APPLICATION OF HAND-EYE COORDINATION IN REACHING WITH A HUMANOID

BY

ZEYUAN YU

SENIOR THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the Undergraduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Advisor:

Professor Stephen E. Levinson

Abstract

Reaching is the prerequisite for all kinds of physical manipulation by which robots interact with the physical world. In this research project, we try to build an application on the iCub humanoid platform to reach visually seen objects located in workspace. We also investigate humans' ability to reach to objects using visual feedback and intend for the application to reproduce human-like behavior with a simplified model. This application includes fine motor control, object recognition, spatial transformations, stereo-vision, gaze control, and hand-eye coordination. We also discuss different methods in both vision and control that we use in the application.

Subject Keywords: Humanoid; Hand-eye coordination; Reaching

Acknowledgments

My advisor, Dr. Stephen Levinson, has helped me through this project and encouraged me to discover more unknows not only in the robotics field, but also in cognitive sciences. I learned new ideas and concepts in every conversation with him and other lab members. Jacob Bryan helped me with the installation of iCub and YARP on different Ubuntu systems and I wouldn’t be able to conduct any research without his help. Yuchen He generously offered help in the lab, from stereo vision to motor control. Luke Wendt always brought me another critical and creative view of the same topic. I would also like to thank Felix Wang, for his emotional support at Perkins at 4am.

Other than lab members, I would like to thank my friend Kangyan Zhou, for all conversations and arguments we had in the topic of intelligence. My cats, Q and Perry, also encouraged me to keep thinking about the difference between animals and human beings, from self-recognition to intelligence.

Contents

1. Introduction 1

2. Literature Review 3

3. Description of Research Results 4

3.1 Application Platform Setup 4

3.1.1 The iCub Platform 4

3.1.2 YARP 5

3.1.3 The iCub Simulator 5

3.2 Application Design 6

3.3 Application Components 8

3.3.1 Motor Control and Kinematics 9

3.3.2 Object Recognition 11

3.3.3 Stereo Vision Depth Map and Spatial Transformation 14

3.3.4 Hand Tracking with Optical Flow 15

4. Conclusion and Future Work 18

References 20

1. Introduction

A well-known dichotomy between machines and humans is described by Moravec's paradox [1], where machines can execute tasks that humans find hard but tend to have a much less satisfying performance on "easy" tasks for humans like motor activities and "perception". We believe reproducing human's motor abilities on machines might reveal insightful ideas about general intelligence questions. Further, motor activities provide ways for robots to physically interact with the world, including other objects and humans. Reaching is clearly the prerequisite for all kinds of physical manipulation. Here we focus on an application for a humanoid to reach to an object of interest in workspace and then attempt to grasp and lift the object.

Researches in developmental psychology showed that infants can touch and grasp objects without sight of the hand [2]. Other experiments [3] also shown that visual guidance has an increasing effect as infants develop motor abilities. These results revealed that humans' ability to reach, with an analogy to control system, also has two different control mechanisms: an "open loop" controller where one makes the best effort to position the hand near the target object with "proprioceptive" sensory information only, and a "close loop" controller with visual feedback to compensate for errors introduced in the "open loop" controller.

In this research project, our goal is to adopt such an approach with motor control and stereo vision to mimic human behavior. We start by using iCub's vision system to explore surrounding environments and detect an object of interest. The stereo vision cameras then enable us to calculate a depth disparity map to extract the gravity moment's distance to the head. Using spatial transformation to get coordinates of object in 3D space, we feed the coordinates into an inverse kinematics solver to get the target joints space values and activate "open loop" motor control for each joint to bring end-effector close to the target location. Finally, with objects in sight, we use an "optical flow" algorithm to track moving features in image frames from both cameras. We approximate the hand location from edges of those features and estimate the off distance between the robot's hand and the object's gravity moment. That error value is then used to correct the reaching behavior. After we reduce the error to a tolerable range, the robot moves its fingers in a pre-programmed sequence to grasp the targeted object.

2. Literature Review

The problem of reaching visual targets with robot arm requires a priori knowledge of mappings between visual location and arm position. It typically includes the camera projective map (from pixel coordinates to the hand reference frame), the robot forward and inverse kinematics (mappings between the hand reference frame and robot reference frame). There exist many related works in kinematics [4] and hand/eye calibration [5] describing different ways to retrieve these mappings.

