



8-2015

Reactive Motions In A Fully Autonomous CRS Catalyst 5 Robotic Arm Based On RGBD Data

Arad Haselirad

Follow this and additional works at: <https://commons.und.edu/theses>



Part of the [Mechanical Engineering Commons](#)

Recommended Citation

Haselirad, Arad, "Reactive Motions In A Fully Autonomous CRS Catalyst 5 Robotic Arm Based On RGBD Data" (2015). *Theses and Dissertations*. 1330.

<https://commons.und.edu/theses/1330>

This Thesis is brought to you for free and open access by the Theses, Dissertations, and Senior Projects at UND Scholarly Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UND Scholarly Commons. For more information, please contact zeinebyousif@library.und.edu.

REACTIVE MOTIONS IN A FULLY AUTONOMOUS CRS CATALYST 5
ROBOTIC ARM BASED ON RGBD DATA

by

Arad Haselirad
Bachelor of Science, Ferdowsi University of Mashhad, Iran, 2012

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

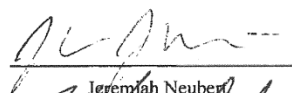
Master of Science

Grand Forks, North Dakota

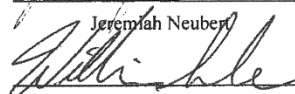
August

2015

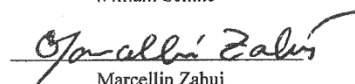
This thesis, submitted by Arad Haselirad in partial fulfillment of the requirements for the degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.



Jeremiah Neuber

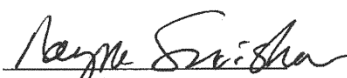


William Semke

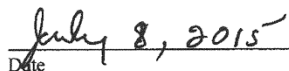


Marcellin Zahui

This thesis meets the standards for appearance, conforms to the style and format requirements of the Graduate School of the University of North Dakota, and is hereby approved.



Wayne Swisher
Dean of the School of Graduate Studies



Date

PERMISSION

Title Reactive motions in a fully autonomous CRS Catalyst 5 robotic arm
 based on RGBD data

Department Mechanical Engineering

Degree Master of Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this university shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in his absence, by the chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Arad Haselirad

May, 21, 2015

TABLE OF CONTENTS

LIST OF FIGURES	vi
LIST OF TABLES	ix
ACKNOWLEDGMENTS	x
ABSTRACT	xi
CHAPTER	
I. INTRODUCTION	1
Background and motivation	2
Thesis overview.....	9
II. THEORY AND METHOD.....	10
Object Detection and Tracking.....	13
Camera Calibration.....	13
Object Detection.....	15
Velocity Estimation	24
Velocity Estimation Formulation	24
Control Model.....	31

III.	EQUIPMENTS AND EXPERIMENTAL SETUP.....	38
	Experimental Setup.....	38
	Coordinate Transformation.....	39
	Equipments	41
	Open-Architecture Configuration	45
IV.	EXPERIMENT, RESULTS, AND DISCUSSION.....	52
V.	CONCLUSION.....	67
	REFERENCE.....	70

LIST OF FIGURES

Figure	Page
1. A racquetball player.....	2
2. KUKA robotic arm playing table tennis against Timo Boll.....	3
3. The ball juggling robot from DARPA	3
4. EPFL arm while catching a tennis racquet which has an irregular shape.....	4
5. Multifingered robotic arm juggling with two balls.....	5
6. Humanoid robot while throwing and catching a ball.....	6
7. The complete chart of the process.....	11
8. The graphical image blur of a moving ball.....	15
9. The real world target's RGB representation with its approximated center.....	17
10. The rotated object along with the horizontal and vertical intensity profiles a) The intensity profile along the horizontal line b)The intensity profile along the vertical line.....	18
11. The fitted rectangle and the area to determine the blur parameters.....	19
12. The calculation of motion blur parameters and the intensity profile sample...	21
13. The calculated distances on the actual object.....	21

14.	The step by step process on a sample object.....	22
15.	The system flowchart for the blurred parameters detection procedure.....	22
16.	Determining the angle between the object's motion direction and camera plane.....	25
17.	Pinhole camera model for speed estimation.....	27
18.	Pinhole model for the special case of parallel moving object.....	28
19.	The world coordinate frame of the system on the base of the robotic arm.....	30
20.	The velocity estimation along with position.....	30
21.	The inverse-kinematics model solution for the manipulator.....	34
22.	The schematic configuration of the system, the information is transferred from Kinect to PC2 without feedback.....	38
23.	The position of the camera and the robot.....	40
24.	CRS Catalyst 5 with the world coordinate system, the origin of the coordinate system is on the base of the arm.....	42
25.	Kinect's coordinate system.....	44
26.	Schematic of the discrete controller inside the real-time kernel.....	46
27.	The stream server block in Simulink.....	47
28.	The motor commands block details.....	49
29.	Smooth absolute joint commands block.....	50
30.	CRS position controller.....	51

31.	The RGBD data after filtering the contours.....	53
32.	The location of the Kinect and the robot.....	54
33.	The 3D representation of the end-effector's motion boundaries within 0.1 sec.....	58
34.	The top view representation of the end-effector's motion boundaries within 0.1 sec.....	59
35.	The top view of the working volume and a moving object.....	60
36.	The 3D view of the working volume and a moving object.....	60
37.	The images from left to right and top to bottom show the interception with the thrown cup, the last image depicts the moment of interception.....	61
38.	The images from left to right show the interception with the thrown cup in a different location, the last image shows the moment after interception.....	62

LIST OF TABLES

Table	Page
1. Actual position of the points on the edges of object and blur in the sample picture...	23
2. DH tables with parametrical elements.....	34
3. The joints and their corresponding angular ranges.....	41
4. The Cartesian limits.....	42
5. The DH table of CRS Catalyst 5.....	43
6. World coordinate kinematic limits of the equipment.....	55
7. The velocity data of the experiments.....	63

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor, Dr. Jeremiah Neubert for all the help, guidance and cooperation he has given me over the past years. I would like to thank my committee members, Dr. William Semke and Dr. Marcellin Zahui for their support and guidance towards my thesis.

In addition, I thank and appreciate all the help and assistance I received from Ashraf Qadir during the past years. He is a helpful, kind and knowledgeable friend.

To all those who lost their lives for the freedom of humankind

ABSTRACT

This study proposes a method to perform velocity estimation using motion blur in a single image frame along x and y axes in the camera coordinate system and intercept a moving object with a robotic arm. It will be shown that velocity estimation in a single image frame improves the system's performance. The majority of previous studies in this area require at least two image frames to measure the target's velocity. In addition, they mostly employ specialized equipments which are able to generate high torques and accelerations.

The setup consists of a 5 degree of freedom robotic arm and a Kinect camera. The RGBD (Red, Green, Blue and Depth) camera provides the RGB and depth information which are used to detect the position of the target. As the object is moving within a single image frame, the image contains motion blur. To recognize and differentiate the object from blurred area, the image intensity profiles are studied. Accordingly, the method determines the blur parameters based on the changes in the intensity profile. The aforementioned blur parameters are the length of the object and the length of the partial blur. Based on motion blur, the velocities along x and y camera coordinate axes are estimated. However, as the depth frame cannot record motion blur, the velocity along z axis in the camera coordinate frame is initially unknown. The vectors of position and velocity are transformed into world coordinate frame and subsequently, the prospective position of the object, after a predefined time interval, is predicted. In order to intercept, the end-effector of the robotic arm must

reach this predicted position within the time interval as well. For the end-effector to reach the predicted position within the predefined time interval, the robot's joint angles and accelerations are determined through inverse kinematic methods. Then the robotic arm starts its motion. Once the second depth frame is obtained, the object's velocity along z axis can be calculated as well. Accordingly, the predicted position of the object is recalculated, and the motion of the manipulator is modified.

The proposed method is compared with existing methods which need at least two image frames to estimate the velocity of the target. It is shown that under identical kinematic conditions, the functionality of the system is improved by 4.96 times for our setup. In addition, the experiment is repeated for 25 times and the velocity data is recorded. According to the experimental results, there are two major limitations in our system and setup. The system cannot determine the velocity along z in the camera coordinate system from the initial image frame. Consequently, if the object travels faster along this axis, it becomes more susceptible to failure. In addition, our manipulator is an unspecialized equipment which is not designed for producing high torques and accelerations. Accordingly, the task becomes more challenging.

The main cause of error in the experiments was operator's. It is necessary to have the object pass through the working volume of the robot. Besides, the object must be still inside the working volume after the predefined time interval. It is possible that the operator throw the object within the designated working volume, but it leaves it earlier than the specified time interval.

CHAPTER I

INTRODUCTION

This work proposes a method to autonomously detect a moving object outside the working volume of a robotic arm, estimate its velocity along x and y axes in the camera coordinate system in a single image frame using motion blur, predict its future location after a predefined time interval and intercept the object in the predicted location.

It will be shown how the velocity estimation in a single image frame increases the robot's maximum reach within the predefined time interval. The results of the method will be compared to those which need at least two image frames to calculate the velocity of the target. The comparison is made under identical kinematic conditions for the equipment and object. It will be demonstrated that the method is independent from the equipment's architecture and it is valid for any other type of robotic arm.

In this project the interception is executed without a specialized equipment such as high torque motors. While the majority of previous research works in this area have employed specialized robots to perform fast motions. Accordingly, the task becomes considerably more constrained for our robotic arm.

Background and Motivation

The topic of reactive and fast motions have long been studied in the area of robotics. The motivation for such research comes from a similar human ability to react to the surrounding dynamic environment [1]. Humans are able to generate short and quick motions in response to a stimulus. Prior research has proved that humans can anticipate the motion to be successful at a task [2]. This type of motion is defined as reactive. The ability to respond to the immediate situation is of great importance to an agent which must function in an unpredictable world [3].

In order to have a better idea of the topic of this research, we should imagine someone playing racquetball as illustrated in Fig. 1. As the player visually perceives the movement of the ball, he must be able to make a prediction of the future position of the target and how to approach it to successfully intercept.



Figure 1. A racquetball player [4].

Fast motions are utilized in several areas of robotics for a variety of tasks that moving object interception can be considered as only one of them. A significant amount of work has been devoted to the autonomous control of fast movements such

as catching [5-13], hitting flying objects [14,15], and juggling [16-18]. One famous example of fast motions in robotics is KUKA (Keller und Knappich Augsburg) table tennis player. Its manufacturing company suggests that it is the fastest existing robotic arm in the world. Fig. 2 depicts this robot [19,20].



Figure 2. KUKA robotic arm playing table tennis against Timo Boll [20].

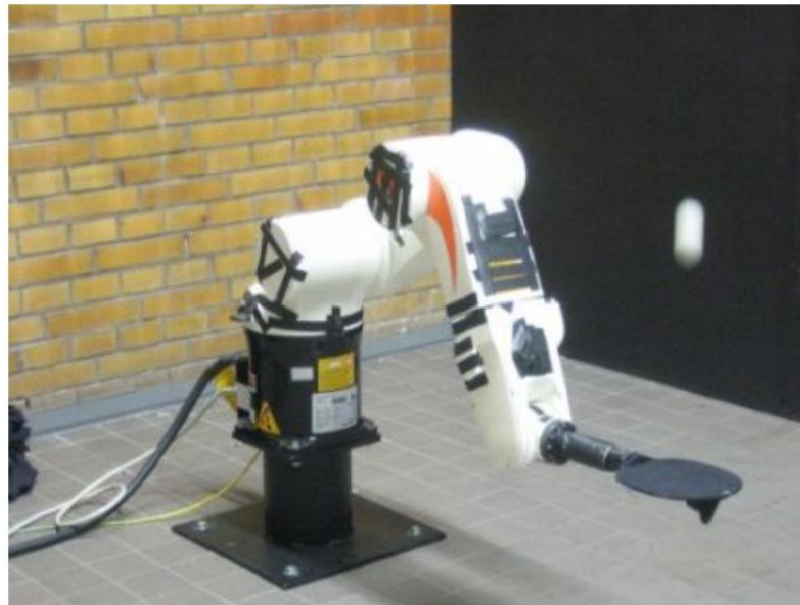


Figure 3. The ball juggling robot [21].

In another project which was completed at Karlsruhe Institute of Technology in Germany, the possibility of real time control of industrial standard robot arms is studied. This has been performed by a ping-pong ball juggling system which is able to catch a ball, thrown by a human operator. It takes advantage of two cameras with 60 HZ frame rates. In this research, they put a great amount of emphasis on the image processing of the fast moving object and comparing the widespread linear model with a novel physically correct model of the spatial trajectory of the moving object. The system can be observed in Fig. 3 [21].

A project which its functionality is closely related to our system has been designed by Swiss Space Center at Learning Algorithms and System Laboratory (LASA). The arm itself is a product of KUKA company and the method gives the equipment the ability to catch projectiles of various irregular shapes in less than five hundredths of seconds. LASA claims that it is a unique system and is set to be used in space.

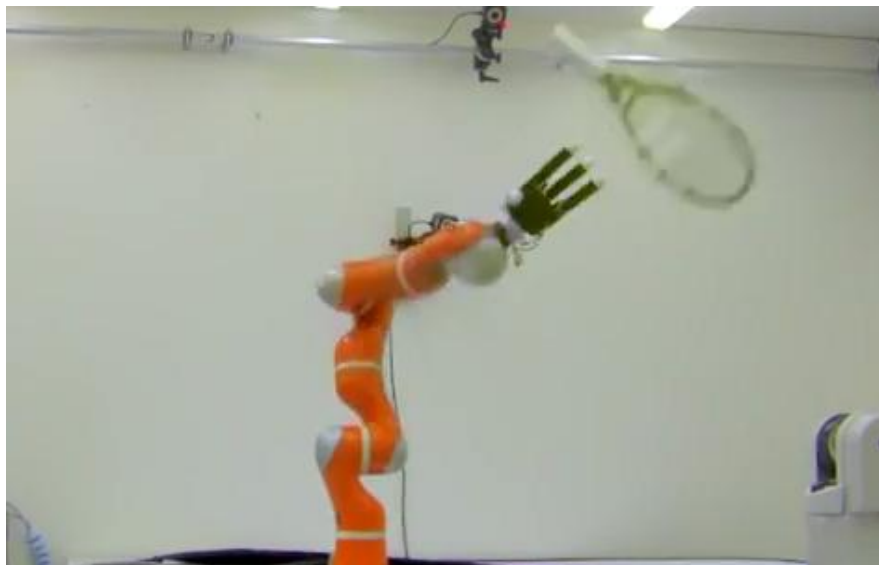


Figure 4. EPFL arm while catching a tennis racquet which has an irregular shape [22].

Fitted on a satellite, the robot would have the task of catching flying debris, whose dynamics are only partially known. However, the robot will not be able to work out such dynamics with precision until in space and observing the movement of approaching objects [22]. As a solution, the researchers employ the method of programming by demonstration which is the manual movement of the robot and teaching the possible trajectories. Then once the robot is in action, its stereo camera system will refine the movement [22]. Fig. 4 depicts a picture of this robot while catching a tennis racquet. In this figure one of the cameras can be seen as well [23].

There are robotic systems which can juggle multiple balls. Fig. 5 illustrates one of these systems which is a DARPA production. In this case, the robot is a multi-fingered hand-arm. The image processing is executed at 500 fps using a high-speed vision system and graphics processing unit [24,25].

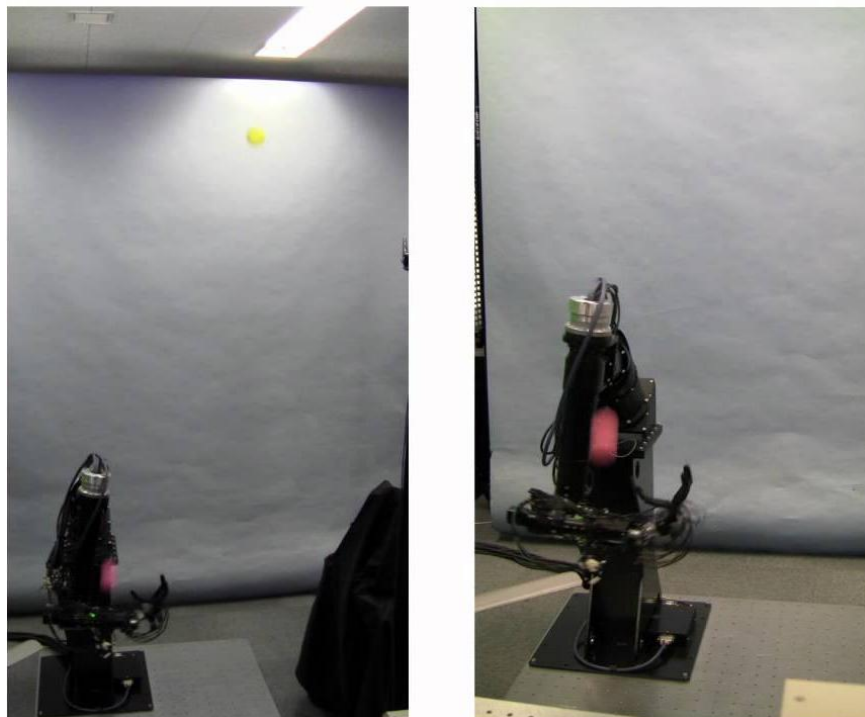


Figure 5. Multifingered robotic arm juggles with two balls [25].

