \]  \hspace{1cm} (3.13)

The rotation of the object is defined in discrete angles. The object is kept rotating until the adjacent side comes in contact with the finger. We define this
as a complete rotation. For example an extruded hexagon must be rotated within the hand by $\pi/3$ for one complete rotation. To calculate the angle by which the finger should rotate for a complete rotation of the object, the side which is supposed to come in contact with the finger is tracked. Rotation is completed when the orientation of this side becomes the same as the orientation of the finger. This can be seen more clearly in Fig. 3.6.

The finger angles can also be calculated using constraint in (3.9). We observe that in case of anticlockwise rotation of the object $d_L$ becomes $d_L + w_o$ and $d_R$
becomes $d_R - w_o$, where $w_o$ is the side of the square. Similarly for clockwise rotation of the object $d_L$ becomes $d_L - w_o$ and $d_R$ becomes $d_R + w_o$. We update these values in the equation to find $\theta_L$ and $\theta_R$. Since, the relation between $(d_L, d_R)$ and $\theta_L, \theta_R$ is one-to-one (for given values of $(d_L, d_R)$ there exists only one solution of $(\theta_L, \theta_R)$.
Chapter 4

Planning and Control

4.1 Motion Planning

The VF gripper is non-holonomic and switching system. Therefore, right sequence of actions need to be determined to achieve a target object pose. This WIHM planning task can be formulated as a graph search problem of finding the shortest path between nodes. We construct a motion-primitive\(^1\) based graph offline. In our graph, the nodes are the poses of the object and edges are the actions of the VF system. This graph search problem can be solved using various techniques [73]. We use the A* algorithm with modifications in the cost function that allow smooth and efficient manipulation trajectory. The details of the search algorithm are discussed in the following sections.

\(^1\)The term “motion primitive” is used in planning literature to represent a basic (atomic) feasible motion or sometimes high level actions. We denote motion primitives to represent high level actions (Table 4.1)
Table 4.1: State Action Transition Table, for Moving the Object from State $s$ to State $s'$

<table>
<thead>
<tr>
<th>Action</th>
<th>Next State $s'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT SLIDE UP</td>
<td>$(d_L + \gamma_{slide}, d_R, \alpha)$</td>
</tr>
<tr>
<td>LEFT SLIDE DOWN</td>
<td>$(d_L - \gamma_{slide}, d_R, \alpha)$</td>
</tr>
<tr>
<td>RIGHT SLIDE UP</td>
<td>$(d_L, d_R + \gamma_{slide}, \alpha)$</td>
</tr>
<tr>
<td>RIGHT SLIDE DOWN</td>
<td>$(d_L, d_R - \gamma_{slide}, \alpha)$</td>
</tr>
<tr>
<td>ROTATE CLOCKWISE</td>
<td>$(d_L + l, d_R - l, \alpha - \gamma_{rotate})$</td>
</tr>
<tr>
<td>ROTATE ANTI-CLOCKWISE</td>
<td>$(d_L + l, d_R - l, \alpha + \gamma_{rotate})$</td>
</tr>
</tbody>
</table>

4.1.1 State Space, Action Space and Cost Functions

For the WIHM planning problem, the state variables are expressed in finger space, as this provides an intuitive way to formulate cost functions for the actions (an alternative way would be using Cartesian space, which has a one-to-one mapping to the finger space for the feasible object poses, and a very similar formulation would follow). The state variables are $s = (d_L, d_R, \alpha)$, where $\alpha$ is the object orientation, and $d_L$ and $d_R$ are as defined in Section 3.2 and shown in Fig. 3.3.

We discretize $d_L$ and $d_R$ by 1 mm displacements. The discretization of $\alpha$ depends on the object geometry. The action space $A$ with corresponding state action transitions are expressed in the Table 4.1. In this notation, LEFT SLIDE UP means sliding the object along the left finger in the positive (distal) direction, and so on.