3. Description of Research Results

3.1 Application Platform Setup

3.1.1 The iCub Platform

The application described in this thesis were developed on the iCub robot [6], as shown in figure 1. Our iCub robot, Bert, is a whole-body humanoid robot which consists of 53 actuated degrees of freedom. In this project, we only explore the upper-body part of Bert that is essential for reaching tasks, i.e. hand, arm, shoulder and head. The hands and shoulders have tendon driven joints and fingers are flexed by Teflon-coated cable tendons. The arm has 7 degrees of freedom: 3 in the shoulder, 1 in the elbow and 3 in the wrist. The hand has 9 degrees of freedom: 3 in the thumb, 2 in the index, 2 in the middle finger, 1 in the coupled ring and little finger and 1 for adduction/abduction.

The head has two stereo cameras with 30 Hz frame rate and 3 degrees of freedom, mounted on a 3-DOF neck. It was provided with a gaze tracking controller that can be used to look at specific target points in 3D space.

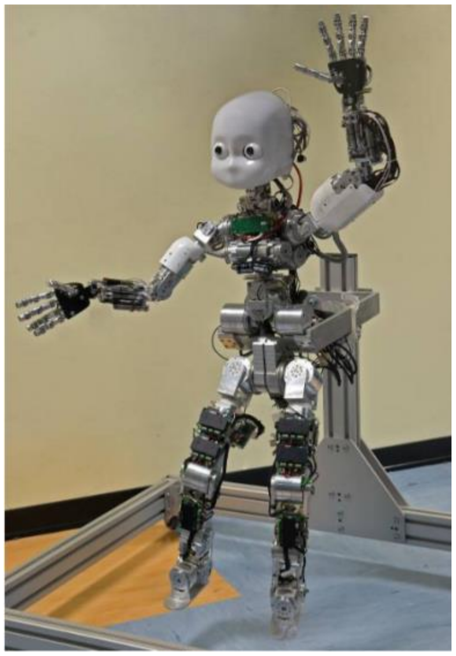


Figure 1: The iCub humanoid robot from [6]

3.1.2 YARP

Yet Another Robot Platform (YARP) is the backbone of the iCub humanoid and described in [7]. YARP is an open-source project of the robotics communication library. It offers an inter-process communication system with different ports, data types and hardware drivers. It is dedicated to modularity and cross-platforms. It also implements a distributed controller in a cluster.

3.1.3 The iCub Simulator

In this stage of this research project, most of the algorithms and modules were tested in the iCub simulator developed by Vadim Tikhanoff, et al. [8]. The simulator, as shown in figure 2, is an attempt to reproduce the physics and dynamics of the robot with only open-source libraries. One of the libraries is the open dynamics engine (ODE), which simulates rigid bodies and physical interactions. The experiment result from the simulator is quite reliable since it was constructed by directly collecting data from the robot design with the same physical attributes.