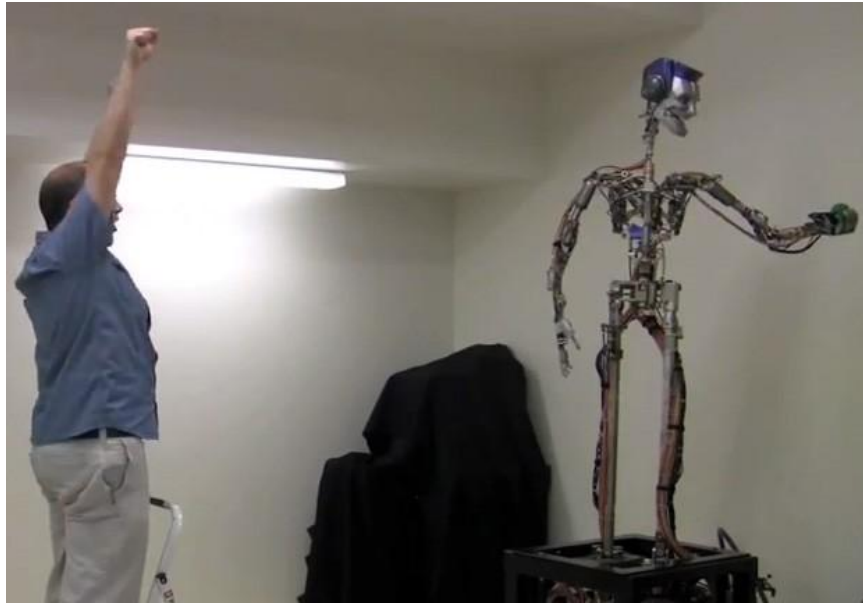


Figure 6. Humanoid robot while throwing and catching a ball [26].

In [26] a humanoid robot is presented which is able to play throw and catch with a human operator. This robot has been presented at Disney research center. It takes advantage of an external camera system to locate the ball and Kalman filter to predict the ball destination and timing. Fig. 6 shows this robot while playing with a human operator.

If we concentrate our review solely on the topic of robotic grasping of moving objects, a variety of works can be found in areas such as visually guided object grasping, hand-eye coordination, visual servoing of robots and vision-based control of robot manipulators [27-31]. We will discuss each of these works, their methods, limitations and how our system can be compared with them.

The majority of works have utilized stereo camera systems and their robotic arms have been built for the sole purpose of high acceleration motions. One of the earliest works in this area was performed by Allen, Timcenko and Yoshimi in [30].

This work focused on tracking and grasping a moving object. They took advantage of two stereo cameras to measure the position of the object. The tracking was performed with the help of Kalman filter. However, the object was travelling on a predefined course such as a toy train's circular path. Every time the object completes the trajectory, the system becomes more robust in performing the task.

In another work, a visual tracking and grasping system is proposed. Its vision system takes advantage of color detection and its control model performs three dimensional linear trajectory prediction to successfully grasp the object [31]. The moving object has a circular motion on the base table of the robot and every time it completes the trajectory, its neural network of the control system becomes more robust. As the trajectory is constrained, the task becomes easier to be performed. This system has the same limitation of the previous system. The major difference between these two is the use of Kalman filter and neural networks for object tracking.

In [32] the task of thrown ball catching is pursued by a multi-purpose 7-DOF lightweight arm and a sophisticated 12-DOF four-fingered hand. In this work the ball is again detected by a stereo camera system. The work focuses on the use of extended Kalman filter as a mean to follow the path of the ball and subsequently manipulates the arm. The main limitation of their system is that if the ball's speed exceeds a certain limit, the system will be unable to detect the presence of the object and consequently, the target will be lost. On the other hand, if the system is unable to detect the object at the beginning but it finds it after the speed decreases, it may still fail to perform the task, as the ball may have already passed the working volume boundaries.

In [33], the concentration is on a similar task. A robot is built specifically for the purpose of ball catching. The arm has exceptionally high dynamics with joint velocities up to 470 deg/s and power consumption of up to 5 KW. But the main focus of this work is on teleoperated ball catching procedure rather than the design of an autonomous system. The kinematic abilities of the robot makes the task of moving object interception considerably easier. And the system may not preserve the expected performance if implemented on an unspecialized equipment such as ours.

In all the related works which were presented in [34-36] the mechanism of catching a ball, the best control method for catching a moving object with the least amount of jerk or catching a flying object in the space have been discussed. However in all of which, the works have been based on computer simulations. Accordingly, it has not been proved if the methods will be able to preserve the expected quality once implemented on a real system.

In one of the recent and most similar works, Lippiello and Ruggiero propose a ball interception system with Comau Smart-Six 7au robotic arm [37]. They apply an eye-in-hand camera to detect the presence of the ball in the working volume of the robot. Their algorithm employs an extended Kalman filter to perform iterative trajectory refinement. In addition, the control system ensures that the ball is always in the field of view until the end effector successfully intercepts it. The main limitation of the system is that the ball needs to be continuously tracked until the moment of interception. If, for any reason, the target is lost, the system fails [37].

There have been only two recent and related works by Neubert and Ferrier [38] which implements a direct mapping of visual inputs to motor torques with the

help of neural networks and Johnson and Neubert [39] which studies robot motion with the help of human-inspired motor programs. The latter paper has a concentration on maximum absolute jerk reduction.

The present work proposes intercepting a moving object without a specialized equipment such as high speed motors. It employs a RGBD camera to obtain RGB and depth information. They are used to detect the position and velocity of the object. To intercept, a control model has been implemented to perform conversion of Cartesian coordinate parameters, (x, y, z) , to joint angles, $(\theta_1 \theta_2 \theta_3)$. At the same time, I have combined a number of different methods to perform object detection, tracking, position extraction and most importantly, velocity estimation along x and y axes in the camera coordinate frame through motion blur in a single image.

Thesis Overview

Chapter 2 will introduce and fully discuss the vision system for object detection and tracking, velocity estimation and the control model. Chapter 3 describes the socket communication, memory sharing, overall setup and the system architecture. Chapter 4 presents the experiments and the results. In addition, a thorough discussion of the results, the benefits of the method, the errors and the difficulties of the experiment will be provided. The last chapter will conclude the work and present future possible research in this area. In this work, the notation of c on the top right side of a parameter stands for camera and represents the camera coordinate frame such as x_o^c . The o stands for the center of the object. Similarly, the notations of w and i stand for world and image as in world and image coordinate frames.

CHAPTER II

THEORY AND METHOD

As explained in the previous chapter, this thesis describes a method for moving object detection and velocity estimation in a single image frame with the goal of faster interception. In this chapter, the theory and complete process for detection to interception will be described in detail. It consists of three main sections, explaining the position determination, velocity estimation and the control model of the method.

In the first step, the system must be able to detect the object of interest. Once the object is detected, the corresponding initial frame will be used to estimate the position and velocity along x and y axes in the camera coordinate system for the center of the object. The obtained position vector is initially in image coordinate frame. So they must be transformed into camera coordinate and subsequently world coordinate system. To approach the problem, it is necessary to obtain the focal length, principal point, skew coefficient and the image distortion of the camera. Consequently, the camera must be calibrated.

Once the position in the camera coordinate system is obtained, the system must determine the object's velocity as well. To estimate the velocity of the object, the system employs the method of speed estimation from motion blur. From the initial frame, the system will be able to determine v_x^c and v_y^c which are the velocity vectors

in the camera coordinate system. As the depth frame cannot record the motion blur, v_z^c cannot be estimated from the initial image frame. So the initial value for v_z^c is unknown and the robot starts its motion after receiving the initial image frame.

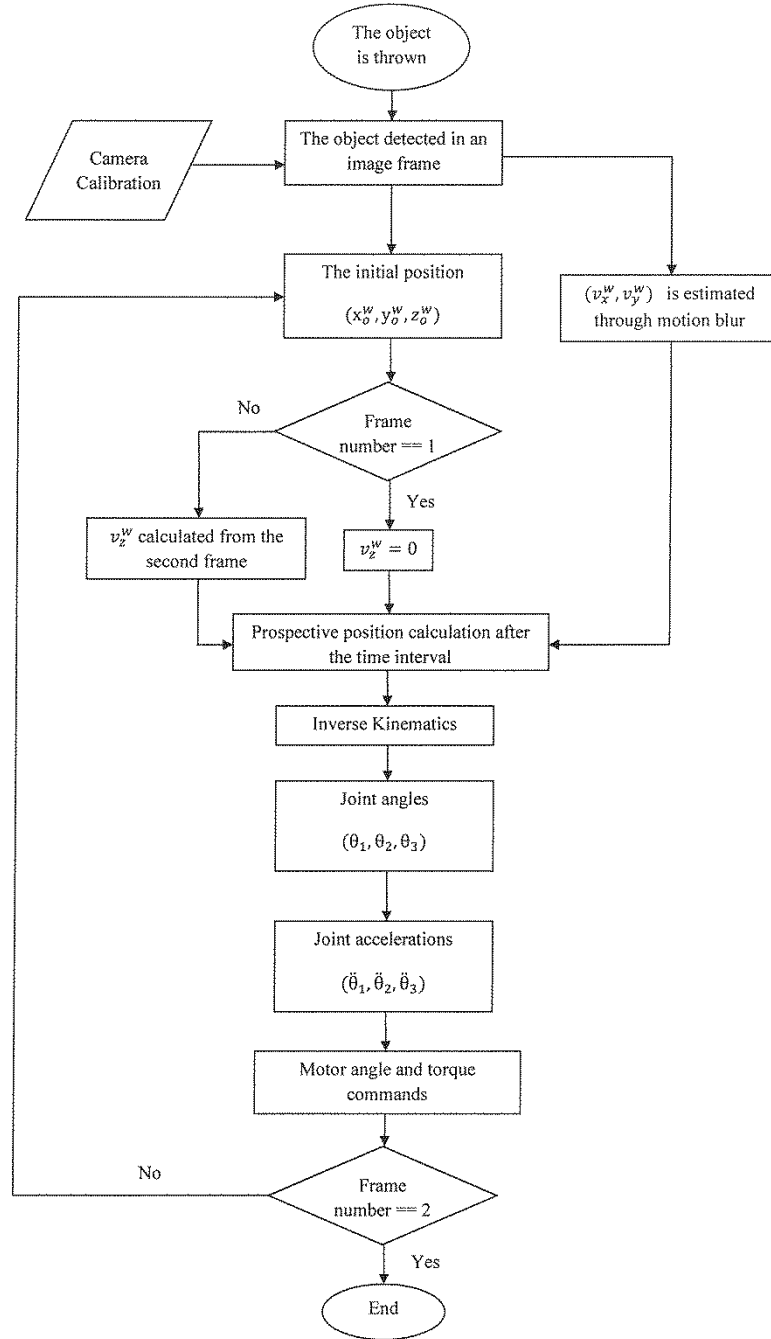


Figure 7. The complete chart of the process

Once the second depth frame is acquired, v_z^c will be estimated, the prospective position of the object will be recalculated and the arm's motion will be accordingly modified. Consequently, the arm does not have to wait for a second image frame to starts its motion. It must be noted that the aforementioned v_x^c , v_y^c and v_z^c are present in the camera coordinate frame. To employ them in the world coordinate which is set at the base of the robotic arm, we must consider the transformation between the camera and world coordinate frames. In this project, because of the specific relation between the coordinate systems, the v_z^c of the camera becomes the v_y^w of the world. Consequently, the motion blur from the initial image frame will be able to estimate v_x^w and v_z^w vectors of the moving object in the world coordinate frame. If we change the location of the camera, this relationship needs to be modified as well. In the proceeding chapters, this will be explained in detail to resolve any type of possible confusion.

To successfully intercept with the moving object, the end-effector must reach the predicted point within the predefined time interval. The predicted position of the moving object is determined in the world coordinate system. To calculate the corresponding joint angles, inverse kinematic methods must be employed.

According to the joint angles and the time interval for reaction, the angular accelerations will be calculated as well. At the end, based on the joint angles and accelerations, the motor angles and torque commands will be determined. The schematic presentation of the process from the moment of object detection until interception can be seen in Fig. 7.

The predefined time interval for the prospective position of the moving object is set according to the kinematic limitations of the equipment and the existing lag in the communication between the computers for programming the camera and controlling the arm. In the following sections, each of the steps of the procedure will be explained in detail.

Object Detection and Tracking

As explained earlier, the first step is to detect and track the object of interest. This step will provide the position of the target. To find the correspondence between the position in the image frame and the position in the camera frame, it is essential to obtain the focal length, principal point, skew coefficient and the image distortion matrix of the device. For obtaining the aforementioned parameters the camera must be calibrated.

Camera Calibration

Camera calibration is a necessary procedure in 3D computer vision for extracting metric information from 2D images which are called the intrinsic and extrinsic parameters.

The parameters which encompass the intrinsic are focal length, principal point, skew coefficient and lens distortion [40]. The focal length and principal point are presented by 2×1 vectors as

$$f_c = [f_x, f_y] \tag{1}$$

$$c_c = [c_x, c_y] \tag{2}$$

Once the central pixel of the object is detected, we obtain the following vector

$$pos = (u_{o,0}, v_{o,0}, z_{o,0}^c) \quad (3)$$

in which (u, v) is the object center's position in the image plane which can be shown by $x_{o,0}^i$ and $y_{o,0}^i$. Then $z_{o,0}^c$ is the depth value and the only parameter which is already in camera coordinate system. To proceed with the image to camera coordinate transformation, we must find x_i and y_i as

$$x_i = \frac{u - c_x}{f_x} \quad (4)$$

and

$$y_i = \frac{v - c_y}{f_y} \quad (5)$$

which are the coordinate values of the object's center in the camera coordinate frame on $z = 1$ plane. The point on z_n plane is determined as

$$x_{o,0}^c = x_i * z_{o,0}^c \quad (6)$$

and

$$y_{o,0}^c = y_i * z_{o,0}^c \quad (7)$$

Then the complete vector of position will become

$$Pos = [x_{o,0}^c, y_{o,0}^c, z_{o,0}^c] \quad (8)$$

The camera's distortion is ignored as its impact on the results were minimal. The reason is the short distance between the camera and the target. The intrinsic camera

parameters were obtained using MATLAB camera calibration toolbar [41] which is based on [42,43].

Object Detection

After calibrating the device, it will become possible to proceed with estimating the position of the object in camera coordinate system. As the object is moving, the image will be blurred. Consequently, the system must be able to recognize the blur area from the actual object.

Motion blur is caused by rapid movement of objects in a still image or a sequence of images such as movie or animation. It can be a result of either rapid movement or long exposure. When a camera creates an image, it does not represent an instance of time, but the scene over a period of time. Consequently, when there is movement within that period, which is referred as exposure time, the result is motion blur. In Fig. 8 an animated blurred moving ball is presented. Cameras with shorter exposure times are less susceptible to motion blur.

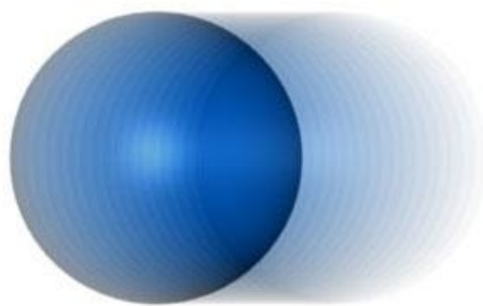


Figure 8. The graphical image blur of a moving ball [44].

As objects in a scene move, an image of the scene is supposed to represent an integration of all the positions of the moving objects. This representation will be according to the camera's viewpoint and the exposure time. The moving object(s) will seem blurred or smeared along the direction of the relative movement. Such a graphical artifact will look natural in a moving picture like an animation. However, this would not be the case in a single image frame.

The initial step in using the motion blur for velocity estimation is to identify the blur in an image. The purpose of blur identification is to estimate the parameters of the blur which are the starting position and the length of the partial blur along the direction of the motion. Some commonly used techniques such as the periodic zero patterns in the frequency domain [45,46], must deal with the motion of the camera. But in this project, the camera is stationary.

We are employing a RGBD camera and we can acquire two image frames, one depth and one RGB. The motion blur can only be recorded in the RGB image frame. Because of the blur, the intensity profile changes smoothly and ends at the edge of the blur or the object. Intensity profile is the intensity of the pixels along a certain direction or path in the image. To receive accurate results, the background must be clean. If the pixel intensity of the background objects mix with the target or its blurred area, there is considerable possibility that the method fails to determine the correct edge of the blur.