The costs for sliding and rotating actions are given as below:

$$c_{sliding}(s, s') = \gamma_{slide}$$ (4.1)

$$c_{rotation}(s, s') = 2*l + \gamma_{rotate}$$ (4.2)
The heuristic function, \( h(s) \), is the estimated cost to move from the current state to the goal state.

\[
h(s) = h_{\text{pos}}(s) + h_{\text{or}}(s) \quad (4.3)
\]

where \( h_{\text{pos}}(s) \) is the Manhattan distance between the current position and goal position, and \( h_{\text{or}}(s) \) is the orientation difference between the current pose and the goal pose in absolute values.

### 4.1.2 A* Search Algorithm

The conventional A* [42] algorithm finds the shortest path from the start state to the goal state by expanding nodes in the order of increasing cost. The cost function is given as

\[
f(s') = g(s') + h(s') \quad (4.4)
\]

\[
g(s') = g(s) + c(s, s') \quad (4.5)
\]

where \( f(s') \) is the estimated cost from the start state to goal state, through \( s' \). \( g(s) \) is the accumulated cost of all traversed states from the start state to \( s \). \( g(s') \) is the cost of the state \( s' \) from the start state to \( s' \). \( h(s') \) is the estimated cost to reach goal state from \( s' \). Applying the conventional A* algorithm to our problem generates a sequence of actions connecting the start state to the goal state with the lowest possible cost.

A path generated by the A* planner for the initial pose \((X, Y, \alpha) = (7.0, 7.5, -90)\) and goal pose \((12, 12, 0)\) is shown in the Fig. 4.1a together with the expanded nodes for the search. It is important to note that, for the given initial position, it is not possible to directly perform the rotation action due to work-space limitations on the finger angles. Nevertheless, the path planner successfully leads the system to a
configuration where the rotation becomes possible, and then performs it. After the rotation, the object is slides on the left and right fingers until it reaches the goal pose.

The sliding actions after the rotation includes high frequency chattering. The reason for this non-smooth behavior is that the sliding actions need to switch between the left and right fingers frequently in order to reach the goal position in the shortest possible way. This is due to the non-holonomic nature of the VF finger system. Such chattering is highly undesired for our system, since it requires frequent switching between finger modes (i.e. \{high\_friction, low\_friction\} to \{low\_friction, high\_friction\} and vice versa), which does not only take time, but also causes wear and tear of the hardware and energy inefficiency. To solve this problem, we are unable to utilize conventional trajectory smoothing techniques [74], which are based on polynomial interpolation and curve fitting, as our system performs discrete rotations that are difficult to model in those frameworks. Therefore, we propose the following modifications in the heuristic function.

### 4.1.3 Modified Heuristic Cost Function

Generally, a node in the A* search expansion tree comprises of the following information: state $s$, action $a$, cost $f(s)$, and a pointer to the parent node $P$. The parent node is the node from which the current node is expanded, and the parent node action is given by $P_a$. Smooth paths can be achieved by relaxing the optimality constraint, and directing the search algorithm to balance between the path smoothness and length. This is achieved by assigning low costs to nodes which are expanded by the same action as their parent node. We define the heuristic cost $h(S)$ of a node as follows:
\[ h(S) = \beta \ast h(S) \]  \hspace{1cm} (4.6)

where

\[
\beta = \begin{cases} 
0 < \beta < 1, & \text{if } a == P_a \\
1, & \text{otherwise}
\end{cases} \hspace{1cm} (4.7)
\]

Since the A* search algorithm expands the nodes with lowest cost first, the modified heuristic cost function will make the algorithm choose a smoother trajectory by favoring the parent node action, sometimes at the expense of the shortest path. The balance between the path length and trajectory smoothness is determined by the parameter \( \beta \), with a lower \( \beta \) meaning a smoother path. This heuristic function is applied to the same initial and goal positions as previously defined in Section 4.1.2. Results are given for \( \beta = 0.7 \) in Fig.4.1b.

It can be observed that the path is smooth, but the number of nodes expanded are more than the conventional A* search in Fig. 4.1a which makes the search algorithm significantly slower.