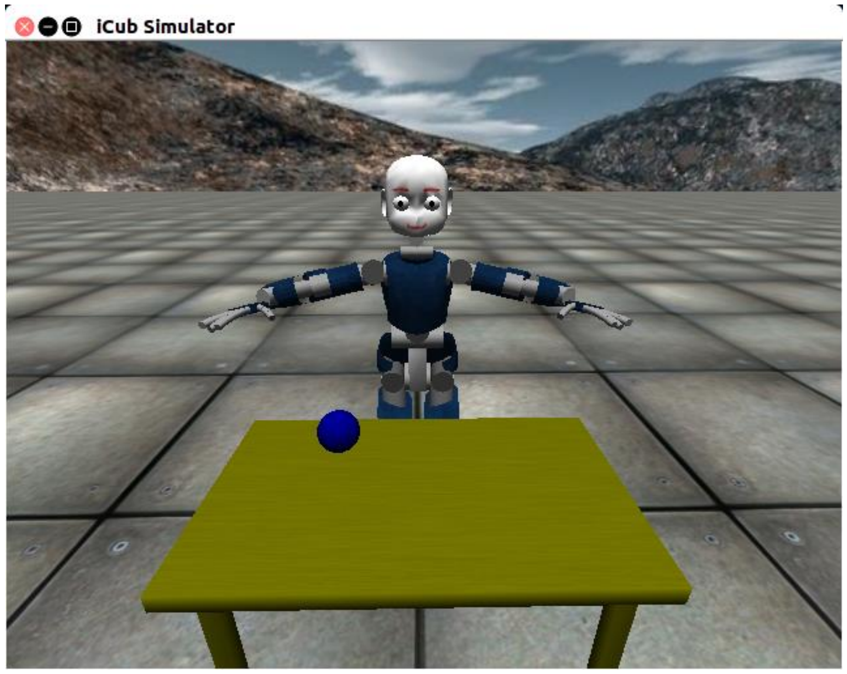
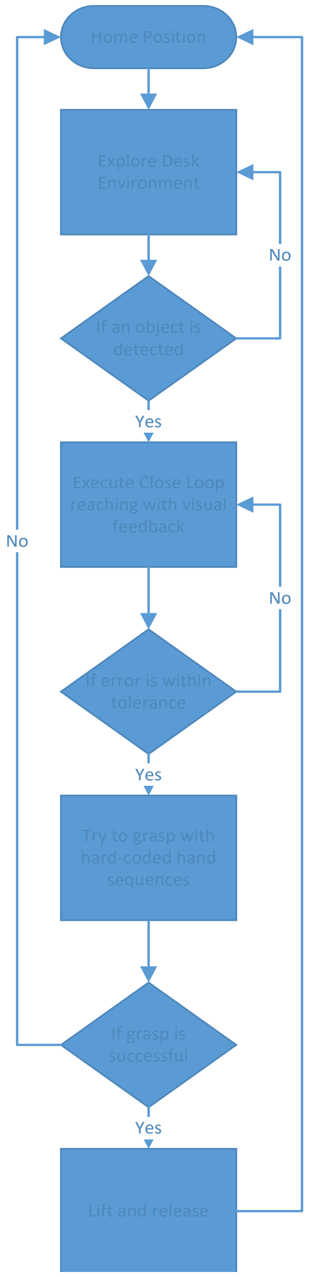


Figure 2: iCub Simulator in Ubuntu

3.2 Application Design

We attempted to reconstruct the way a human performs a reaching and grasping task with visual feedback. Due to the limited time and knowledge as well as limited sensory information on the robot, we intended for this application to be less complicated and yet enough to represent a working model. With this idea in mind, we designed the application logic as shown in figure 3.



Home Position
Explore Desk Environment
If an object is detected
No
Yes
Execute Close Loop reaching with visual feedback
If error is within tolerance
No
Yes
Try to grasp with hard-coded hand sequences
If grasp is successful
No
Yes
Lift and release

Figure 3: Application logic flowchart

The iCub humanoid robot will start in its home position, with arms, torso, and head motors activated. In the home position, the robot places the arms out in front of the body. The "open" hand pose with all fingers naturally extending is used in the home position, before grasping and after releasing. It simulates the opened hand and reduces the complexity of features that appear to the stereo cameras. The robot also looks down at the table in front with an absolute orientation of [0.0 -20.0 0.0] (deg).

Then the robot explores the table environment by simply turning its head to look in both the right and the left directions. In the process, our application retrieves images from both cameras and runs an object detection algorithm. Once an object of interest is identified to exist in both images, the application pauses the current head movement and asks for user confirmation. If positive, the application then steers the head and eyes such that the robot's stereo vision focuses on the object.