Once the object is detected in the depth image frame, it is fitted inside a rectangle. The position of the center of that rectangle is taken as the position of the

object. In addition, the detected area in the depth image frame provides the possibility to determine the edge of the target in the RGB image frame as well.

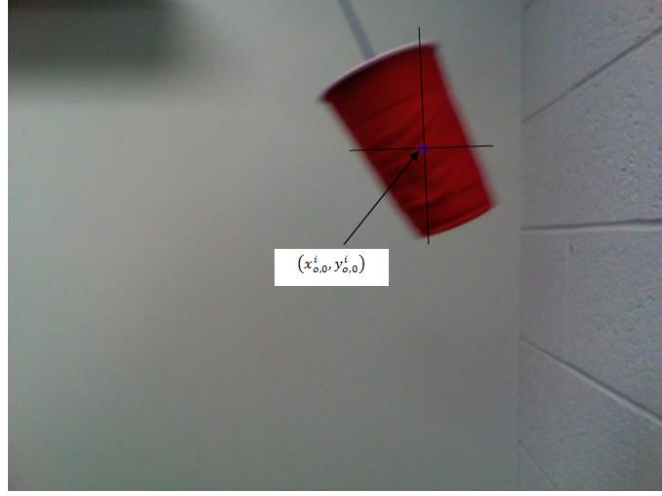


Figure 9. The real world target's RGB representation with its approximated center.

Fig. 9 depicts an actual object with the detected center. To proceed, the algorithm must be able to determine:

- The direction of the motion
- The area with the longest blur

Based on [47], if we consider the motion of a moving object on the horizontal or vertical directions in the image frame, there will be motion blur on the right and left or top and bottom sides of the object. Once the object is detected, the system does not have any idea which area contains the blur. To simplify the process, based on the orientation of the fitted rectangle around the object, we can imagine the object being rotated as much as β to approximately align the object's edges (the fitted rectangle) with the image coordinate frame. To determine where the blur exists, the intensity profile is measured along the vertical and horizontal directions, passing through the

center of the object. Fig. 10 depicts the rotated sample object along with both of the intensity profiles. The intensity profiles are measured along the depicted perpendicular blue lines. If the comparison between the length of the blur to the distance between the center of the object to the object's edge is smaller than a certain threshold, it will not be considered as blurred area. In this work, this threshold is set to 5%.

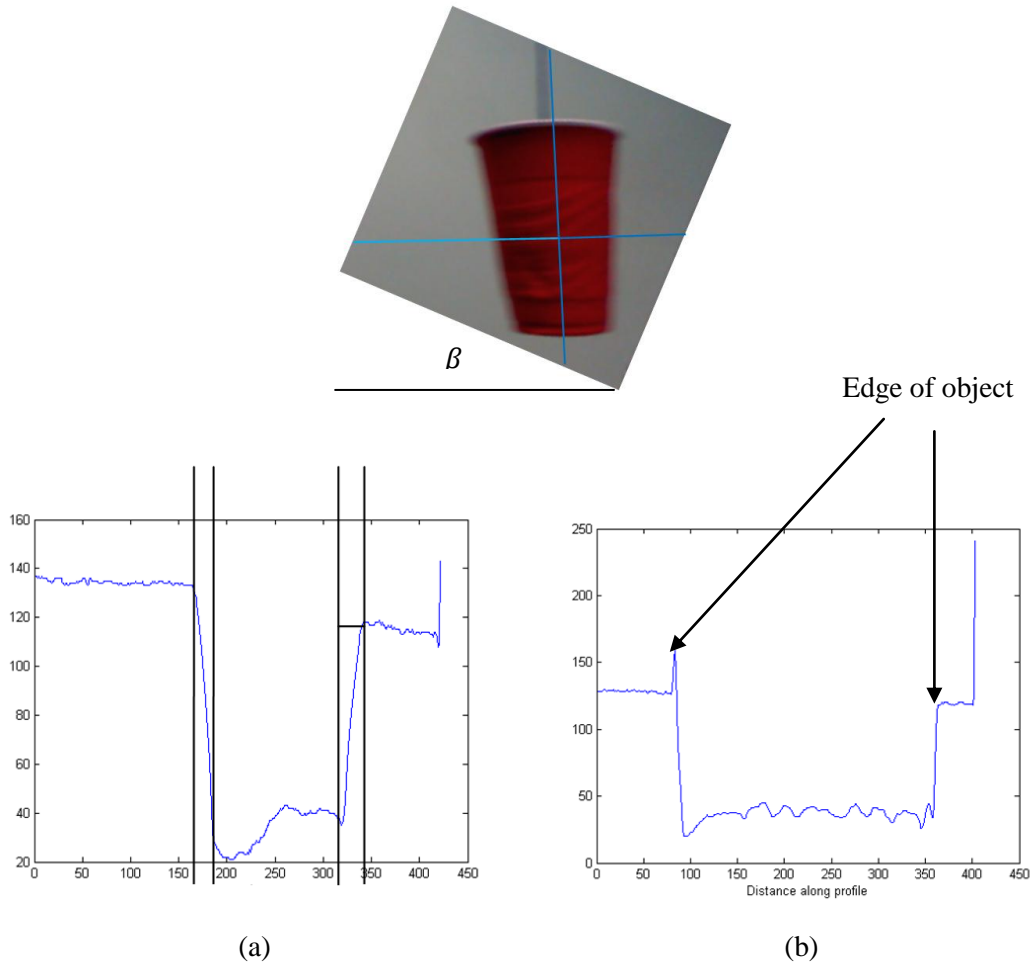


Figure 10. The rotated object along with the horizontal and vertical intensity profiles. a) The intensity profile along the horizontal line. b) The intensity profile along the vertical line.

In Fig. 10, the top image depicts the actual object along with the directions of the intensity profiles. β shows the angle of rotation. In the bottom-right side of the

figure, the intensity along the vertical line is shown. As it can be seen, the profile suddenly changes when the edge of the object starts and ends. There is no considerable smooth variation to be taken as blur. The left image shows the intensity profile of the horizontal direction. In this profile, blur is recorded on both sides of the object. The edges of object and blur are shown with the vertical lines. Because of the blur's presence, the horizontal path will be considered as the direction of motion in the rotated image. By having β , the direction of motion in the original image can be found as well. The rotation formula is under affine transformations. However, the algorithm still needs to determine whether the motion is from left to right or the opposite.

According to [47], although the blur seems symmetrical on both sides of the object, the widths of the intensity profile prove differently. By paying attention to the offset between the pairs of vertical lines in Fig.10, it can be seen that the distance between the pairs on the right side of the object is longer. Consequently, this side is selected as the side for blur parameter determination. In addition, it means that the object is moving from left to right in the image frame.

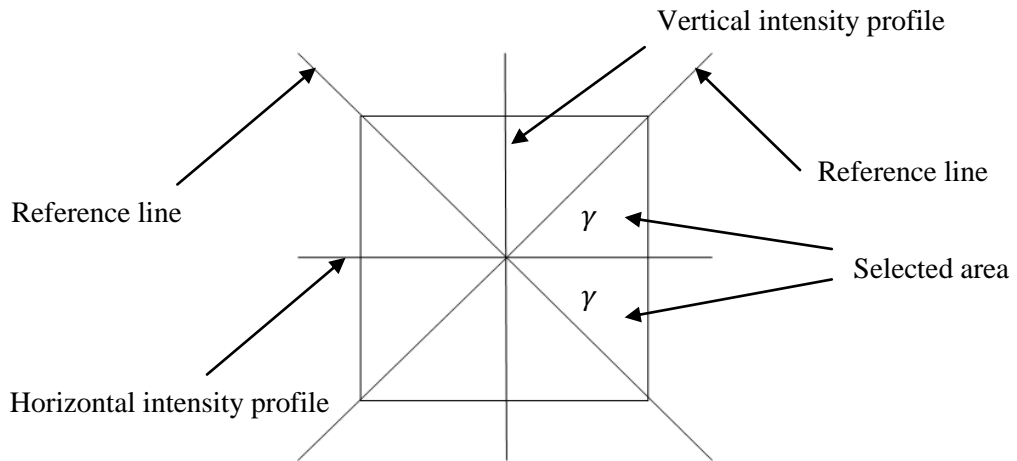


Figure 11. The fitted rectangle and the selected area to determine the blur parameters.

Fig. 11 shows a presentation of the fitted rectangle around the object after rotation. The perpendicular vertical and horizontal lines are the ones along which the intensity profiles are measured. Then based on the side which has the longest width of blur, an approximate symmetrical area, bounded with the laterals of the rectangle is chosen. This area is approximately $\frac{1}{4}$ of the whole rectangle. In Fig. 11, this area is shown with two γ angles and the rectangle's laterals are the reference lines. At this point, the algorithm chooses n directions with random angles, falling inside this area. Along these directions, the intensity profiles are measured again. This time the intensity profiles provide the algorithm with the points on the edge of the blurred area along the directions. The distances between the center of the object and its edge, and the center of the object and its blurred edge will be calculated along every direction. Each distance is determined by

$$d_i = \sqrt{(x_{o,0}^i - x_i)^2 + (y_{o,0}^i - y_i)^2} \quad (9)$$

and

$$d_j = \sqrt{(x_{o,0}^i - x_j)^2 + (y_{o,0}^i - y_j)^2} \quad (10)$$

in which $(x_{o,0}^i, y_{o,0}^i)$ is the position of the center, (x_i, y_i) is a point on the blur's edge and (x_j, y_j) is the point on the object's edge.

These new distances are averaged as well. The difference between the averaged distances will approximate the length of the partial blur, D , as

$$D = \frac{\sum_{i=1}^n d_i - \sum_{j=1}^n d_j}{n} \quad (11)$$

By choosing more distances the algorithm becomes more robust but at the same time it will increase the computational cost of the algorithm. Fig. 12 depicts the animated blurred ball along with a sample intensity profile. The distances, as described above, are shown in this figure as well. The horizontal black line is the reference of the random angles and α_i is a random angle.

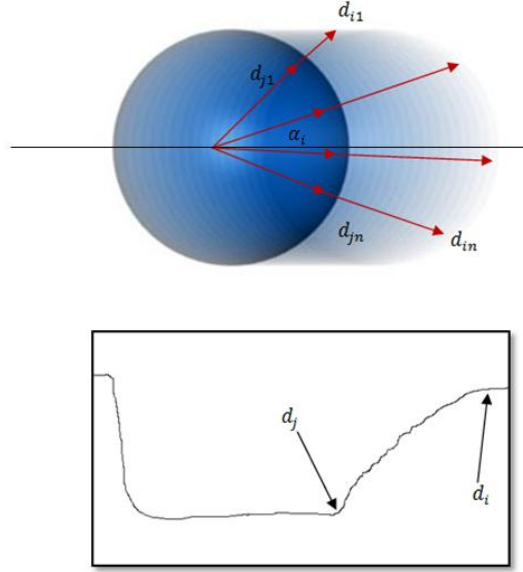


Figure 12. The calculation of motion blur parameters and the intensity profile sample.



Figure 13. The calculated distances on the object.

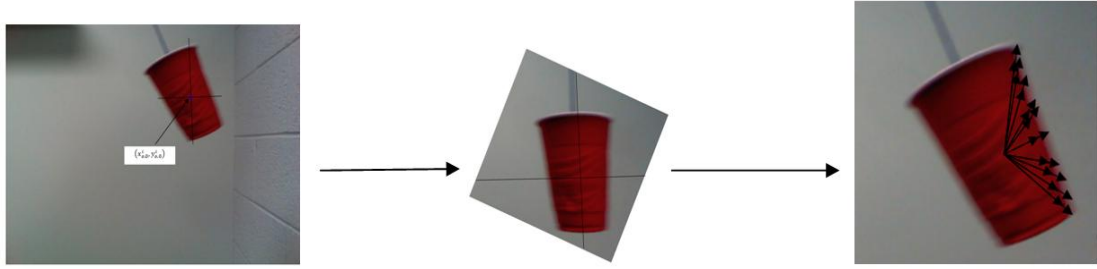


Figure 14. The step by step process on a sample picture.

Fig. 13 depicts the sample object with the calculated distances. As it can be seen, all the distances are randomly selected based on the center of the object. Fig. 14 presents the complete process for the sample in Fig. 9 until the end. Fig. 15 shows the complete process of the explained procedure under this section. Table (1) provides the position of the points on the object's edge. The average distance of the blur for this set of data is 14.4814.

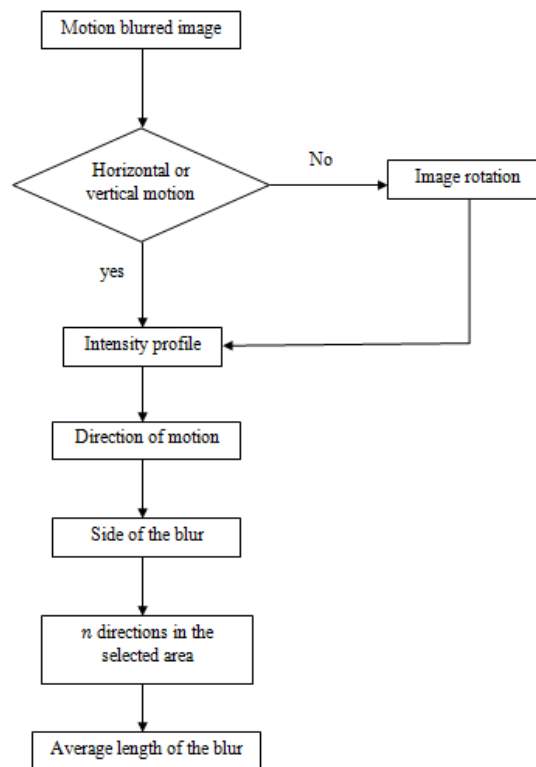


Figure 15. The system flowchart for the blurred parameters detection procedure.

Table 1. Actual position of the points on the edges of object and blur in the sample picture.

x_j^i	y_j^i	x_i^i	x_i^i	D
412.2391	57.4565	414.8478	43.5435	14.1270
413.9783	70.5000	417.4565	55.7174	15.1539
420.0652	84.4130	424.4130	73.1087	12.0793
425.2826	106.1522	432.2391	95.7174	12.5249
427.0217	111.3696	436.5870	100.9348	14.0856
433.9783	131.3696	447.0217	125.2826	14.3936
443.5435	153.1087	457.4565	154.8478	14.0125
449.6304	163.5435	464.4130	168.7609	15.6758
453.9783	178.3261	470.5000	187.0217	18.6371
460.0652	192.2391	470.5000	201.8043	14.1241

According to [47], this method is proposed for highway vehicle speed estimation. So the camera is able to estimate the speed at a certain distance from the moving vehicle. However, as in this work, it is implemented on a RGBD camera, the distance between the camera and the object is always known. In addition, it must be noted that it is more likely to have an error, if the object is spherical. The method heavily relies on the orientation of the fitted rectangle around the object. When the object is a sphere, the rectangle's orientation may not correctly reflect the motion direction of the object and consequently, the correct direction of motion will not be found.

Velocity Estimation

In the previous section, we explained the step-by-step procedure to obtain the position of the object in camera coordinate system as $(x_{o,0}^c, y_{o,0}^c, z_{o,0}^c)$. The next step is to estimate the velocity of the object. From the initial image frame and based on the motion blur, we will obtain (v_x^c, v_y^c) . First v_z^c is set to zero and the robot starts its motion. Once the second depth frame is obtained, the actual value of v_z^c will be estimated. Subsequently, the prospective position of the object will be recalculated and the robot's motion will be modified. In the coming sections we will first explain how the whole blurred area is calculated and then how the blur parameters will be used to estimate the velocity.

Velocity Estimation Formulation

Most of the video-based speed estimation methods use reference information such as the travelled distance across the image frame and then estimate the speed according to the inter-frame time. Nevertheless, because of the limited imaging frame rate, the video camera has to be installed far away from the device in order to avoid motion blur.

The benefit of using this method in this research is that the speed can be estimated in a single image frame. For any fixed time interval, the displacement of the object is proportional to the length of blur in the image.

Speed estimation from motion blur was proposed by Lin et al in [47] for the purpose of vehicle speed detection for law enforcement. In the original work, the method is implemented on a passive system. However, we have implemented the

theory for a real time application. According to the imaging procedure, image degradation caused by motion blur can be categorized as either a spatially invariant or a spatially variant distortion. The latter one corresponds to the cases in which the image degradation model does not rely on the position in the image. Such kind of motion blur image is usually a result of movement during the imaging process [47]. This blur mostly appears in the images which contain moving objects and recorded by a static camera.

The speed estimation is performed on the primary image frame in which the moving object is detected. The pinhole camera model can be seen in Fig. 17. Let's take θ as the angle between the direction of the moving object and the image plane of the camera. And d is the displacement of the object in the frame over T which is the camera's exposure time. The exposure time of our device is 0.0167 sec. To determine θ , we must use the distance between the center of the object and its edge as in Fig. 16.