### 4.1.4 Weighted A* Search

In order to speed up the search algorithm, the modified heuristic cost \( h(s) \) is multiplied with a constant \( \epsilon \) and then added to \( g(s) \) to calculate the final cost as shown in (4.8). This results in a weighted A* search [75].

\[
f(S) = g(s) + \epsilon \ast h(S) \quad \text{where } \epsilon > 1 \hspace{1cm} (4.8)
\]

This creates a bias towards the goal state by expanding more nodes, which are closer to the goal to speed up the search. However, again, the solution doesn’t guarantee an optimal path, but an \( \epsilon \) sub-optimal path. Fig. 4.1c shows the path from
Figure 4.1: Planned path using motion planner from start state (X, Y, α = 7.0, 7.5, -90) to goal state (12, 12, 0) (a) Conventional A* search (not smooth). (b) A* search with modified cost function (smoother) (d) Weighted A* search (not smooth but fast solution). (d) Weighted A* search with modified cost function (smooth and fast solution). Axis units are cm.

the weighed A* without the heuristic modification in Section 4.1.3, and Fig. 4.1d shows the path from the weighed A* and the heuristic modification combined. The combined case provides smoother and faster trajectories compared to the conventional A* approach shown in Fig. 4.1a. The parameters $\epsilon$ and $\beta$ can be tuned as per requirements.
4.2 Visual Servoing for WIHM

We assume an eye-to-hand scheme, i.e. a camera observes the hand-object system. We propose two vision-based control schemes: a pure vision-based control, and a hybrid approach that utilize both the offline plan and the online feedback which is discussed in Section 4.3.

We use a classical Position-based Visual Servoing scheme [76], which finds the error in the Cartesian space and generates the velocity reference to minimize the error between the current and desired poses. The velocity reference is generated using the equation,

\[ v_{ref} = -\lambda J_i^\dagger e \] (4.9)

where \( J_i^\dagger \) is the pseudo-inverse of the image Jacobian also known as the interaction matrix, \( e \) is the feature error vector (\( e = [e, \theta u] \)), and \( v_{ref} \) is the Cartesian velocity reference for the object. The image Jacobian is given as

\[
L_e = \begin{bmatrix}
R & 0 \\
0 & L_{\theta u}
\end{bmatrix}
\] (4.10)

Where \( L_{\theta u} \) is given as \( L_{\theta u} = I_3 - \theta [u]_x/2 + (1 - \frac{\text{sinc} \theta}{\text{sinc}^2 \theta}) [u]^2_x \) (\( \text{sinc} \theta \) is defined so that \( x \text{sinc} x = \sin x \) and \( \text{sinc} 0 = 1 \)).

This velocity reference cannot be directly applied to the system due to two reasons. First, the system is non-holonomic, and cannot directly follow the linear velocities generated by the visual servoing algorithm. Secondly, the system can only perform discrete rotations, and therefore cannot follow continuous rotational velocities. Therefore, the position error and the orientation error is decoupled.
Position correction step

When the equation (4.10) is decoupled, the velocity reference for position correction is formulated as

\[ v_{ref} = -\lambda R^T e \]  \hspace{1cm} (4.11)

Where \( R \) is the rotation of the camera frame with respect to the base frame of the robot. The orientation of the camera is set to the orientation of the base frame so that \( R \) becomes an identity matrix. The equation is now simplified to

\[ v_{ref} = -\lambda e \]  \hspace{1cm} (4.12)

The position errors is corrected by sliding the object either along the left finger or the right finger. Now, \( v_{ref} \) in equation 4.12 is the velocity reference in the Cartesian space. To convert it to actuator velocity references we have to multiply it by the inverse of the Jacobian for sliding motion. As explained in Section 3.2, there are separate Jacobians for each of these cases. So, in each time step, we calculate both of these Jacobians then calculate the expected error for each action, and choose the one that reduces the error more.