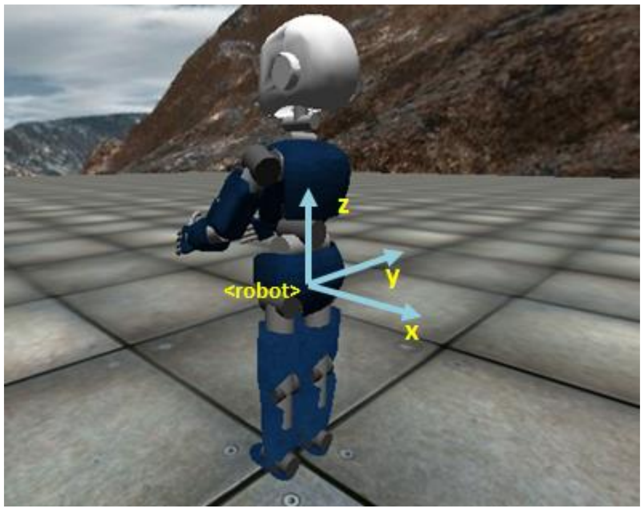


Figure 4: Root frame of iCub robot from [9]

After fixating on the object, the application extracts depth information by using a stereo vision algorithm and estimates the 3D location of the object with respect to the iCub's root frame [9], as illustrated in figure 4. With the approximate coordinates in memory, the application prepares waypoints to be used in the open loop reaching. The waypoints are chosen to make sure the end-effector's trajectory does not collide with the table or the object.

Next, the robot will try to reach to the object. This process is divided into two parts as discussed in the introduction: an open-loop control when the end-effector is far from the object and a close-loop control when the end-effector is close to the last waypoint. With 3D coordinates, the application then calls a provided module which solves the inverse kinematic and moves the motors accordingly. The trajectory is important and discussed further in 3.3.1.

In the third state “grasp”, ideally the robot should have the capability of learning different grasp models for different objects and choose accordingly. However, due to limited time and resources, we decided to use a simple grasp model. Visual feedback is also important because it is the only judgment on the performance. The robot we used does not have tactile sensors.

3.3 Application Components

We designed the application with human hand-eye coordination in mind. Our eyes (the vision system) and our hands (the effector system) are separate functions. In this application, the analogy would be the use of three threads, each representing a function in hand-eye coordination. They are the motor control thread, visual thread and disparity thread.

The motor thread handles all end-effector control, including joint angles and velocities. In this thread, we utilize a controller based on a minimum jerk model to simulate human-like trajectories. This thread will take in control commands from the visual thread.

The visual thread acts as the main thread in the whole application, which communicates between threads and communicates with users through terminal commands. It implements object recognition and hand tracking to issue commands to the motor control thread.

The disparity thread will constantly run, regardless of other threads' status. It simply mimics the subconscious part of our vision system, where we keep extracting depth and position information based on stereo vision. The coordination among 3 threads is described in figure 5.