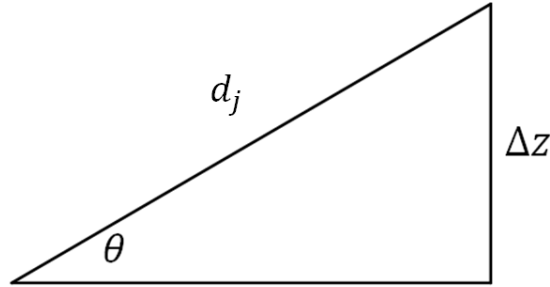


Figure 16. Determining the angle between the object's motion direction and camera plane.

According to the previous section, d_j is the distance between $(x_{o,0}^i, y_{o,0}^i)$ and (x_j, y_j) in the image frame. It is transformed to the camera coordinate frame based on the model explained under camera calibration. In addition, we have the depth of each of

these points as Z_o^c and Z_j^c . Considering the absolute difference between the depth values, we can determine Δz as

$$\Delta z = |Z_o^c - Z_j^c| \quad (12)$$

and θ as

$$\theta = \arcsin\left(\frac{\Delta z}{d_j^c}\right) \quad (13)$$

which can be seen in Fig. 16. Then we have

$$\frac{b}{d_j + D} = \frac{d \sin \theta}{f} \quad (14)$$

and

$$\frac{d \cos \theta - b}{D} = \frac{z_o^c}{f} \quad (15)$$

where d_j and D are the distance between the center of object to its edge and the length of the partial blur area on the image plane. By substituting (14) into (15), and eliminating b , we obtain

$$d = \frac{z_o^c D s_x}{f \cos \theta - (d_j + D) \sin \theta} \quad (16)$$

in which z_o^c is the distance between the object and the camera plane and f is the focal length of the camera. If the time interval is represented by T and the CCD pixel size in the horizontal direction is s_x , then the velocity along each coordinate axis can be formulated as

$$v = \frac{d}{T} = \frac{z_0^c D s_x}{T[f \cos \theta - s_x (d_j + D) \sin \theta]} \quad (17)$$

then d_j and D are the distance between the center of object to its edge and the length of the partial blur area (in pixels). The vectors of velocity along x and y axes will be calculated by

$$v_x = v \cos \theta \quad (18)$$

$$v_y = v \sin \theta \quad (19)$$

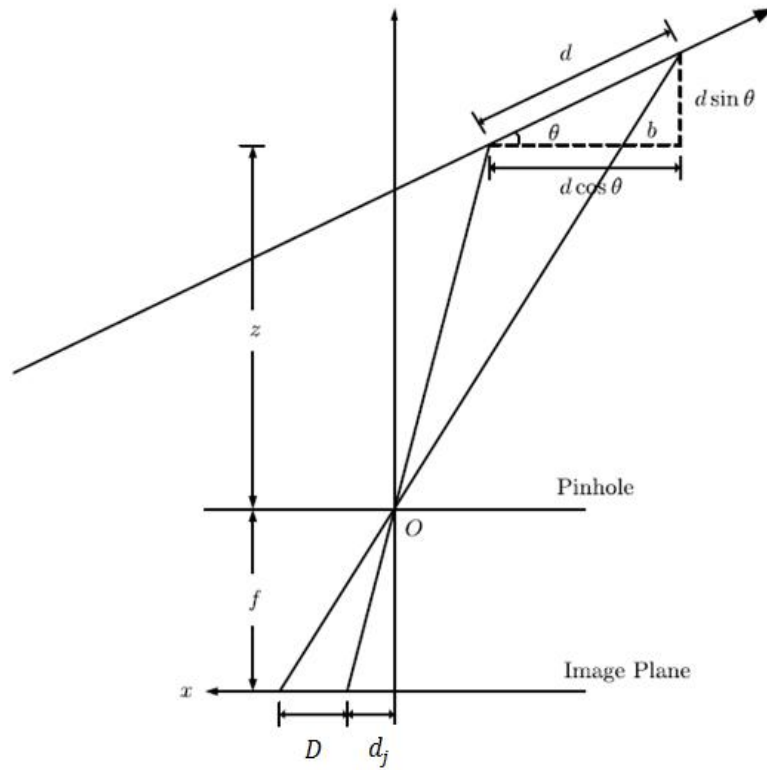


Figure 17. Pinhole camera model for speed estimation.

It should be noted that if the exposure time, focal length, CCD pixel size, starting position, length of the motion blur, and the motion direction are known, then the velocity of the moving object can be derived as

$$v = \frac{z_0^c D s_x}{T f \cos \theta \left[1 - \frac{s_x}{f} (d_j + D) \tan \theta \right]} \quad (20)$$

and the speed can be approximated by

$$v = \frac{z_0^c D s_x}{T f \cos \theta} \quad (21)$$

if $f \gg s_x$ and the angle θ is less than 45° .

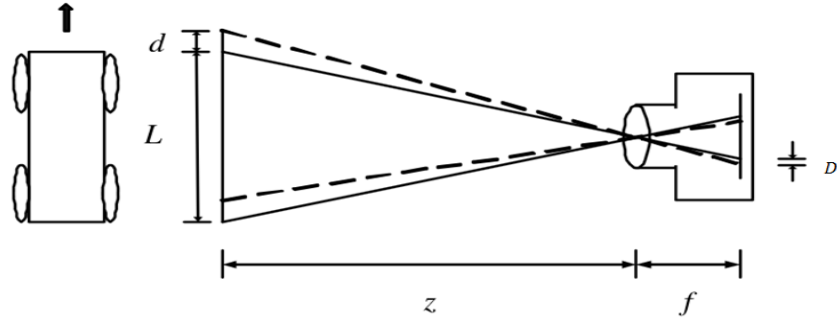


Figure 18. Pinhole model for the special case of parallel moving object [47].

In these equations the focal length f and the CCD pixel size s_x are the internal parameters of the camera. The focal length is obtained from camera calibration and s_x is given by the manufacturer's data sheet. The exposure time (or shutter speed) T is assigned by the camera settings during image acquisition. Thus, for the speed measurement of a moving object using a motion blurred image, the parameters to be identified include the blur parameters and the relative position and orientation between the object and the camera.

For a special case that the object is moving along a certain direction parallel to the image plane of the camera, the displacement of the object can be obtained using similar triangles for a fixed camera exposure time. Fig. 18 depicts this special case.

Accordingly, the equation to obtain the speed can be simplified as

$$v = \frac{z_0^c K s_x}{T_f} \quad (22)$$

in which the angle θ is set to zero. In this case, the position of the object is not required for speed estimation. The only parameter which must be identified from the recorded image is the length of the motion blur.

With the proposed method and its implementation on a camera, it is only possible to estimate (v_x^c, v_y^c) . The arm starts its motion with these velocity vectors and assumes v_z^c to be zero as it is known. After obtaining the second frame and calculating v_z^c , the prospective position of the object will be recalculated. Subsequently, the motion of the arm will be modified. This way the manipulator does not have to wait for the second frame to start its motion. In this project, the angular displacement of the motors are recorded by the encoders. So the displacement between the first and second image frames are known.

Once the second depth frame is acquired, we take advantage of depth frame subtraction to estimate the vector of velocity along z axis in the camera coordinate system. The first frame after the detection of the object provides the initial distance between the object and the camera coordinate system as z_1^c . The next frame would provide z_2^c . If the time interval between the two frames is T , then v_z^c can be approximated by

$$v_z^c = \frac{z_2^c - z_1^c}{T} \quad (23)$$

Finally, the vectors of position $(x_{o,0}^c, y_{o,0}^c, z_{o,0}^c)$ and velocity (v_x^c, v_y^c, v_z^c) in camera coordinate system are obtained. With these vectors, the future position of the object can be obtained again. Accordingly, the joint angles and accelerations of the robot will be recalculated.

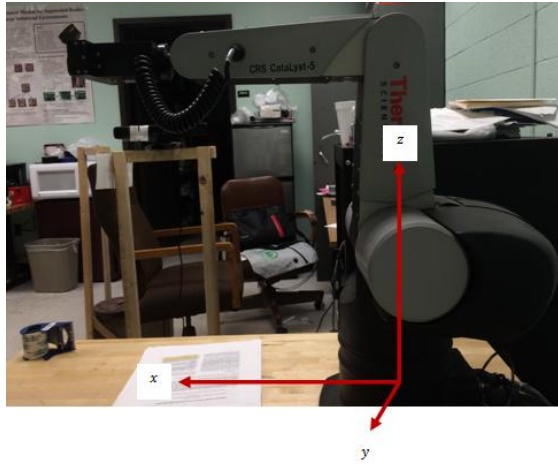


Figure 19. The world coordinate frame of the system on the base of the robotic arm.

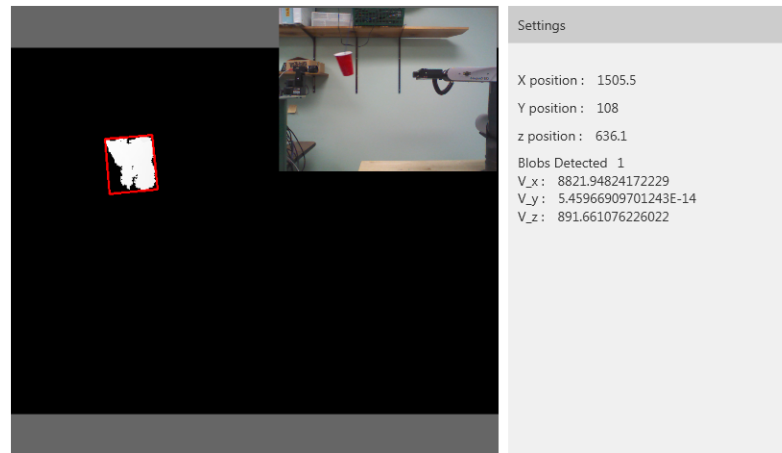


Figure 20. The velocity estimation along with position.

Fig. 19 depicts the coordinate frame of the system (world coordinate frame) and Fig. 20 shows an example of how the position and velocity of the target are estimated. It must be noted that the robot and world coordinate frames are the same

and these terms are used interchangeably. The provided data are in mm and according to the world frame (robot's coordinate frame). This figure shows the graphical display of the position and velocity parameters of the moving object along with the RGB picture on the top right side of the figure and the depth image as well. As it can be seen in the RGB image, the moving object is blurred and its position and velocity vectors are calculated. All the parameters of velocity have positive values. The positive value of y shows that the object is on the left side of the robotic arm. In this presentation, v_y^w is almost zero. It means that the object has minimal displacement on y axis.

Control Model

As discussed in the previous section, the initial position and velocity of the moving object are determined. Consequently, its future position can be predicted after any time interval. This time interval is actually the response time for the arm. In other words, this would be the time that the end-effector has to reach the predicted position of the target.

When the object is thrown, the only considered acceleration is gravity and air resistance is ignored. The position of the target can be predicted in world (robot) coordinate frame by

$$x_{o,1}^w = v_x^w t + x_{o,0}^w \quad (24)$$

and

$$y_{o,1}^w = v_y^w t + y_{o,0}^w \quad (25)$$

and

$$z_{o,2}^w = -\frac{1}{2}gt_1^2 + v_z^w t + z_{o,1}^w \quad (26)$$

where t is the time interval, t_1 is the time interval after the first depth frame, g is the constant acceleration of gravity and $(x_{o,1}^w, y_{o,1}^w, z_{o,1}^w)$ is the predicted position of the target after t . At this point, $(x_{o,1}^w, y_{o,1}^w, z_{o,1}^w)$ must be used to derive the corresponding joint angles for the robot. Based on the time interval, the necessary joint accelerations will be calculated as well. The corresponding joint angles are derived with the help of inverse kinematics.

Inverse kinematics is the application of the kinematic methods and equations to calculate the joint parameters which will provide the desired position of the end-effector. In other words, we will be able to determine the value of each joint in order to place the arm at the desired position and orientation. To explain this procedure, we must practice forward kinematics first. This is based on Denavit-Hartenberg representation of forward kinematics of robots. According to the DH table of the manipulator, we can form the transformation matrix of each joint by

$$T_{n+1}^n = A_{n+1} = Rot(z, \theta_{n+1}) \times Trans(0,0, d_{n+1}) \times Trans(a_{n+1}, 0,0) \times Rot(x, \alpha_{n+1}) \quad (27)$$

in which n is the joint number, starting from 0. By calculating A_{n+1} , the following matrix will be found as

$$A_{n+1} = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (28)$$

in which θ_{n+1} is the angle to rotate around the current corresponding z axis, d_{n+1} is the displacement along the current corresponding z axis, a_{n+1} is the displacement along the current corresponding x axis and α_{n+1} is the angle to rotate around the current corresponding x axis. S and C represent sine and cosine. If the number of joints to be controlled in the manipulator are equal to three, we will be able to find three matrices as (A_1, A_2, A_3) . These matrices can be used to find the forward transformation from the base of the robot as the first joint to the robot's end-effector according to

$${}^R T_H = A_1 A_2 A_3 \quad (29)$$

where ${}^R T_H$ is called the robot's total transformation matrix. This matrix is represented as

$${}^R T_H = \begin{bmatrix} n_x & o_x & a_x & x^w \\ n_y & o_y & a_y & y^w \\ n_z & o_z & a_z & z^w \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (30)$$

in which \bar{n}, \bar{o} , and \bar{a} are the unit vectors of object's orientation in the world coordinate frame. These vectors are named normal, orientation and approach, consecutively.

As we have already noticed, forward kinematic equations have a multitude of coupled angles. It is impossible to discover sufficient number of elements in the matrix to solve for individual sines and cosines to calculate the angles of the joints. To decouple the unknown angles, we will premultiply the ${}^R T_H$ matrix with the individual A_n^{-1} matrices. Fig.21 depicts the inverse kinematics model of our manipulator. This

presentation is called elbow-up as the third joint of the robot is considered the elbow the same as human's arm.

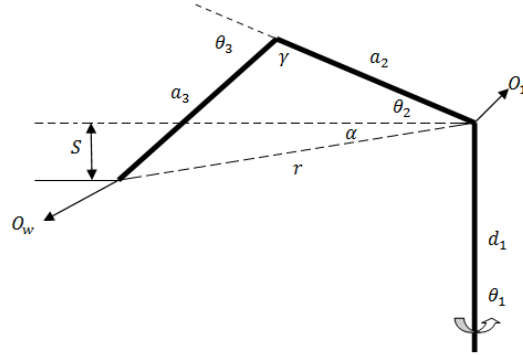


Figure 21. The inverse-kinematics model solution for the manipulator.

This solution is based on the model presented in [48]. However, it may similarly be repeated for other robots with different configurations. Let's assume table (2) represents the DH table of the manipulator.

Table 2. DH tables with parametrical elements.

	θ	d	a	α	Var
1	θ_1	d_1	a_1	α_1	θ_1
2	θ_2	d_2	a_2	α_2	θ_2
3	θ_3	d_3	a_3	α_3	θ_3
4	θ_4	d_4	a_4	α_4	θ_4
5	θ_5	d_5	a_5	α_5	θ_5

We consider the matrix of position in the world coordinate frame to be

$$P = \begin{bmatrix} x^w \\ y^w \\ z^w \end{bmatrix} \quad (31)$$

According to the configuration of the manipulator, shown in Fig. 21, the first angle will be found as

$$\theta_1 = \arctan \left(\frac{y^w}{x^w} \right) \quad (32)$$

If we take A as the tip of the end-effector, it will be found by

$$A = d_5 \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad (33)$$

in which d_5 is taken from table (1). According to (31) and (33), we will be able to find the location of the wrist as

$$O_w = P - A = \begin{bmatrix} OW_1 \\ OW_2 \\ OW_3 \end{bmatrix} \quad (34)$$

In addition, we form another matrix as

$$O_1 = \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \quad (35)$$

which is the location of the second joint. d_1 is taken from table (1) as well. According to (34) and (35), we can find r as

$$r = O_w - O_1 = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \quad (36)$$

Based on (36), we can find the length of r as

$$r = \sqrt{r_1^2 + r_2^2 + r_3^2} \quad (37)$$

At this point, it is possible to find the value of θ_3 angle by

$$\theta_3 = \arccos\left(\frac{r^2 - a_2^2 + a_3^2}{2a_2a_3}\right) \quad (38)$$

where a_2 and a_3 are taken from the DH table.

if we take S as

$$S = |ow_3 - d_1| \quad (39)$$

then

$$\alpha = \arcsin\left(\frac{S}{r}\right) \quad (40)$$

and

$$\beta = \arccos\left(\frac{r}{a_2 + a_3}\right) \quad (41)$$

Accordingly, θ_2 will be found as

$$\theta_2 = \alpha - \beta \quad (42)$$

As explained earlier, for the task of interception the first three joints will manipulate the arm to the corresponding location in the working volume. Consequently, the fourth and fifth joint angles will be kept constant.