The Jacobians of the robot we derived in Section 3.2 were independent of the parameters \( d_L, d_R, \theta_L, \theta_R \) and the object dimensions. When it comes to accuracy, this gives visual servoing an upper hand compared to the motion planner. The motion planner relies on exact modelling of the system and cannot handle any inaccuracies making it less accurate. On the other hand, since visual servoing is independent of actuator and finger variables it is not relying on the modelling of the system.
Orientation correction step

A complete rotation of the object (e.g. 90° for a rectangular prism) cannot be initiated from all points in the workspace, since this action requires large object displacements that may lead the object to the workspace limits. The motion planner in Section 4.1 finds a path that brings the object to a location, where rotation is feasible (e.g. Fig. 4.1), but visual servoing algorithm is essentially a local optimizer, and lacks the mechanisms to do so. In order to overcome this disadvantage, we first move the system close to the edges of the workspace, where rotations can easily be conducted (right-side edge for clockwise rotation and left-side edge for anti-clockwise rotation of the object), and then execute the rotation action. These locations are predetermined and hard-coded.

The rotation correction is performed by tracking the orientation of the object. When rotation occurs the side which is in contact with the finger gives way to its adjacent side. The object is kept rotating until the orientation of the adjacent side becomes equal to the orientation of the finger (refer Fig.3.6).

Once the rotation is corrected, the position correction steps are carried out. The object trajectories obtained with this strategy may end up being far from optimum. Nevertheless, combining the offline planner and the visual servoing algorithm provides an alternative solution as follows.

4.3 Hybrid Approach

Executing the path of the offline motion planner directly on the real hardware results in large inaccuracies (as shown in the Fig. 6.4-b) mainly due to the inaccuracies of the hand-object model. While utilizing vision feedback can make the system robust to these inaccuracies, using it in a standalone manner may not be preferable either
Figure 4.2: The course of action in the hybrid approach

since 1) we need to utilize some rules to manage the discrete rotation motions, which results inaccurate trajectories, 2) high frequency chattering happen during convergence (Fig. 6.4-a). Therefore, we propose a hybrid controller, in which the offline planner determines via points, and the visual servoing algorithm tracks these intermediate goals in a close loop system. Such an approach results in both smooth and accurate WIHM Fig. 6.4-c. The hybrid approach is illustrated in fig. 4.2.
Chapter 5

Object Tracking

5.1 Object tracking using ArUco Markers

ArUco markers [48] is a square marker that consists of a codification of black and white bits. Each combination of these binary codification is uniquely labelled with a marker identifier. These markers consists of a black boundary which makes its detection easier. In this thesis, we use these markers to find the position of the object in the image. We use two ArUco makers, the first ArUco marker is present at the origin and the second is on the object that is being manipulated. Since the detection of the markers is done in the image space we have to calibrate the camera to find its position in Cartesian space. After we calibrate the camera, the aruco_ros library is able to publish the position of the markers in metres and the orientation in radians. The position of the object is the difference between the marker coordinates present on the object and the marker coordinates present on the origin. The orientation of the object is also found similarly. However, it is not always possible to rely on the use of markers for the object detection. In Section 5.2 we discuss tracking of the object without the use of these markers.
5.2 Object tracking using Mean shift

Mean shift algorithm has proved to be an efficient technique tracking objects. It tracks the object by making use of the probability distribution of the features in the target object and matches it in the image by finding the area with a similar probability distribution. The Mean shift algorithm also proves to be efficient because it detects the object using less expensive exhaustive search.