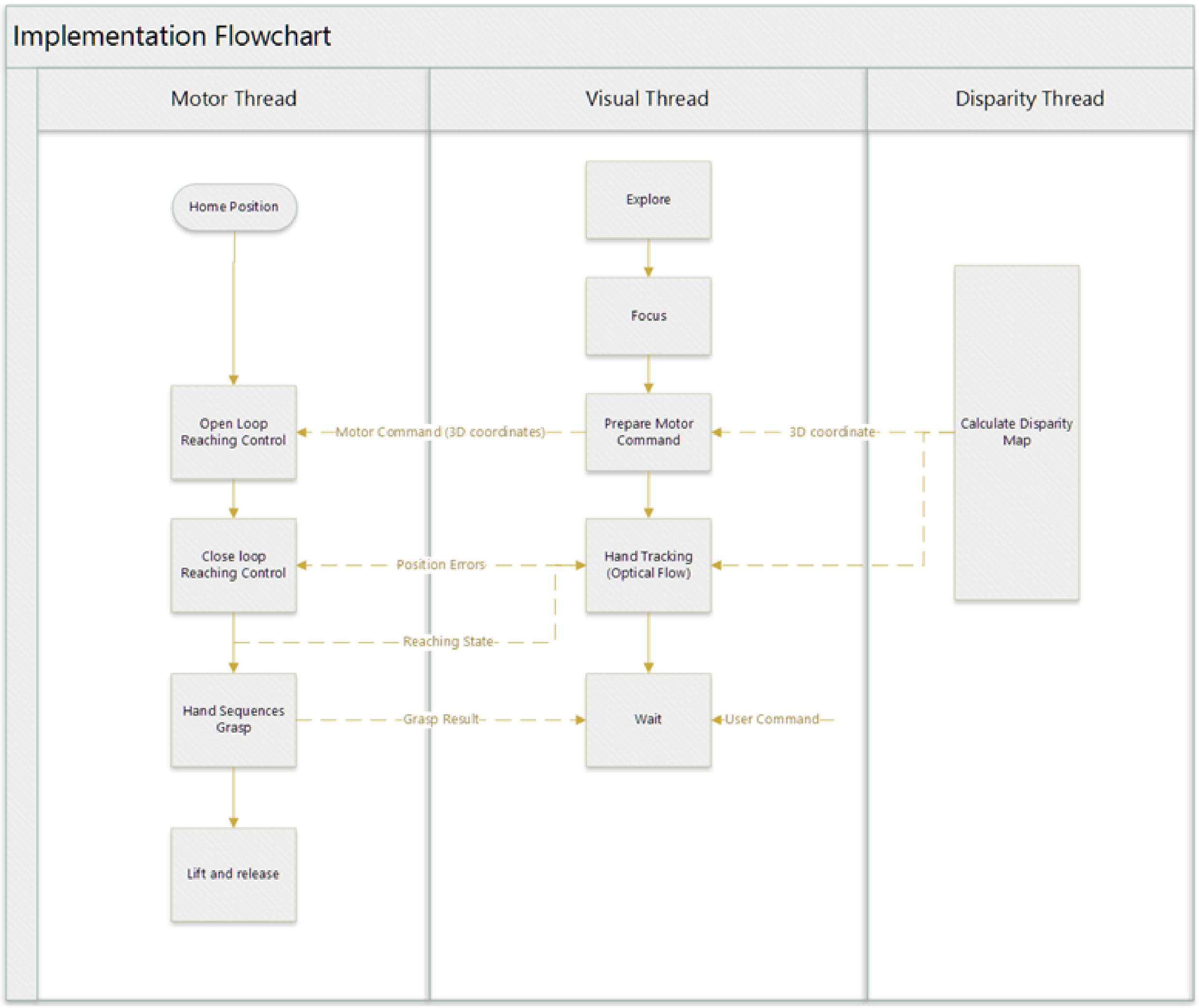


Figure 5: Implementation workflow among three threads

3.3.1 Motor Control and Kinematics

This module directly interfaces with motors in the upper body, including the arm, hand, and head.

The motor controller for the head is developed by Alessandro Roncone, et al. [10]. It features stabilized gaze, saccadic movements, and vestibulo-ocular reflex.

The hand is controlled by programming the angles for each joint. With a hardcoded sequence of joint movements, we mimic the simple behavior of grasping an object from the side.

For the arm manipulator, the individual joints determine the position and orientation of the end-effector or hand in this case. Since our intention is to move the end-effector to target point in 3D space, our goal is to calculate the desired values of the joint variables as followed:

$$\boldsymbol{q} = [q_1 \quad q_2 \quad \dots \quad q_n]$$

from the specified end-effector position and orientation

$$H = \begin{bmatrix} R_{desired} & T_{desired} \\ 0 & 1 \end{bmatrix},$$

where $R_{desired}$ is the rotation matrix and $T_{desired}$ is the translational vector. Hence, we would need to solve the equation:

$$\prod_{k=1}^{n} A_k(q_k) = H,$$

where A is the homogeneous transformation matrix for each joint.

However, simply knowing the starting configuration and target configuration is not enough for controlling arm joints, the movement itself is a continuous process. We would need to control individual joint velocities at each time instant to specify a desired trajectory. Given the infinite number of possible trajectories from one point to another, it would be difficult to hard-code with fixed velocity. Hence, we looked at dynamic system controllers that generate trajectories with a closed loop. After comparing different kinematic controllers from the literature, we decided to use minimum-jerk Cartesian controller [11] for following reasons:

(1) The method proposed uses IpOpt [12], an open source software package for the large-scale nonlinear optimization, which is both reliable and fast enough to satisfy the real-time requirement of motor control for kinematics tasks like reaching.

(2) The minimum-jerk controller is designed to reproduce human-like trajectories with more smooth velocity profiles compared to other models like Vector Integration-To-Endpoint (VITE) [13] and typical Damped Least-Squares (DLS) [14]. Although the minimum-jerk controller has slightly larger position and orientation errors, the effect on performance is limited due to the degree of error [11].