Let θ represent the vector of joint angles as

$$\theta = [\theta_1, \theta_2, \theta_3]^T \quad (43)$$

If we assume that the angular displacement can be determined by

$$\Delta\theta = \frac{1}{2}\ddot{\theta}t^2 + \theta_0 t \quad (44)$$

and $\theta_0 = 0$, then the necessary acceleration for each joint can found by

$$\ddot{\theta}_i = \frac{2\Delta\theta}{t^2} \quad (45)$$

where $\ddot{\theta}_i$ represents the acceleration of joint i, $\Delta\theta$ is the angular difference between the initial and final position of the end effector and t is the time interval. This process is only performed for the first three joints of the manipulator.

In this chapter, we presented the complete method and theoretical basis of our system. In the first part of the chapter, we discussed the object detection and camera calibration model. In the second part, we fully explained the velocity estimation method which provides the major contribution of this work. And in the final part, the control model and dynamical analysis of the method were illustrated.

CHAPTER III

EQUIPMENT AND EXPERIMENTAL SETUP

In this chapter we will present the equipments which are used for testing and the setup for the implementation of our method. The kinematic specifications and limitations of the devices will be provided and the Simulink control model will be thoroughly described. In addition, it will contain information about the communication means between the computer devices in our setup.

Experimental Setup

Fig. 22 shows a schematic diagram of the system's architecture. The vision system and all the necessary calculations are processed on a separate windows 7 system. These two machines communicate through UDP socket. The main reason that we preferred UDP over TCP/IP was the delay in the procedure [49].

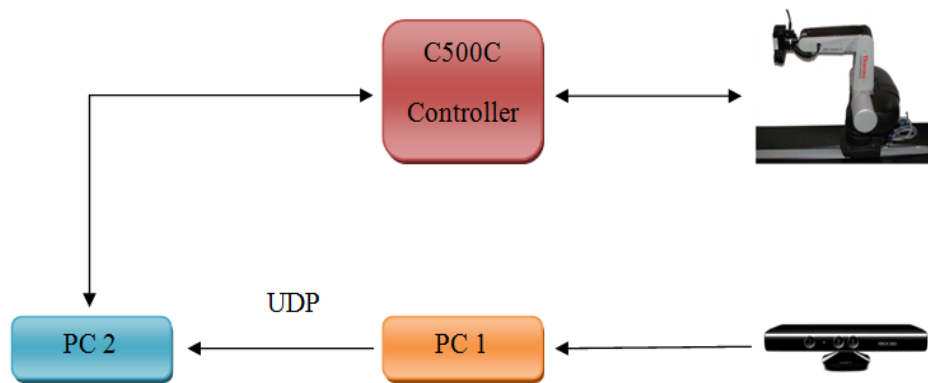


Figure 22. The schematic configuration of the system. The information is transferred from Kinect to PC 2 without feedback.

It is desirable to have the vision system process the data and transfer the processed information to the controller's machine without considering any possible malfunction or blockage on the other side of the communication line.

In the previous chapter, we fully explained how the vectors of position and velocity are obtained. However, both of these vectors are present in the Kinect's coordinate system. These vectors must be transformed to the arm's coordinate system (world frame).

Coordinate Transformation

The calibration is performed according to the first section of second chapter and the provided model is used. The procedure provides the intrinsic parameters of the Kinect as

$$f = [521.509, \quad 520.271] \quad (46)$$

$$CC = [317.675, \quad 234.552] \quad (47)$$

$$\alpha c = [0] \quad (48)$$

$$kc = [0.04810, -0.17419, -0.00076, -0.00098, 0] \quad (49)$$

which f represents the vector of focal length, CC the vector of principal points, αc the skew coefficient and kc is the vector of distortion. All of the provided values are in millimeters.

To calibrate the relationship between the Kinect and the robot, the accurate origin of the camera coordinate system must be obtained. In this project, we found the origin of the Kinect coordinate system by manually manipulating the end-effector in

front of the camera. The end-effector is detected. The position of the end-effector is known in the world coordinate frame. The image processing algorithm provides its position in camera coordinate frame as well. Then the coordinate relation between the end-effector and the camera's position can be determined.



Figure 23. The position of the camera and the robot.

As it can be observed in Fig. 23, the camera is mounted on the right side of the robot at $(0.533\text{m}, -0.812\text{m}, 0.431\text{m})$ in the world coordinate frame. To align the Kinect's coordinate frame with the world's, a rotation of $\pi/2$ around x and a rotation of π around y are needed.

Considering the location of the Kinect and the coordinate origin of the arm, the transformations are executed as

$$P_{\text{arm}} = \text{Rot}_{(x, \pi/2)} \cdot \text{Rot}_{(y, \pi)} \cdot T_{(\Delta_x, \Delta_y, \Delta_z)} \cdot P_{\text{cam}} \quad (50)$$

and

$$V_{\text{arm}} = \text{Rot}_{(x, \pi/2)} \cdot \text{Rot}_{(y, \pi)} \cdot T_{(\Delta_x, \Delta_y, \Delta_z)} \cdot V_{\text{cam}} \quad (51)$$

where \vec{P}_{cam} is the position of the object according to the Kinect's coordinate system and \vec{P}_{arm} is the position of the object according to the arm's coordinate system. The same condition is valid for \vec{V}_{cam} and \vec{V}_{arm} .

In this project, the robot's working volume contains a 3D rectangular space with the dimensions of 0.180m by 0.540m and 0.360m in front of the robot. When the robot is in home position, the end-effector is at the center of this volume. For fast motions, the kinematic limits of the arm can dominate the robot's behavior. To ensure the robot's safety, if any of these limits is exceeded, the controller automatically disconnects the arm.

Equipments

The equipment for performing the object interception is a robotic arm which is depicted in Fig. 24. This equipment is a 5-DOF manipulator named as CRS Catalyst-5, designed and produced by Thermo Science company, which is mainly employed for the purpose of object manipulation, force measurement and analysis. All the joints in this equipment are rotational and the arm has an articulated configuration. In this picture the world coordinate system can be observed at the base of the robot.

Table 3. The joints and their corresponding angular ranges.

Joint number	Range (deg)
1	(−180,180)
2	(0,50)
3	(−90,30)
4	(−45,45)
5	(−75,75)

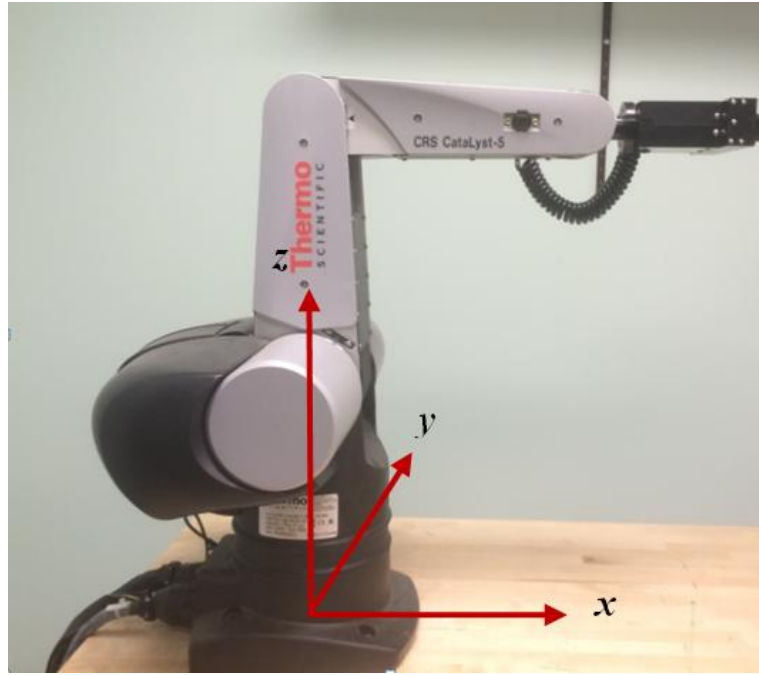


Figure 24. CRS Catalyst 5 with the world coordinate system. The origin of the coordinate system is on the base of the arm.

Table (3) presents the joint limits of the equipment. Table (4) provides the corresponding working volume of the robot in Cartesian coordinate values. And in table (5), we can find the DH table of the equipment.

Table 4. The Cartesian limits.

Axis name	Range (mm)
X	(-90,90)
Y	(-270,270)
Z	(-180,180)

The angular limits in table (3) ensure that the robot will not be damaged over the course of manipulation. It is possible to alter the control algorithm and increase these limits but it can cause severe damage to the arm.

Table 5. The DH table of CRS Catalyst-5.

	θ	d	a	α	Var
1	0	254	0	$\frac{-\pi}{2}$	θ_1
2	$\frac{-\pi}{2}$	0	254	0	θ_2
3	$\frac{\pi}{2}$	0	254	0	θ_3
4	$\frac{-\pi}{2}$	0	0	$\frac{-\pi}{2}$	θ_4
5	0	159	0	0	θ_5

The images were acquired with a Kinect camera. Kinect is a RGBD camera which measures the position of objects of interest in the camera coordinate system. The first generation of this device was designed for Xbox. Then in 2011, Microsoft released a software development kit (SDK) which made it possible to program this device in Windows 7.

This device is a horizontal bar established on a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device contains a RGB camera and a depth sensor.

The depth sensor is made of an infrared laser projector which is combined with a monochrome CMOS sensor. It obtains video data in 3D under any ambient light conditions. The new Microsoft SDK makes it possible to adjust the range of the depth and it can automatically calibrate the sensor based on the physical environment,

accommodating for the presence of furniture or other obstacles. The depth range is between 0.8 to 4.0 meters. This means that the objects which are closer or farther from these distances from the Kinect will not be detected. At the same time, the device has a near mode depth which covers a range of 0.4 to 3.5 meters. Fig. 25 shows a Kinect along with its coordinate system (camera coordinate frame).

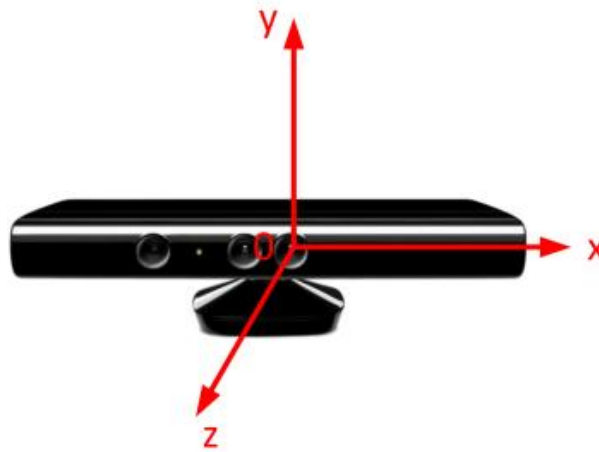


Figure 25. Kinect's coordinate system.

As it is shown in Fig.25, the depth range is determined along the z direction. The figure demonstrates the rough location of the coordinate system but to precisely determine the location of the frames origin and its relative location with respect to the robot frame, the procedure must be completed as explained in the previous section.

The Kinect for windows SDK makes it possible to use and control a Kinect camera in windows by C++/C# compilers. The compiler language which is used for this project is C#. The Kinect API provides the chance of using the kinect's image/video stream similar to a webcam. As discussed earlier, the depth sensor provides the distance of the objects. And it would be possible to leverage the video

and depth data together for detecting objects. The SDK provides the essential libraries to access the Kinect's video and depth stream.

The Kinect and arm are programmed and controlled in separate computer devices. Once the position and velocity vectors of the target are processed and the prospective position of the object is determined, the data is transferred to the other computer device which controls the robot. This data transfer is possible through UDP socket communication. In the proceeding sections more information on this type of communication will be provided. The transferred data arrives at the second computer. The data will be processed in C++ for the robot's Simulink model. Consequently, the new processed data needs to be transferred again. This last procedure is possible through memory sharing. In the next section, the *stream server* block and sharing configuration will be described in detail as well.

Open-Architecture Configuration

As explained earlier, the equipment in this project is a CRS Catalyst-5. It is a 5-DOF manipulator with a C500C controller which has a closed-architecture. Advanced robotic experiments such as coupled control, adaptive control, and other advanced algorithms cannot be implemented on this architecture. However, if the control is set on open-architecture model, the controller would be able to send commands directly to the motors. The designed open-architecture controller can be switched back and forth to the closed-architecture CRS controller seamlessly.

The open-architecture configuration allows us to change the control model. This architecture is designed to make adding, upgrading and swapping components easy [50]. With the help of open-architecture configuration, we make the control

model to have the arm perform the task of interception. Fig. 26 provides the control model for the arm.

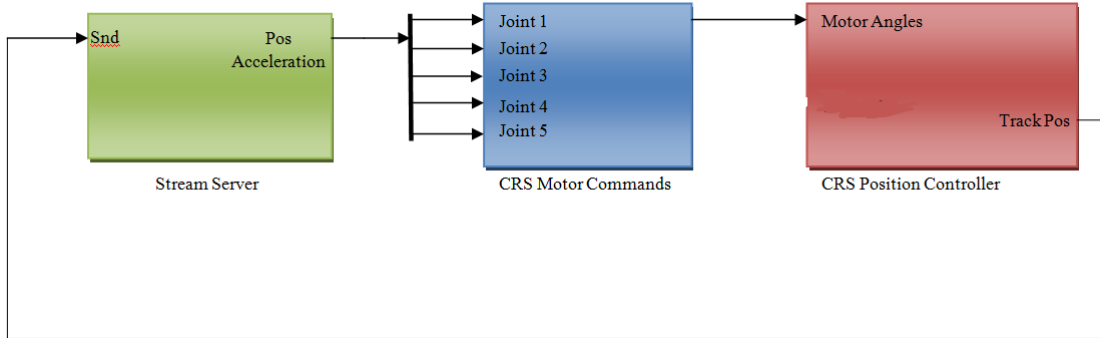


Figure 26. Schematic of the discrete controller inside the real-time kernel.

This figure is a schematic presentation of the control Simulink model. To further investigate how the model is configured and how the system functions, we will explain each of the blocks in detail.

As discussed earlier, inverse kinematics provides the corresponding joint angles for end-effector's manipulation. This process is performed in C++ and the vector of angles and angular accelerations must be used to determine the motor commands. In Fig. 26, the *stream server* block provides the mean for communication between the C++ and Simulink model.

Fig. 27 depicts an actual image of this block in Simulink environment. This block listens for connections from local hosts. When a remote host attempts to connect, it accepts the connection and establishes a persistent connection with that host. Only one connection is accepted at a time. The host may be local or remote. If the connection to the host is lost, this block automatically accepts a new connection. The current state of the connection is available at the *state* output.

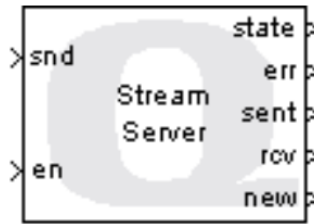


Figure 27. The stream server block in Simulink.

When the block is configured to receive data and data arrives, the **new** output is set to *true* and the data appears at the **rcv** output.

When the block is configured to send data, the signal at the **snd** input is sent to the host each sampling instant in which the **en** input is non-zero. This signal may be multi-dimensional. The **sent** output is set to true (non-zero) if the data was sent. If the send operation would have blocked then the **sent** output is set to false (zero). If an error occurs then a standard negative QuaRC error code is output at the **err** output. An error condition will cause the block to close the connection and attempt to accept a new connection.

The basic stream blocks are always non-blocking. They do not wait for data to be sent or to be received. However, the signals sent or received are always treated as atomic units-the block will never receive part of the data or send part of the data.

The stream block supports a number of different communication options. It may be configured to use non-blocking I/O for the underlying stream or to use blocking I/O in separate threads. In either case, the block itself never waits for data to be sent or received and thus does not interface with the sampling rate of the model. The blocking I/O option requires more knowledge to use but has the advantage that it

reconnects faster when the connection is lost and typically requires less computation time in the control thread.

It may also be configured to maximize throughput or minimize latency. When maximizing throughput, the block buffers the data prior to flushing it to the underlying stream in order to make optimum use of the communications bandwidth available. However, this option can lead to latencies at the peer because the data is not necessarily sent immediately. The size of the buffers used for internally buffering the data are set via the *Send buffer size* and *Receive buffer size* parameters.

When the block is configured to minimize latency, it flushes the data to the underlying stream every sampling instant. This option can result in poor use of communications bandwidth but minimizes the time for the data to arrive at the host. For example, a single TCP/IP packet can store 1460 bytes of data, or 182 doubles. Suppose the *stream server* block is sending and receiving a single double. If the send and receive buffer size are set to 1460 bytes, and the maximize throughput option is selected, then the stream block will send 182 doubles in each TCP/IP packet, making efficient use of the communications bandwidth. However, if the minimize latency option is selected, then the stream server block will send one double in each TCP/IP packet. In this case there is a lot of wasted bandwidth but the latency is minimized.