In this present work, we choose the HSV(Hue-Saturation-Value) values to be the feature space. In the first frame, we select a region of interest around the object to be detected. The next step is to create a histogram of the HSV values of the region of interest. This histogram is used to come up with the probability distribution function of the features in the target object. Each combinations of the HSV values is considered as a bin. For a total of $u$ bins, the probability distribution function is represented as $\hat{q} = \{q_u\}_{u=1,2,3...m}$ and should satisfy $\sum_{u=1}^{m} \hat{q}_u = 1$. The probability distribution function is calculated using a weighted histogram.

$$p(u) = C \sum_{i=1}^{n} k(\|x_i\|^2)\delta(S(x_i) - u)$$  \hspace{1cm} (5.1)

Here, $n$ is the number of pixels in the region of interest window and $x_i$ is the ith element in the set. $k$ is the kernel function which gives the weights depending on the distance of the pixel from the center and $\delta$ is the Kronecker delta function which is 1 if $S(x_i) = u$ and 0 otherwise and the function $S$ associates the pixel to its feature value. $C$ is the constant so that $\sum_{u=1}^{m} \hat{q}_u = 1$. We use the same formula to find the probability distribution of features in any image window. In each frame that follows, we need to find the probability distribution for candidate windows where
the target object could be present. Employing the same formula, we arrive at

\[ \hat{p}_u(u) = C_c \sum_{i=1}^{n} k(\|y - x_i\|^2)\delta(S(x_i) - u) \]  \hspace{1cm} (5.2)\]

where \( y \) is the center of the image window and \( C_c \) is a constant so that \( \sum_{u=1}^{m} \hat{p}_u = 1 \).

To find the location of the object in the image, we need to find an image window with the closest distribution of the initial region of interest. Two functions are similar when the dot product of them is maximized. To find the similarity between the target distribution and candidate distribution, we use the Bhattacharya coefficient. The Bhattacharya coefficient given by the sum of the element-wise product of each bin between the target distribution and the candidate distribution \( \rho(y) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y)q_u} \). We use this to find the distance between the target and the candidate

\[ d(y) = \sqrt{1 - \rho(y)} \]  \hspace{1cm} (5.3)\]

We note that if two functions are similar, the Bhattacharya coefficient \( \rho(y) \) is maximum and the distance \( d(y) \) is minimum. Therefore we need to maximize the Bhattacharya coefficient. If we approximate the Bhattacharya coefficient at \( y_0 \) using Taylor series expansion we get,

\[ \rho(\hat{p}(y), q) \approx \rho(\hat{p}(y_0), q) + \frac{1}{2} \sum_{i=1}^{m} \hat{p}_u(y) \sqrt{\frac{q_u}{\hat{p}_u(y_0)}} \]  \hspace{1cm} (5.4)\]

Substituting the value of \( \hat{p}_u(y) \) in (5.2) we get,

\[ \rho(\hat{p}(y), q) \approx \rho(\hat{p}(y_0), q) + \frac{C_c}{2} \sum_{i=1}^{n} \left[ \sum_{u=1}^{m} \delta(S(x_i) - u) \sqrt{\frac{q_u}{\hat{p}_u(y_0)}} \right] k(\|y - x_i\|^2) \]  \hspace{1cm} (5.5)\]
Since $\rho(\hat{p}(y_0), q)$ is a constant, the value of $\sum_{u=1}^{m} \delta(S(x_i) - u) \sqrt{\frac{q_u}{p_u(y_0)}}$ has to be maximum for the distributions which are similar. To do this, we use mean shift.

Mean shift is an algorithm that uses gradient ascent to determine the local maxima of a distribution. The mean shift vector gives the direction towards which the mean is to be shifted so that we move in the direction of the maximum increase. For a set of $n$ data points, the mean shift for a point located at $y_0$ is calculated using the formula,

$$M_h(y_0) = \left[ \frac{\sum_{i=1}^{n} w_i(y_0) x_i}{\sum_{i=1}^{n} w_i(y_0)} \right] - y_0$$

(5.6)

where $w_i$ is the weight of $x_i$. For a kernel function $k$ the density is given by $P(x) = \frac{1}{n} \sum_{i=1}^{n} k(\|x - x_i\|)$. When we compute the mean shift for this we get,

$$M_h(y_0) = \left[ \frac{\nabla P(x)}{\frac{1}{n} \sum_{i=1}^{n} g_i} \right]$$

(5.7)

where $g_i = g(\|x - x_i\|^2) = \nabla k(\|x - x_i\|^2)$

Algorithm 1: Mean shift algorithm for object tracking

Result: Object tracking using mean shift

Create a Histogram of the object region with the HSV values as bins;