3.3.2 Object Recognition

This module is used to find an object of interest and mostly used in the "explore" state. After detecting an object, this module also calls gaze control to fixate on the object.

Object recognition has been a well-researched area in computer vision and many complex methods are present for better performance in the real-world scene. My partner, Yu Sun, worked on such an algorithm to be used with this application.

Objects can be recognized using many different features. Here we will describe a fast color image segmentation method based on color information of each pixel from two cameras, with the intention that such a less complex algorithm would show acceptable performance in the lab environment and simulation. By using color as the only feature, we avoid the requirement on database and computation.

The objects we use are uniform in shape: pure colored cubes and soda cans. They will be easily separated from background and hence guarantee the efficiency and accuracy. Each object is a blob of one dominant color in the raw image, as shown in figure 6. However, the uncertainties in lighting condition require us to use an approximate range to define the dominant color. Thereby we first normalize the RGB channels intensity, then we use the difference between the desired channel and mean value of other two channels as grayscale values, as shown in figure 7. Next, we threshold the image such that all pixels are either white or black. Then we find all contours in the picture and select the largest one and calculate the geometric moment $\boldsymbol{m} = [u_0 \quad v_0]$. The threshold image is shown in figure 8 with moment marked by a crosshair.

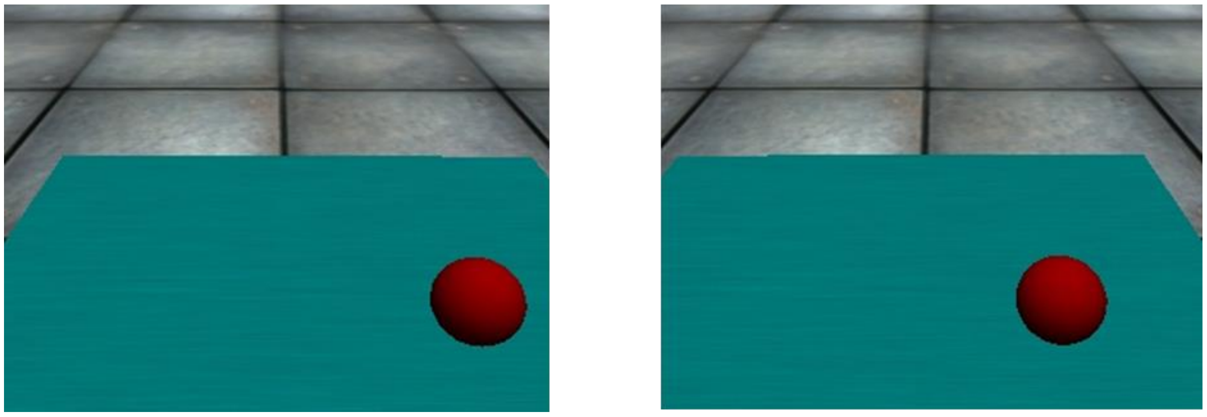


Figure 6: Raw images of a red ball on the table

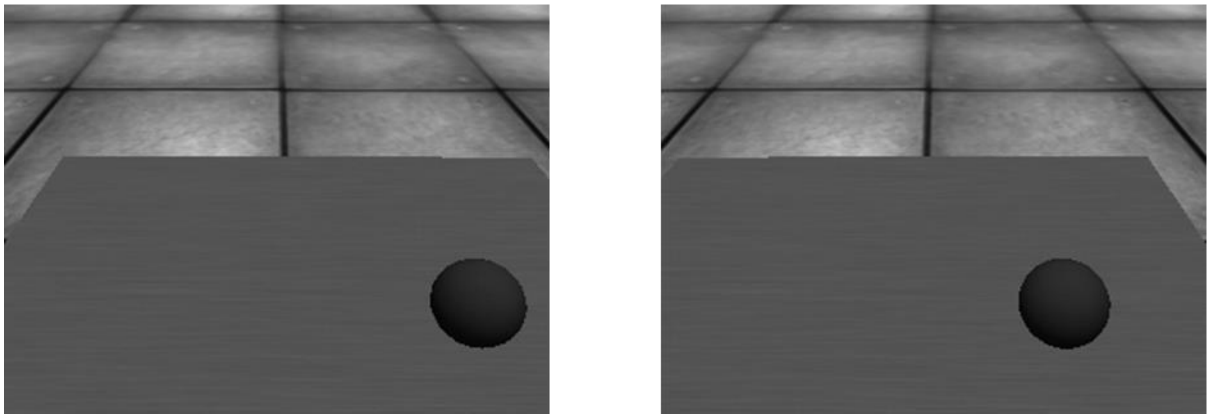


Figure 7: Grayscale images by calculating relative red intensities