In most circumstances, the server and client only need to exchange the most recent data and old data may be discarded. The stream server block supports this option as well, which also helps to minimize latency when only the most recent data is required.

As the vectors of joint angles and joint accelerations are transferred through the stream server, they will be used to generate the trajectory of the arm. Fig. 28 depicts the CRS motor commands block details.

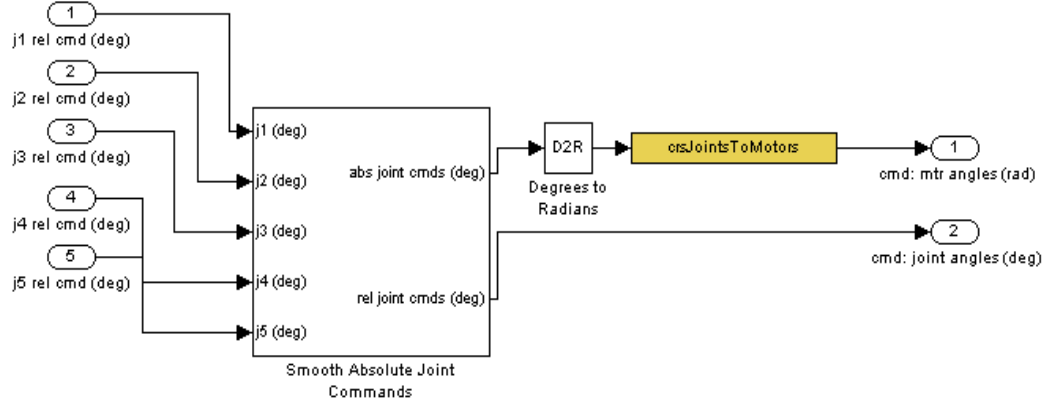


Figure 28. The motor commands block details.

As illustrated, the joint angles are transferred in degrees into the block of *smooth absolute joint commands*. Fig. 29 presents the aforementioned block. This block determines absolute joint commands for each joint according to the home position of the robot. The commands are converted into radians. Subsequently, they will be transferred to be converted to motor commands. The vectors of θ and $\ddot{\theta}$ are transferred to the *continuous sigmoid* functions through the *saturation* blocks. The kinematic limits of the arm are set by the *saturation*. This block limits the input signal to the upper and lower saturation limits. The limitations are the angular position, velocity and acceleration range for each joint. If any of these limits are exceeded, the controller disconnects the robot to prevent motors from being damaged.

Each *continuous sigmoid* block generates a sigmoid trajectory from the current position and acceleration to the target's position. Position and acceleration profiles are

generated, as well as a signal to indicate when the trajectory has reached the target. The output of each *continuous sigmoid* block is the corresponding absolute joint command. The *Abs to Rel Offset* block sets the angular offset for the arm in its home position. It can be seen that there is a 90° offset for the third joint.

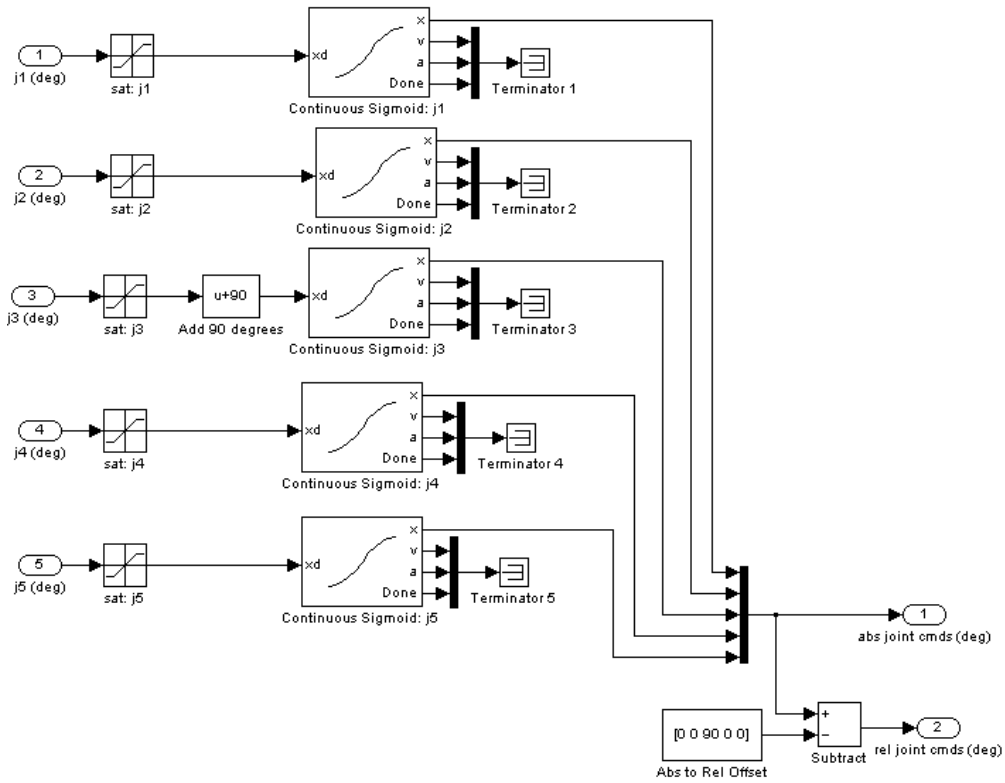


Figure 29. Smooth absolute joint commands block.

The joint commands are transferred to *CRS position controller* block. The model can be seen in Fig. 30. As it can be observed, the vector of joint commands are fed into PD control system and consequently, the motor current limits will be determined. In this picture, the *HIL Read Encoder* block reads the specified encoder channels every time the block is executed. The channels are read immediately. The output is the count values from the encoders. On the other hand the *HIL Write Analog* block writes

the specified analog channels every time the block is executed. The channels are written immediately.

Figure 30. CRS position controller.

In this chapter, we provided detailed information about our equipments, testing setup and network communication in the system. In the proceeding chapter, the testing results and the discussion will be provided.

CHAPTER IV

EXPERIMENT, RESULTS AND DISCUSSION

In this chapter, we provide the testing results and demonstrate the benefits of velocity estimation in the initial image frame. A thorough discussion of the obtained data will be provided. At the end, the errors and limitations of the system will be described.

To study the system's functionality, the setup is tested by throwing a cup inside the working volume of the robot. When the system is armed, the cup is held out of the Kinect's view angle. Then it is thrown within the working volume by an operator.

In this project, we take advantage of the area size that each object has in the depth image. The object must meet a specific area size range to be considered as the target. We have the possibility to accumulate the depth pixels as separate contour areas. As there is a knowledge of the 2D contour area size range of the object of interest, the contours which are bigger or smaller than a certain size will be removed.

The size range in this project is between 1600 mm^2 to 6400 mm^2 . It must be noted that the area size is a function of depth. Consequently, the proposed area size range is valid for our setup. It is determined based on the straight distance between the camera and the arm's working volume. So for a different setup, the area size range

must be modified as well. Fig. 31 shows the environment with the objects. The RGB picture can be viewed on the top right side of the figure.

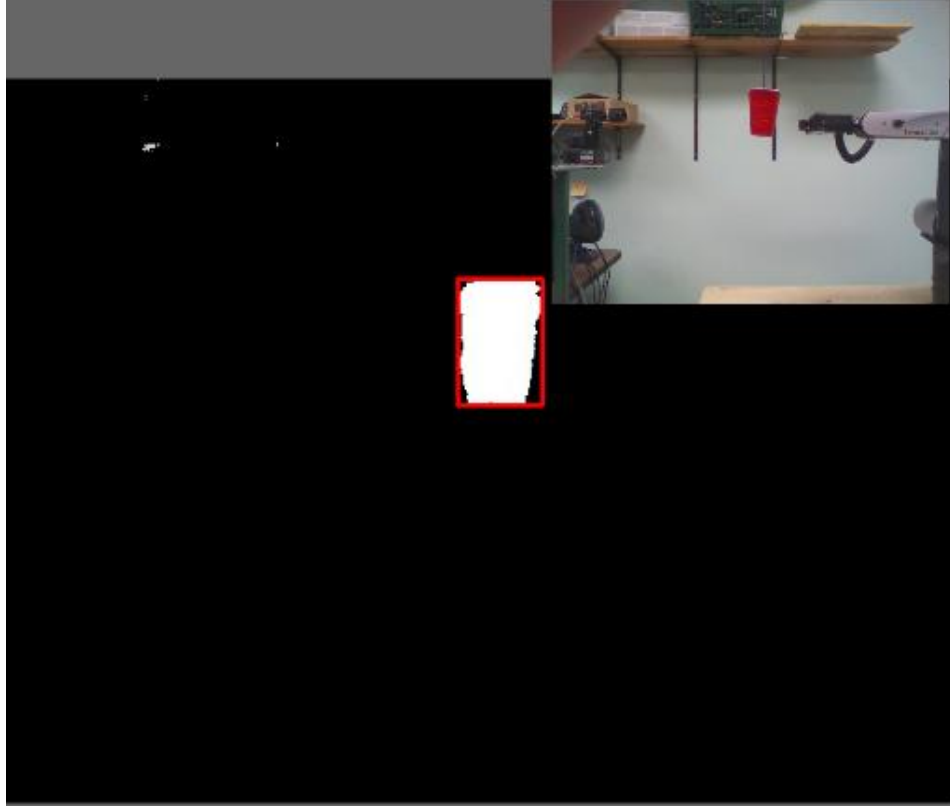


Figure 31. The RGBD data after filtering the contours.

In Fig. 31 the object of interest is detected and shown with a red rectangle. As the contour is detected and the red rectangle is drawn, the center of the rectangle is approximately determined in pixel values as (x_o^i, y_o^i) . The image frame size is 640×480 . Accordingly, the pixel number of the center will be calculated by

$$N = [y_o^i] * 640 + [x_o^i] \quad (52)$$

Based on N , the pixel's depth can be obtained as z_o^c .

At this point the only parameter which is in the camera coordinate system is z_o^c . To find the corresponding parameters for x_o^i and y_o^i in camera coordinate frame,

we need to employ the model which was provided under camera calibration section in chapter II and proceed with velocity estimation.



Figure 32. The location of the Kinect and robot.

Fig. 32 illustrates the actual system in which both CRS Catalyst 5 and Kinect can be observed. In this figure the object is not present. The arm is in its home position from which its manipulation starts and after each experiment, it will return to this position. The Kinect is placed on the top of a wooden base to compensate for the height of the robot's base table. This allows the Kinect to observe the whole working volume of the arm.

The kinematic limitations of the equipment affects the system's ability for a successful interception. Understanding these limits will help to have a better perception of the practical benefits of velocity estimation in a single image frame. Table (6) represents the kinematic limits of the equipment in Cartesian coordinate system. These limits can be altered by changing the control setup of the equipment.

However, the manufacturing company recommends to avoid exceeding more than 200 % of the proposed limits. Otherwise, it can severely damage the robot.

Table 6. World coordinate kinematic limits of the equipment.

Coordinate	Speed limit ($\frac{mm}{s}$)	Acceleration limit ($\frac{mm}{s^2}$)
X	30000	30000
Y	30000	60000
Z	30000	60000

We must consider that the end-effector always starts its motion from zero velocity. Then its displacement along any axis can be determined solely by the acceleration limits and the motion time. There are ultimate number of ways to throw the cup as it can have an ultimate number of directions and velocities. The theory tells us, if the moving object will be present in the working volume after the predefined time interval, the robotic arm must be able to intercept it. However, if the object passes beyond the boundaries of the working volume, the robot will never be able to perform a successful interception.

Let's consider a condition that the system requires two image frames to estimate the velocity. Taking $t = 0.1\text{ s}$ as the predefined time interval, the motion of the end-effector will start after the acquisition of the second image frame. This means that the robot needs to wait until the second image frame is processed and then start its motion. As the frame rate is 30 fps, the amount of time that the arm has to perform the interception is determined by

$$t_1 = t - \frac{1}{30} \quad (53)$$

which is approximately equal to 0.067 s. We take $a_x^w = 30000 \frac{mm}{s^2}$ from table (4).

Accordingly, the displacement of the end-effector along x axis after t is

$$\Delta x = \frac{1}{2} a_x^w t^2 \quad (54)$$

which would be equal to 67.335 mm . If we consider the condition for the end-effector's motion along other axes, then a_y^w and a_z^w are equal to 60000 $\frac{mm}{s^2}$ and the displacements will be

$$\Delta y = \frac{1}{2} a_y^w t^2 \quad (55)$$

and

$$\Delta z = \frac{1}{2} a_z^w t^2 \quad (56)$$

Accordingly, the value of both will be equal to 134.67 mm . Now, based on these displacements a volume can be calculated. This volume represents the end-effector's chance to successfully intercept with the moving object after the predefined time interval. Let's take V_1^w as the volume for this condition. V_1^w will be calculated as

$$V_1^w = \Delta x \Delta y \Delta z \quad (57)$$

which will be equal to 1221188.16 mm^3 . Now if we consider the condition of estimating the velocity with the initial frame, the results will be considerably different.

If the robot starts its motion after the acquisition of the initial frame, the robot will have 0.1 s to react. we can recalculate the displacements on the axes. According to (54), (55) and (56), the new values for Δx , Δy , and Δz will be 150 mm ,

134.67 *mm*, and 300 *mm*, respectively. As it can be seen, the value of Δy has not changed. The reason is that with the motion blur we can only estimate the object's velocity along x and y axes in the camera frame. After transformation from camera frame to robot's frame, we would have the velocity along x and z axes. Based on the new displacements, the new volume will be

$$V_2^w = \Delta x \Delta y \Delta z \quad (58)$$

which will be equal to 6060150 *mm*³. Now if we compare V_1^w with V_2^w as

$$\frac{V_1^w}{V_2^w} \approx 0.2 \quad (59)$$

we will find out that V_2^w is about 5 times bigger than V_1^w . This difference between the volumes represents the difference between the system's ability for having a successful performance under the explained conditions. Obviously, the kinematic limitations of the equipment remains the same and the procedure would be regardless of the moving object's direction of motion and velocity. Now let's imagine if we could estimate v_y from the initial image frame as well. Then Δy would be equal to 300 *mm*. Accordingly, V_2^w would become equal to 13500000 *mm*. If we compare this value with V_1^w , it can be seen that

$$\frac{V_1}{V_2} \approx 0.09 \quad (60)$$

which means V_2^w would become approximately 11 times bigger than V_1^w . This comparison shows the significant benefit of velocity estimation in the initial image frame.

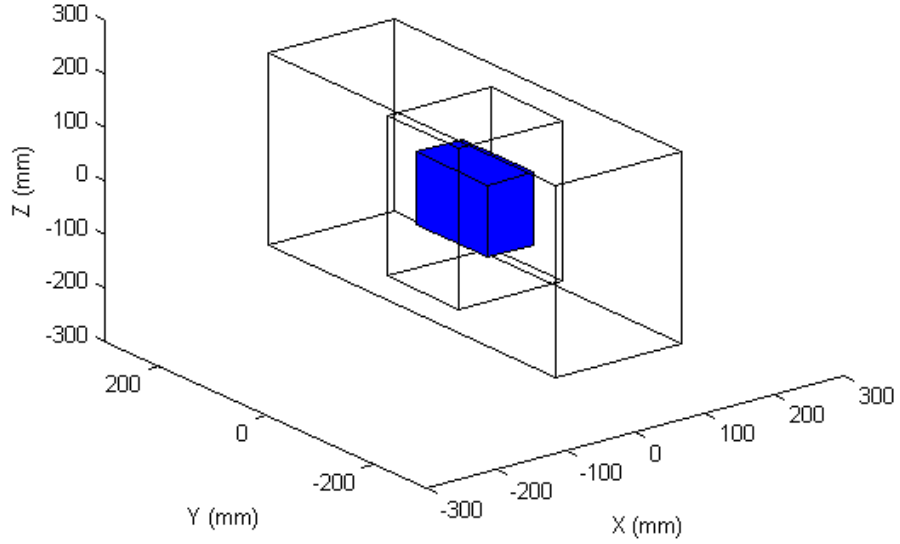


Figure 33. The 3D representation of the end-effector's motion boundaries within 0.1 s.

The main discussion is when the system can estimate the prospective position of the moving object. Let's imagine an object is thrown to our face and it takes 2 s to reach us. If we notice the object 0.5 s after it is thrown compared to 1 s, we would have a better chance of protecting our face.

Keeping in mind that our reaction time remains the same in both conditions. The same situation is present for our robotic system. If our robotic system can detect the object and estimate its future position with the first image frame, it will have a better chance to intercept it compared to when it needs to wait for the second frame.