Compute the probability distribution function;

In the next frame compute the probability distribution of the candidate ;

Maximize the Battacharya coefficient using mean shift;

The region with the largest Battacharya coefficient is where the object is present;
Chapter 6

Experiments and Results

We conduct experiments with the VF system and compared the performance of the three approaches for conducting the within-hand manipulation task: 1) feedforward control method based on a WIHM planner, 2) feedback control method using a visual servoing scheme, 3) the hybrid method that combines the feedforward and feedback methods. The experimental setup is constructed as shown in the Fig. 6.1. The camera is set to 640x480 pixels resolution and 30 f/s capture rate. The distance of the camera to the object top surface is 20 cm.

6.1 Experiments with Objects of different shapes

We test each of these methods for five differently shaped objects (shown in Fig. 6.2). For each task, 4 randomly chosen sets of feasible start and goal states are generated (refer Table 6.3 and Fig. 6.3). goal-1, which requires only position correction, goal-2, which requires position correction and one clockwise rotation, goal-3, which requires position correction with one anti-clockwise rotation, and goal-4, which requires position correction with two clockwise rotations. As mentioned in the earlier section, we define “one rotation” as switching the contact to the adjacent edge of the object.
Figure 6.1: The experimental setup used in this work.

Figure 6.2: Objects used in the experiments
Figure 6.3: Goal positions for the object to reach (Refer Table 6.3)

<table>
<thead>
<tr>
<th>Task no.</th>
<th>Desired Position (in cm)</th>
<th>Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>-0.776, 9.518</td>
<td>No rotation</td>
</tr>
<tr>
<td>2.</td>
<td>1.081, 10.254</td>
<td>1 rotation clockwise</td>
</tr>
<tr>
<td>3.</td>
<td>1.817, 9.762</td>
<td>1 rotation anti-clockwise</td>
</tr>
<tr>
<td>4.</td>
<td>4.003, 9.923</td>
<td>2 rotations clockwise</td>
</tr>
</tbody>
</table>
For objects of different shapes, the mean and variance of the performance metrics are applied to each method. The results for the four different goals shown in Table 6.3 and Fig. 6.3. The parameters for the motion planner described in Section 4.1 are set as $\epsilon = 2$ and $\beta = 0.7$ to get a smooth solution.

The results for each task is shown in Table 6.1. We see that, the feedforward planner is the most efficient because it takes the least number of actions. The planner is also able to complete the tasks in the shortest time and distance. However, there is always a steady state error after the execution. Visual servoing proves to be an accurate method but we see that it is inefficient in terms of time taken, path length and switching actions. This is due to large number of switching actions caused by trying to follow a straight path with the non-holonomic system. The difference in these quantities is more visible where rotation is involved. The planner has an optimal strategy to perform the rotation of the object. It is also important to note that visual servoing algorithm does not require any offline planning, which can be an advantage in some scenarios. The hybrid method provides us with the best performance since it can handle both position and orientation correction precisely by combining the advantages of the offline and online methods. The performance of these methods are illustrated in Fig. 6.4, showing the comparison in performance of the three methods.

### 6.2 Experiments with Modelling inaccuracies

The simplicity of the hand-object model reduces the reliance on an accurate model. While the Planner still depends on the model, we see in Section 4.2 that Visual Servoing is independent of the object size and the state variables $d_L, d_R, \theta_L$ and $\theta_R$. The Hybrid method takes advantages of the planner and the visual servoing and
Table 6.2: Performance of each method done with objects of different shapes