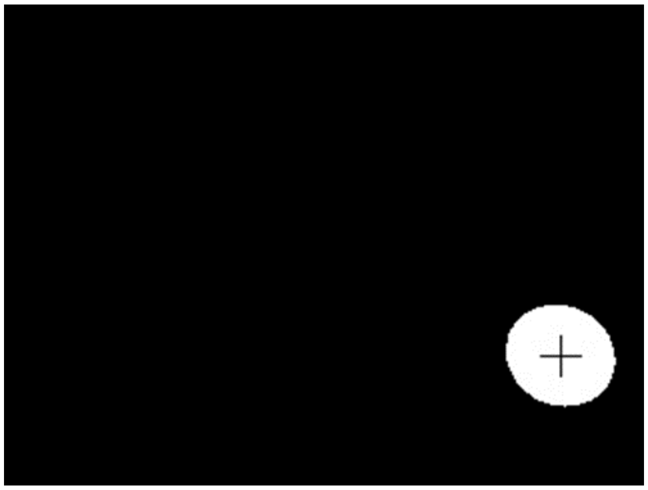


Figure 8: Threshold image with moment marked by crosshair

If such a point is detected in both left and right camera, we adjust eye orientation to fixate on the point. With the point in the center of both image, it is easier to measure the depth in next module. The position of the robot after focusing on the object is shown in figure 9.

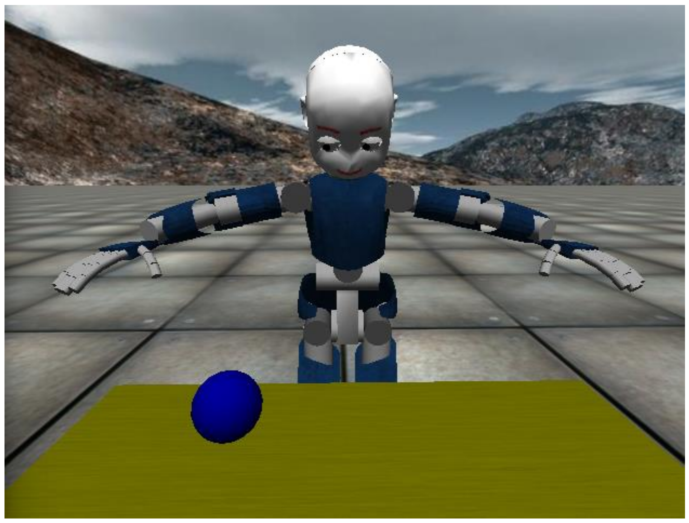


Figure 9: The iCub robot focusing on a ball in simulation

Besides the location, we would also like to know the shape of the object. Since we are expecting uniform shaped objects, we need to know the dimensions, i.e. the diameter for a ball or a cylinder. For this purpose, we select the leftmost point and rightmost point on the contour to roughly estimate the size in pixel values. Later with calculated depth information we can transform this value into distance in the world frame.

3.3.3 Stereo Vision Depth Map and Spatial Transformation

This module estimates the 3D position of target point obtained in the last module by first estimating the depth with a disparity map, and then converting depth into 3D coordinates through the homogeneous transform matrix from camera frame into root frame.

With the assumption that the target 3D point has already been fixated on, we grab frames from both left and right cameras. We then rectify images such that epi-polar lines are aligned, as shown in figure 10. This preprocessing saves noticeable computation time for the next step where we scan the image line by line, looking for matching image features. There are multiple algorithms proposed to find matching features. Here we use a simple block matching with the sum of absolute difference (SAD). The concerned disparity would simply be the horizontal distance between centers of the same feature box. The resulting disparity image is shown in figure 11 with the disparity as a grayscale value.