Fig. 33 depicts both of the volumes. The blue volume represents V_1^w , the middle size transparent volume is V_2^w and the biggest volume is the whole working volume of the robot. Fig. 34 shows the volumes from top view. In all of these figures the robot's end-effector is located at (0,0,0).

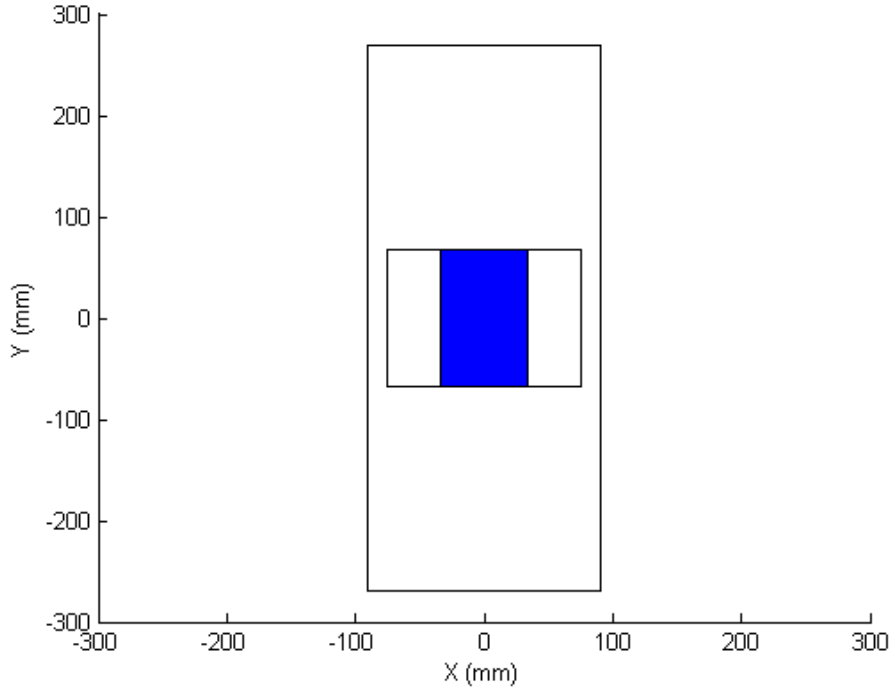


Figure 34. The top view representation of the end-effector's motion boundaries within 0.1 s.

As it can be seen V_1^w and V_2^w are located inside the working volume of the robot. However, the blue volume represents the chance of successful interception, if the motion starts after the second frame. But the middle volume is its chance, if the motion starts after the first frame. In order to better understand the difference, we can study the case with a moving object. Fig. 35 shows a top view of the volumes with a moving object and Fig. 36 shows the 3D view of the same environment.

In Fig.35, the black star represents the initial position of the moving object. We assume that this is once the initial image frame is obtained. The red circle represents the prospective position of the target after 0.1 s. To simplify the example, we can assume that the motion on the $x - y$ plane is linear. As it can be observed, the object will pass through both the blue and transparent volumes.

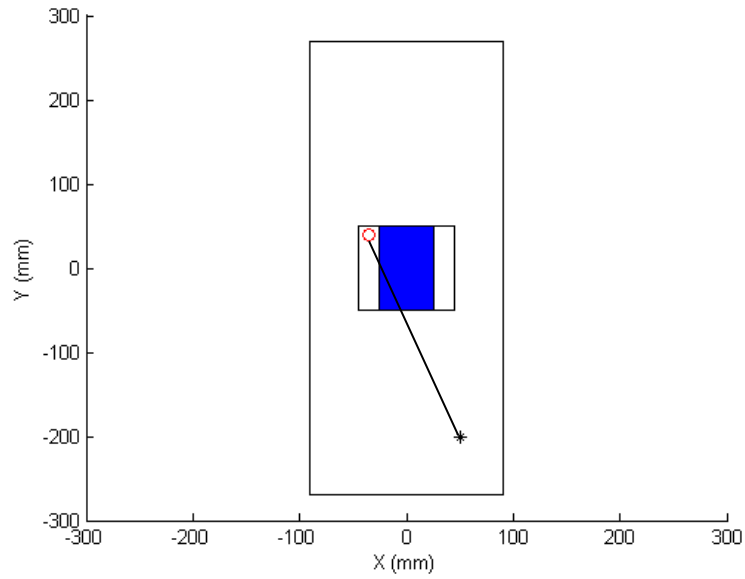


Figure 35. The top view of the working volume and a moving object.

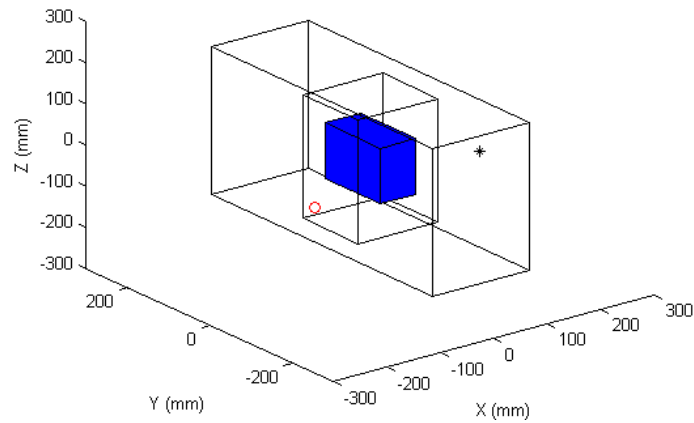


Figure 36. The 3D view of the working volume and a moving object.

Obviously, if the robot starts its motion after the acquisition of the second image frame, the target would have passed the boundaries of the blue volume. However, if the motion starts after the initial image frame, the end-effector will reach the prospective position of the target within the predefined time interval. At this point, a question may come to mind that what if the 0.1 s time interval starts after the second frame. This theory will not increase the chance of a successful interception.

The reason is that the prospective position of the object after the time interval is estimated from the initial image frame and if we consider the second image frame as the origin of the target's motion, the object will be farther away from the position that is currently shown in Fig. 35. So the robot will not be able to successfully intercept with it at all.

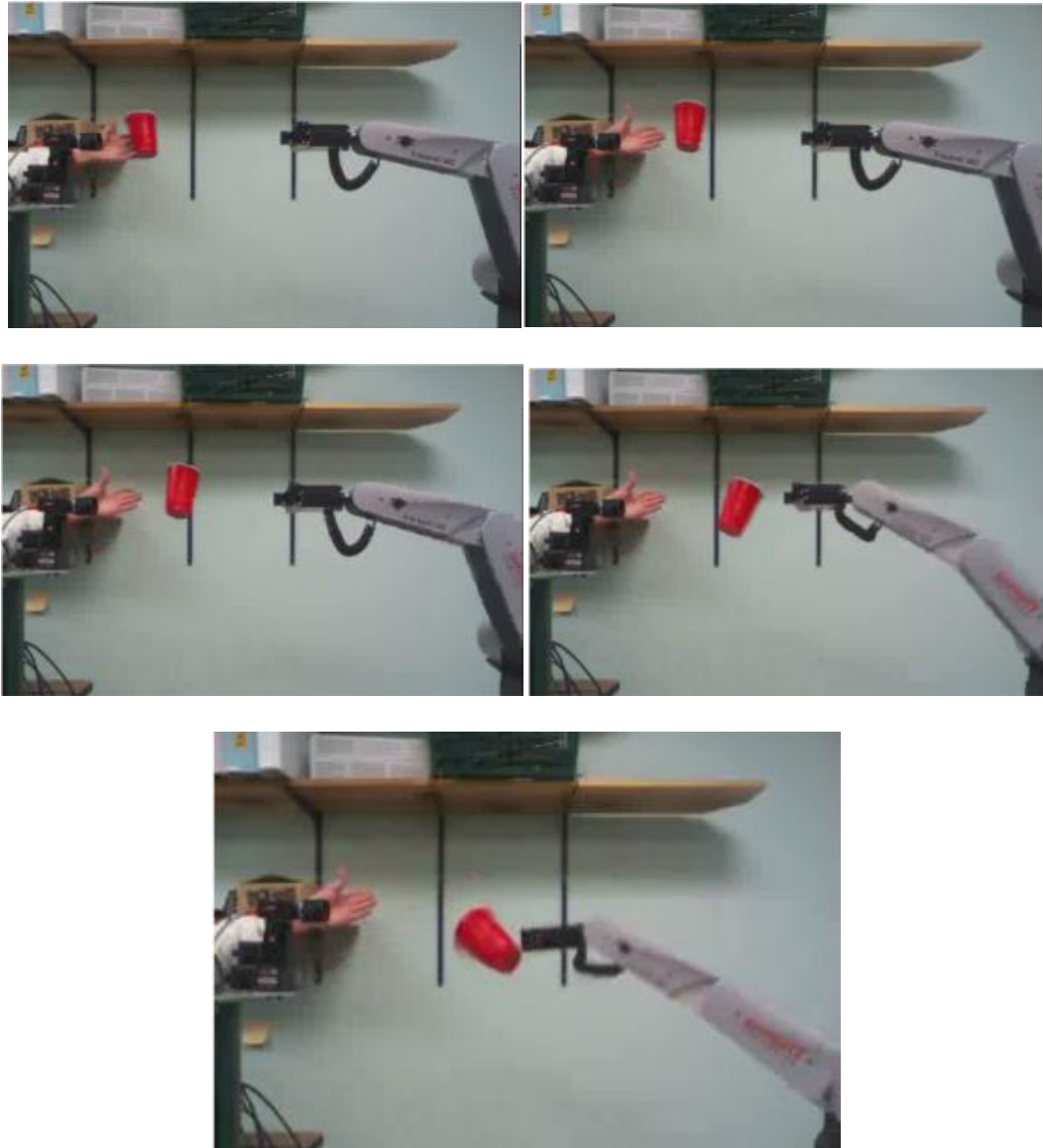


Figure 37. The images from left to right and top to bottom show the interception with the thrown cup. The last image depicts the moment of interception.

Fig. 37 and 38 present two successful attempts while the cup was thrown by the operator. In Fig. 37, the cup is thrown to the far right side of the arm, along x direction and the arm successfully intercepts it.

In Fig. 38, the cup is thrown across the working volume of the robot, approximately along y direction, and again, the robot successfully intercepts the cup within the proposed working volume.

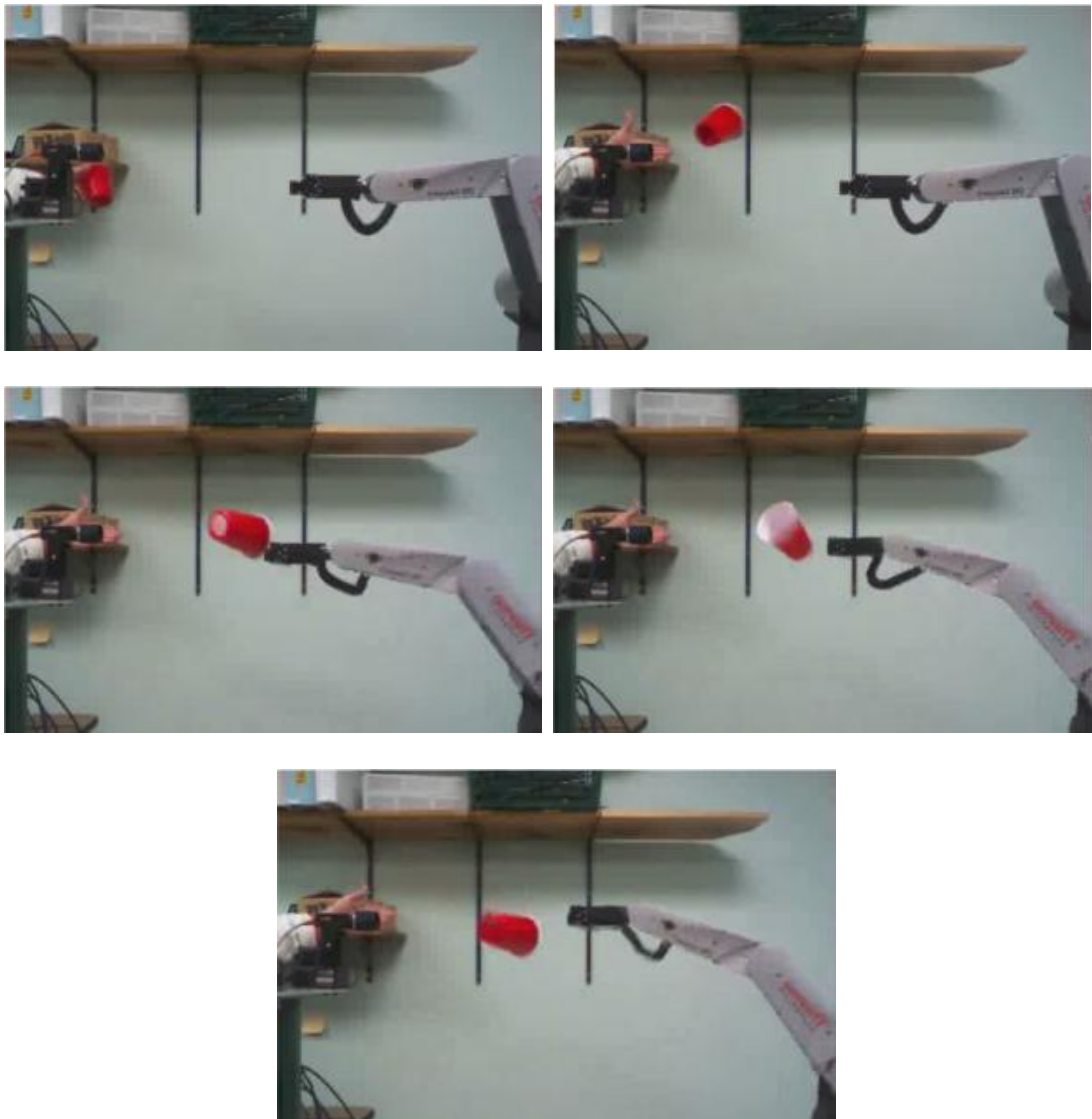


Figure 38. The images from left to right show the interception with the thrown cup in a different location. The last image shows the moment after interception.

In the previous figures, the images are recorded by the Kinect itself. So the images are actually what the Kinect observes during the procedure. The experiment is repeated 25 times.

Every time the experiment is performed, the velocity data is recorded. All of the recorded data of the experiments are presented in table (7). This table contains the velocity of the cup as well as a checkmark indicating the successful completion of the task.

Table 7. The velocity data of the experiments.

Experiment	v_x (m/s)	v_y (m/s)	v_z (m/s)	Result
1	-2.1	-1.5	4.12	✓
2	3.5	3.5	3.1	✓
3	3.01	-4.5	2.5	×
4	-1.45	-2.5	2.3	✓
5	3.5	2.1	-1.5	✓
6	-4.13	-7.3	-2.24	×
7	3.12	3.8	1.1	✓
8	3.1	4.5	1.7	✓
9	-3.2	-0.26	-2.7	✓
10	-1.6	0.71	1.8	✓
11	-1.1	1.2	1.38	✓
12	2.3	-4.1	2.12	✓
13	0.7	7.4	-3.88	×
14	-0.4	-1.73	4.48	✓
15	1.4	-4.64	2.33	✓
16	-2.81	0.78	1.27	✓
17	1.91	-1.37	-3.38	✓
18	2.61	2.21	2.56	✓
19	-2.7	-1.27	0.34	✓
20	2.47	-0.101	2.18	✓
21	-4.34	1.12	-2.62	✓
22	-2.78	-2.34	1.25	✓
23	-1.3	1.54	2.62	✓
24	-3.35	3.12	-2.91	✓
25	-0.7	7.13	3.41	×

As it can be seen in table (7), the object has been thrown with different velocities. And the robot has been successful for approximately 84 % of the time. There are multiple reasons for our system to miss the target. It is not possible to study the chance of a successful interception solely based on the velocity of the target. The direction of motion plays an important role as well. As stated before, there are ultimate number of ways to have the target pass through the working volume of the end-effector. Consequently, there is only one way to surely say that the interception will happen and that is when the object is inside the determined working volume and the boundary limits of the end-effector after the predefined time interval.

According to table (7), experiment 3, 6, 13 and 25 show that the robot has failed to intercept with the target. By paying attention to the recorded velocity data, it can be seen that the velocity varies in a considerably wide range. The smallest recorded value for the velocity is approximately 3 m/s and the biggest is approximately 7 m/s . According to the recorded vectors of velocity, we can conclude that the system posses acceptable symmetrical ability for determining the velocity and prospective position of the moving object. It means the direction of the object, as long as it stays inside the reachable boundaries of the end-effector after 0.1 s, will not affect the chance of a successful interception. This is concluded based on the positive or negative directions of the velocity vectors. For instance if we consider experiments (3) and (4), we will see that v_y is negative in both of the tests. However, (3) is a failure and (4) is a success. This proves that the direction of the velocity vector has not affected the performance of the system and the failure must have had a different reason such as faster motion along y . The same situation is present for the v_z vectors in (5) and (6).