a. **Goal1: Position Correction**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error (cm)</th>
<th>Switching Actions</th>
<th>Path length (cm)</th>
<th>Time Taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Planner</td>
<td>1.88 ±0.98</td>
<td>3.0 ± 0.0</td>
<td>20.20 ± 6.77</td>
<td>20.0 ± 2.16</td>
</tr>
<tr>
<td>Visual Servoing</td>
<td>0.42 ± 0.06</td>
<td>6.25 ± 1.5</td>
<td>26.54 ± 8.69</td>
<td>27.5 ± 1.91</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.35 ± 0.01</td>
<td>4.25 ± 0.5</td>
<td>24.54 ± 4.96</td>
<td>22.75 ± 1.23</td>
</tr>
</tbody>
</table>

b. **Goal2: Position correction with one complete clockwise rotation**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error (cm)</th>
<th>Switching Actions</th>
<th>Path length (cm)</th>
<th>Time Taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Planner</td>
<td>1.15 ± 0.51</td>
<td>4.0 ± 0.0</td>
<td>23.05 ± 4.59</td>
<td>24.2 ± 2.06</td>
</tr>
<tr>
<td>Visual Servoing</td>
<td>0.35 ± 0.07</td>
<td>4.75 ± 1.5</td>
<td>27.57 ± 6.31</td>
<td>30.7 ± 3.60</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.37 ± 0.01</td>
<td>5.25 ± 0.5</td>
<td>27.15 ± 5.22</td>
<td>26.7 ± 2.36</td>
</tr>
</tbody>
</table>

c. **Goal3: Position correction with one complete anti-clockwise rotation**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error (cm)</th>
<th>Switching Actions</th>
<th>Path length (cm)</th>
<th>Time Taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Planner</td>
<td>1.18 ± 0.48</td>
<td>4.25 ± 0.5</td>
<td>21.89 ± 5.17</td>
<td>25.50 ± 3.1</td>
</tr>
<tr>
<td>Visual Servoing</td>
<td>0.41 ± 0.11</td>
<td>5.50 ± 0.58</td>
<td>29.76 ± 7.31</td>
<td>29.75 ± 4.47</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.38 ± 0.09</td>
<td>4.75 ± 1.5</td>
<td>28.16 ± 4.91</td>
<td>31.0 ± 3.54</td>
</tr>
</tbody>
</table>

d. **Goal4: Position correction with two complete clockwise rotations**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error (cm)</th>
<th>Switching Actions</th>
<th>Path length (cm)</th>
<th>Time Taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Planner</td>
<td>1.35 ± 0.69</td>
<td>5.0 ± 0.0</td>
<td>21.53 ± 5.17</td>
<td>27.50 ± 2.65</td>
</tr>
<tr>
<td>Visual Servoing</td>
<td>0.45 ± 0.04</td>
<td>5.5 ± 1.7</td>
<td>27.56 ± 7.73</td>
<td>35.25 ± 3.77</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.38 ± 0.09</td>
<td>5.25 ± 0.6</td>
<td>26.22 ± 6.44</td>
<td>31.50 ± 3.37</td>
</tr>
</tbody>
</table>
a) Visual servoing method

b) Offline motion planner method

c) Hybrid method

Figure 6.4: Trajectory tracked by the object from start pose (7,7,0) to goal pose (12,12,0) for different methods a) Visual servoing b) Offline motion planning c) Hybrid method
Table 6.3: Summary of Experiments with Square Prism focusing on Modelling inaccuracies

<table>
<thead>
<tr>
<th></th>
<th>Accuracy (cm)</th>
<th>Switching Actions</th>
<th>Path length (cm)</th>
<th>Time taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correct Object Size</td>
<td>Wrong Object Size</td>
<td>Correct Object Size</td>
<td>Wrong Object Size</td>
</tr>
<tr>
<td>Motion Planner</td>
<td>1.16</td>
<td>1.97</td>
<td>4.25</td>
<td>4.25</td>
</tr>
<tr>
<td>Visual Servoing</td>
<td>0.40</td>
<td>0.36</td>
<td>5.75</td>
<td>7.75</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.29</td>
<td>0.15</td>
<td>5.75</td>
<td>5.5</td>
</tr>
</tbody>
</table>

provides us with a fast and smooth execution without any steady state errors. The robustness of the methods to modelling inaccuracies is compared by introducing 20% error to the object size in the hand-object model.

Table 6.2 shows a comparison of the performance of each method with an accurate model and an inaccurate model. We notice that the accuracy of the feedforward controller is reduced. This is due to the inaccuracies introduced in the model. The independence of the visual servoing on the model does not affect the performance in spite of the inaccuracy. We see that visual servoing is still able to accurately manipulate the object to the desired location. In the hybrid method, since we use visual servoing for position correction, this method is also able to perform accurately. As in the previous experiments, the hybrid method provides us with the best performance.
6.3 Experiments without the use of ArUco markers

We conducted experiments with the five different objects where no rotation is involved. The mean shift algorithm requires the HSV values of the object as input. We also compare the trajectories when the object is tracked using the ArUco marker and using mean shift. This method proves to be accurate method for tracking objects and can be a reliable alternative when ArUco markers cannot be used.

Table 6.3 summarizes the results with different objects. We see that we are able to achieve good accuracy. Fig. 6.5 shows the path the object follows while it is being manipulated.

We notice that there is no difference in the performance when the object is tracked using the mean shift algorithm in comparison to using ArUco markers. To test the reliability of the mean shift tracking, we compare the resulting paths when the object is tracked using mean shift and ArUco markers. The result is shown in Fig. 6.6.
Table 6.4: Summary of Experiments using mean shift algorithm to track the object

<table>
<thead>
<tr>
<th>Shape</th>
<th>Error(cm)</th>
<th>Switching Actions</th>
<th>Path length (cm)</th>
<th>Time Taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.51</td>
<td>5</td>
<td>11.12</td>
<td>22</td>
</tr>
<tr>
<td>Rectangle</td>
<td>0.46</td>
<td>9</td>
<td>23.03</td>
<td>28</td>
</tr>
<tr>
<td>Pentagon</td>
<td>0.41</td>
<td>8</td>
<td>14.03</td>
<td>28</td>
</tr>
<tr>
<td>Hexagon</td>
<td>0.38</td>
<td>10</td>
<td>19.86</td>
<td>32</td>
</tr>
</tbody>
</table>

Figure 6.6: Comparison of mean shift algorithm with ArUco markers
Chapter 7

Conclusion

In this thesis, we showcase different frameworks to perform in-hand manipulation of objects using the VF finger gripper system. The frameworks have been tested with modelling inaccuracies and with uncertainties in the motion execution. The experimental results show that the hybrid framework performs faster and gives smoother paths as it combines the advantage of a pre-defined path from the motion planner with online correction from visual servoing. This clearly shows that even with inaccurate models of the system it is possible to perform in-hand manipulation of objects with our approach.

7.1 Future work

An online planner can along with visual servoing in the feedback correction loop can address the uncertainties in the environment more efficiently. Additionally, incorporating reinforcement learning methods, that will learn policies for within-hand manipulation would provide for efficient execution of tasks even with uncertainties. Thus making the VF gripper system more robust to uncertainties and allowing us to extend this approach to objects of different shapes and sizes.
The variable friction finger is limited by objects which are thin and smaller sized objects. This is because it cannot grasp the object at all points in the finger. Having multiple joints in the fingers or a movable joint at the base would help us manipulate objects of different sizes.

This system is able to perform only planar within-hand manipulation. Incorporating extrinsic dexterity to the gripper would make within-hand manipulation in other dimensions possible. Using the variable friction finger as the gripper at the end-effector and some of the approaches existing in literature [6, 34] would help us accomplish more tasks. The idea of leveraging friction can be extended to hands with more than two fingers. This is another way to carry out within-hand manipulation in higher dimensions. However, having more fingers poses the challenge of developing simple planning and control techniques.
Bibliography


[9] Seita Nojiri, Kaori Mizushima, Yosuke Suzuki, Tokuo Tsuji, and Tetsuyou Watanabe. Development of contact area variable surface for manipulation re-