Another factor which plays an important role in the process of velocity estimation is the motion along y axis. It must be noted that the robot's y axis is in fact the Kinect's z axis, the axis corresponding to depth. If we remember from the previous chapters, we could not take advantage of motion blur speed estimation for the depth as it does not contain any blur. Consequently, the robot receives the velocity vectors along x and z from the first frame. But the velocity along y is obtained after the acquisition of the second image frame. The robot starts its motion with the initial frame and then it modifies its path after the second image frame. Consequently, the system cannot react to motion along y as fast as motion along the other two axes. If we consider experiments (3), (6), (13) and (25), we can see that in all of which v_y has a greater value compared to v_x and v_z . In (6), (13) and (25) the absolute difference between v_y and the other vectors is greater than approximately 4 m/s.

Another main reason for failure is operator's error. It is very important to throw a cup through the working volume of the robot. The operator has an approximate knowledge of where the working volume of the robot is, but it is still difficult to pass the target exactly through the volume. In addition, the object might partially pass through the view point of the camera. Consequently, the vision system may fail to detect the target at all. Because of the direction of motion, the camera can sometimes detect the object but it leaves the arm's working volume before the robot will be able to react.

In this chapter we provided the results of 25 tests and proved how velocity estimation in a single image frame can increase the arm's chance for a successful interception. The major limitation of our setup was its inability to estimate the

velocity along y axis from the initial image frame. In addition, the kinematic limitations of our robotic arm restricted the system's ability to intercept with faster moving objects.

CHAPTER V

CONCLUSION

In this thesis, we presented a robotic system to autonomously detect the presence of a moving object outside the working volume of a robotic arm, estimate its velocity in a single image frame through motion blur, predict its prospective position after a time interval, and intercept it in the predicted position.

The main contribution of the work was the velocity estimation for a real time application. This method has never been used for a real time system before. In addition, the majority of previous work in this area employed specialized manipulators which were solely designed for the purpose of moving object interception. However, we implemented the method on a robotic arm which is not designed for the task of moving object interception.

The system was implemented on a testing setup which consists of a windows Kinect camera and CRS-Catalyst 5 robotic arm. Our manipulator has not been designed and never been employed for high speed, acceleration and torque tasks in previous research. Besides, the previous works in this area show that most of the projects have been attempted with equipments, which were designed and built for the sole purpose of moving object interception.

The major goal in this thesis was to present how velocity estimation from the initial image frame can improve the performance of our robotic system for the task of

moving object interception. The system's functionality is compared to when it requires at least two image frames to estimate the target's velocity. This work shows that our method improves the performance of the robotic system more than three times. As the motion blur methods cannot be implemented on the depth image frame, the velocity along y direction in the world coordinate frame could not be estimated from the first image. Theoretical results suggested that if we could estimate the velocity along the aforementioned axis in the first image frame as well, the system's performance could be improved more than five times.

The experiment was repeated 25 times and the results were provided. The obtained data showed that the system was able to intercept almost 84% of the time. Among the reasons for failure, the operator's error played a major role. It is significantly important to throw the object inside the working volume of the robot. In addition, as the system cannot determine the velocity along y , it is more susceptible to failure when the object is moving faster along this axis. This faster movement is in comparison with the object's velocity along the other axes.

The importance of this method to the field of robotics can be significant as it makes the task of moving object velocity estimation faster without altering the equipments such as the use of high speed cameras. This method can improve the performance of all the robotic systems which use visual sensing. A considerable number of such systems need to function in an environment with dynamic obstacles. For instance, there are industrial robotic arms which work interactively with human operators. Humans can be considered dynamic obstacles and it is significantly important to ensure their safety in such kind of working environment. This method will help such robots to sense the humans faster at a low cost. In addition, as these

robotic arms need to grasp and manipulate moving objects, this method will assist them to notice the target faster and then it will increase the productivity of the industrial site. Considering the fact that this method can be implemented on any robotic system which takes advantage of RGBD cameras, an increase in the robotic jobs can be expected as well. Moreover, we can imagine rescue or military robots which navigate in the environments with dynamic obstacles too. In the case of humanoid robots, this method can be a significant help to their visual abilities.

As explained under the second chapter, a lot of times other methods for velocity estimation employ more expensive tools to perform the task. By using this method, such kinds of costs, either in the area of research or industry, will be reduced.

For the future research, our project can be extended in several areas. One of the main parts of the project which can be modified is the object detection algorithm. Our object detection model is solely based on contour area size in the depth image. Actually, the model can be improved to detect objects with more details in a greater depth range. The most important ability of the system which can be modified is to estimate velocity along y^w from the initial image frame as well. As the theory showed, if we could estimate the velocity along this axis like the other vectors, the system's ability would be considerably increased.

REFERENCE

- [1] M. Garcia, et al., "Vision-guided motion primitives for humanoid reactive walking: Decoupled versus coupled approaches," *The international journal of robotics research*, 2014.
- [2] M. Land and P. McLoed, "From eye movements to actions: how batsmen hit the ball," *Nature neuroscience*, vol. 3, pp. 1340-1345, 2000.
- [3] M. Richter, Y. Sandamirskaya, and G. Schone, "A robotic architecture for action selection and behavioral organization inspired by human cognition," *IEEE/RSJ international conference on intelligent robots and systems(IROS)*, 2012.
- [4] <http://www.classroomclipart.com> on [April, 7th, 2015].
- [5] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," *Proceedings of IEEE/RSJ International Conference on Robotic Systems*, vol. 3, pp. 1623–1629, 2001.
- [6] W. Hong and J.-J. E. Slotine, "Experiments in hand-eye coordination using active vision," *Proceedings of 4th International Symposium Expo. of Robotics*, pp. 130-139, 1997.
- [7] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," *Proceeding of IEEE/RAS International Conference on Humanoid Robots*, pp. 875-881, 2012.
- [8] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time,"

Proceeding on International Conference in Robotics and Automation, pp. 3719–3726, 2011.

- [9] A. Namiki and M. Ishikawa, “Robotic catching using a direct mapping from visual information to motor command,” Proceeding of IEEE International Conference in Robotics and Automation (ICRA), vol. 2, pp. 2400-2405, 2003.
- [10] G. Park, K. Kim, C. Kim, M. Jeong, B. You, and S. Ra, “Human-like catch-ing motion of humanoid using evolutionary algorithm(ea)-based imitation learning,” in Proc. IEEE Int. Sympo. Robot Human Interact. Commun.,2009, pp. 809–815.
- [11] M. Riley and C. G. Atkeson, “Robot catching: Towards engaging human- humanoid interaction,” Robotics and Automation, vol. 12, no. 1, pp. 119–128, 2002.
- [12] M. Zhang and M. Buehler, “Sensor-based online trajectory generation for smoothly grasping moving objects,” Proceedings of IEEE International Symposium in Intelligence Control, pp. 16–18, 1994.
- [13] S. Kim, A. Shukla, and A. Billard, “Catching objects in flight,” IEEE Transactions on Robotics, vol. 30, no.5, pp. 1049-1065, 2014.
- [14] J. Kober, K. Mulling, O. Kromer, C. H. Lampert, B. Scholkopf, and J. Peters, “Movement templates for learning of hitting and batting,” Proceedings of IEEE International Conference in Robotics and Automation (ICRA), pp. 853–858, 2010.
- [15] T. Senoo, A. Namiki, and M. Ishikawa, “Ball control in high-speed batting motion using hybrid trajectory generator,” Proceedings of IEEE

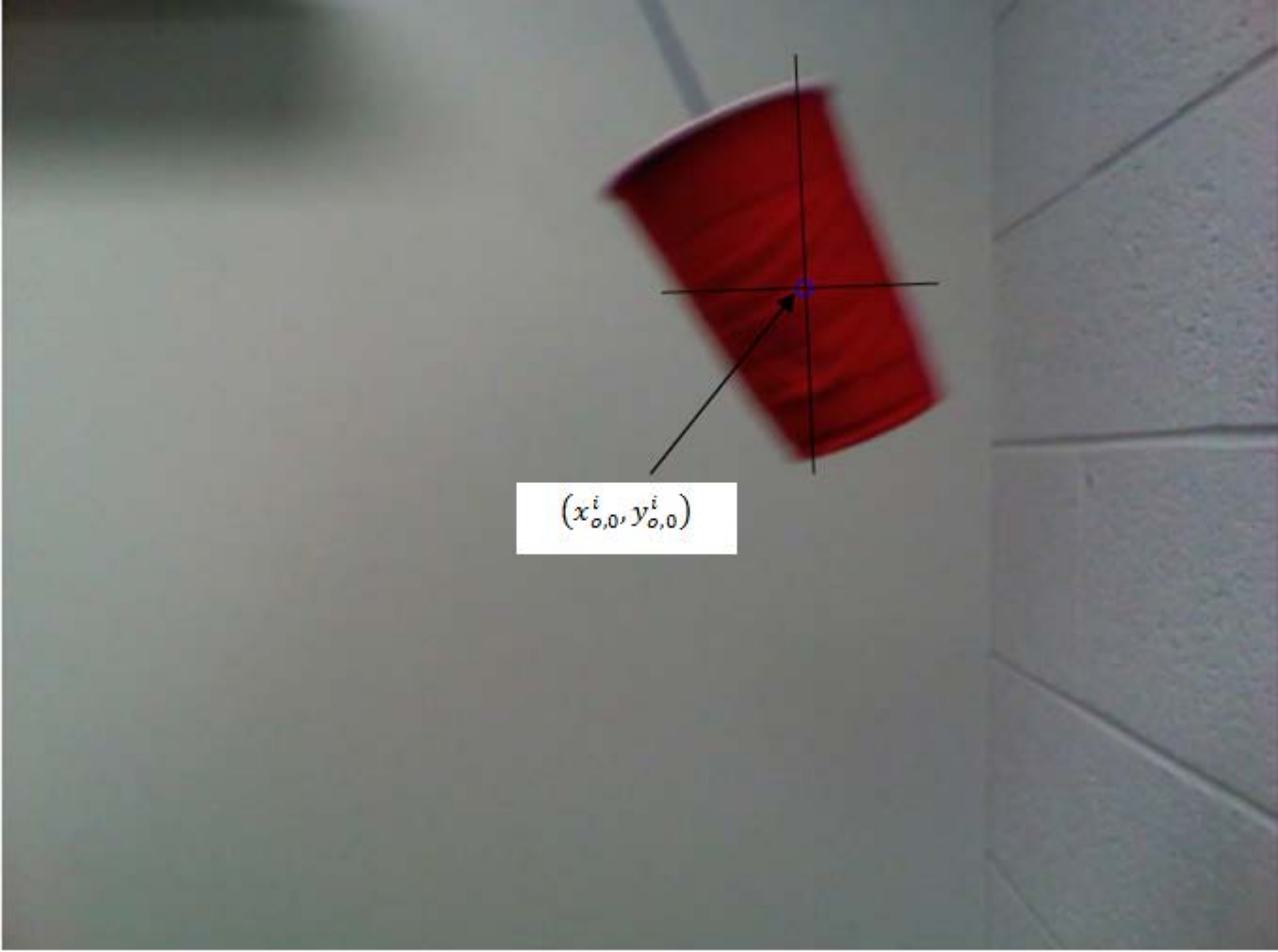
- International Conference in Robotics and Automation (ICRA), pp. 1762–1767, 2006.
- [16] M. Buehler, D. E. Koditschek, and P. J. Kindlmann, “Planning and control of robotic juggling and catching tasks,” *International Journal of Robotics*, vol. 13, no. 12, pp. 101–118, Apr. 1994.
 - [17] A. Rizzi and D. Koditschek, “Further progress in robot juggling: Solvable mirror laws,” *Proceedings of IEEE International Conference in Robotics and Automation (ICRA)*, vol. 4, pp. 2935–2940, 1994.
 - [18] S. Schaal, D. Sternad, and C. G. Atkeson, “One-handed juggling: A dynamical approach to a rhythmic movement task,” *J. Motor Behav.*, vol. 28, pp. 165–183, 1996.
 - [19] <http://www.geek.com> visited on [April, 9th, 2015].
 - [20] <http://www.kuka-timoboll.com> visited on [April, 9th, 2015].
 - [21] H. H. Rapp, “A ping-pong ball catching and juggling robot: a real-time framework for vision guided acting of an industrial robot arm,” *5th IEEE International Conference on Automation, Robotics and Applications (ICARA)*, 2011.
 - [22] <http://www.dailymail.co.uk> visited on [April, 5th, 2015].
 - [23] discovermagazine.com visited on [April, 5th, 2015].
 - [24] www.darpa.mil visited on [April, 10th, 2015].
 - [25] K. Jens, M. Glisson, and M. Mistry “Playing catch and juggling with a humanoid robot,” *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012.

- [26] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," Cambridge University Press. pp. 155–157. ISBN 0-521-54051-8, 2003.
- [27] A. Crétual, and François Chaumette, "Visual servoing based on image motion," The International Journal of Robotics Research vol. 20, no. 11, pp.857-877, 2001.
- [28] K. Hashimoto, "A review on vision-based control of robot manipulators," Advanced Robotics vol. 17, no. 10, pp.969-991,2003.
- [29] F. Chaumette, "Visual servoing," Computer Vision: A Reference Guide, pp.869-874, 2014.
- [30] P.K. Allen, et al., "Automated tracking and grasping of a moving object with a robotic hand-eye system," IEEE Transactions on Robotics and Automation, vol. 9, no. 2, pp.152-165,1993.
- [31] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Binocular visual tracking and grasping of a moving object with a 3D trajectory predictor," Journal of applied research and technology vol. 7, no. 3, pp.259-274, 2009.
- [32] B. Bauml, T. Wimbock, and Gerd Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.
- [33] C. Smith, and H. I. Christensen, "Using COTS to construct a high performance robot arm," IEEE International Conference on Robotics and Automation, 2007.

- [34] P. Piersigilli, I. Sharf, and A. K. Misra, "Reactionless capture of a satellite by a two degree-of-freedom manipulator," *Acta Astronautica* vol. 66, no. 1, pp.183-192, 2010.
- [35] A. Nagendran, W. Crowther, and R. C. Richardson, "Dynamic capture of free-moving objects," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* vol. 225, no. 8, pp.1054-1067, 2011.
- [36] S. H. Yeo, et al., "Eyecatch: simulating visuomotor coordination for object interception," *ACM Transactions on Graphics (TOG)* vol. 31, no. 4, 2012.
- [37] V. Lippiello, and Fabio Ruggiero, "3D monocular robotic ball catching with an iterative trajectory estimation refinement," *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.
- [38] J. J. Neubert, N. J. Ferrier, "Direct mapping of visual inputs to motor torques," *IEEE 18th International Conference on Pattern Recognition*, vol. 4, 2006.
- [39] J. A. Johnson, and J. Neubert, "Robotic reactive motion with jerk reduction," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012.
- [40] P. Sturm and S. Maybank, "On plane-based camera calibration: a general algorithm, singularities, applications," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 432–437, Fort Collins, CO, USA, June 1999.

- [41] www.vision.caltech.edu/bouguetj/calib_doc visited on [October, 20, 2014].
- [42] J. Heikkila, and O. Silvén, “A four-step camera calibration procedure with implicit image correction,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997.
- [43] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 1. 1999.
- [44] <http://www.blenderartists.org> visited on [March, 23, 2015].
- [45] F. Ram and D. Malah, “Robust identification of motion and out-of-focus blur parameters from blurred and noisy images,” CVGIP: Graphical Models and Image Processing, vol.53, no.5, pp. 403-412, 1991.
- [46] M. M. Chang, A. M. Tekalp, and A. T. Erdem, “Blur identification using the bispectrum,” IEEE Transactions on Signal Processing, vol.39, no.10, pp. 2323-2325, 1991.
- [47] H.Y Lin, K.J.Li, and C.H Chang, “Vehicle speed detection from a single motion blurred image,” Image and Vision Computing vol.26, no. 10, pp.1327-1337 , 2008.
- [48] R. P. Paul, “Robot manipulators, Mathematics, Programming and Control,” MIT Press, Cambridge, Mass., 1981.
- [49] A. Dunkels, “Full TCP/IP for 8-bit architectures,” Proceedings of the 1st international conference on Mobile systems, applications and services. ACM, 2003.

- [50] M. R. Banham, and A. K. Katsaggelos, "Digital image restoration,"
IEEE Signal Processing Magazine, vol.14, no.2, pp. 24-41, 1997.



A photograph of a red plastic cup tilted against a light-colored wall. A black coordinate system is overlaid on the cup, with the origin marked by a blue dot. An arrow points from a white box containing the mathematical expression $(x_{o,0}^i, y_{o,0}^i)$ to this blue dot. The background shows a wall with horizontal lines and a blurry object in the upper left.

$$(x_{o,0}^i, y_{o,0}^i)$$