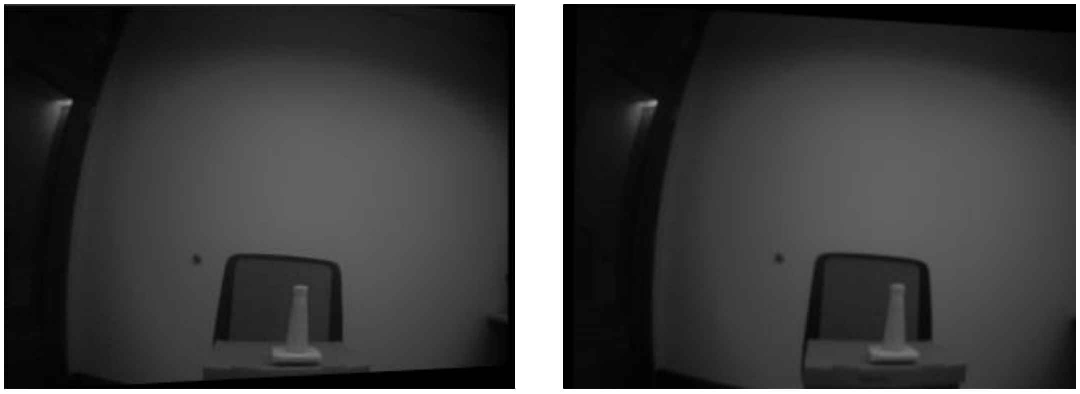


Figure 10: Grayscale rectified images from both cameras

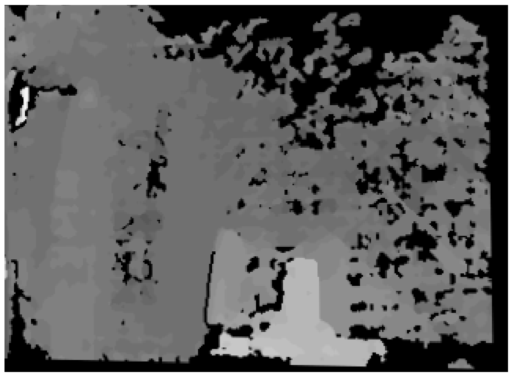


Figure 11: Disparity image using disparity as grayscale value

The images in figures 10 and 11 show an object on a chair in the lab environment, where we keep lighting low to avoid shadows errors. The disparity map succeeded in indicating the object's shape and position, even though there exist a substantial amount of noise. The noise is believed to come from environment shadows and calibration errors.

3.3.4 Hand Tracking with Optical Flow

This module tracks the hand movements to adjust the reaching control to position the hand close enough to the object. The algorithm – Lucas-Kanade method – was developed by Bruce D. Lucas and Takeo Kanade [15] and implemented in OpenCV to track specific points in an image across multiple frames.

First, we decide which features to track in a frame. A lot of well-known algorithms have been developed for this purpose. In this project, we simply use the Shi-Tomasi corner detector [16] to extract a set of corners on each frame.

The Lucas-Kanade method is based on several assumptions, namely brightness constancy, temporal persistence, and spatial coherence. It associates the change of intensity over time with the change of intensity over locations. Consider a point's neighborhood, $q_1, q_2 \dots q_n$. Let I_x and I_y represents partial derivative of intensity with respect to x and y coordinates respectively, and V_x and V_y for partial derivative of speed with respect to x and y coordinates. Then the local image flow must satisfy

$$I_x(q_1) \cdot V_x + I_y(q_1) \cdot V_y = -I_t(q_1)$$

$$\vdots$$

$$I_x(q_n) \cdot V_x + I_y(q_n) \cdot V_y = -I_t(q_n)$$

In the Lucas-Kanade method, a 3×3 neighborhood is selected, which gives nine equations with two unknowns, $\frac{dx}{dt}$ and $\frac{dy}{dt}$. This overdetermined system is usually solved by least square fits. Figure 12 shows the resulting vectors detected during the reaching process by running the Lucas-Kanade method.

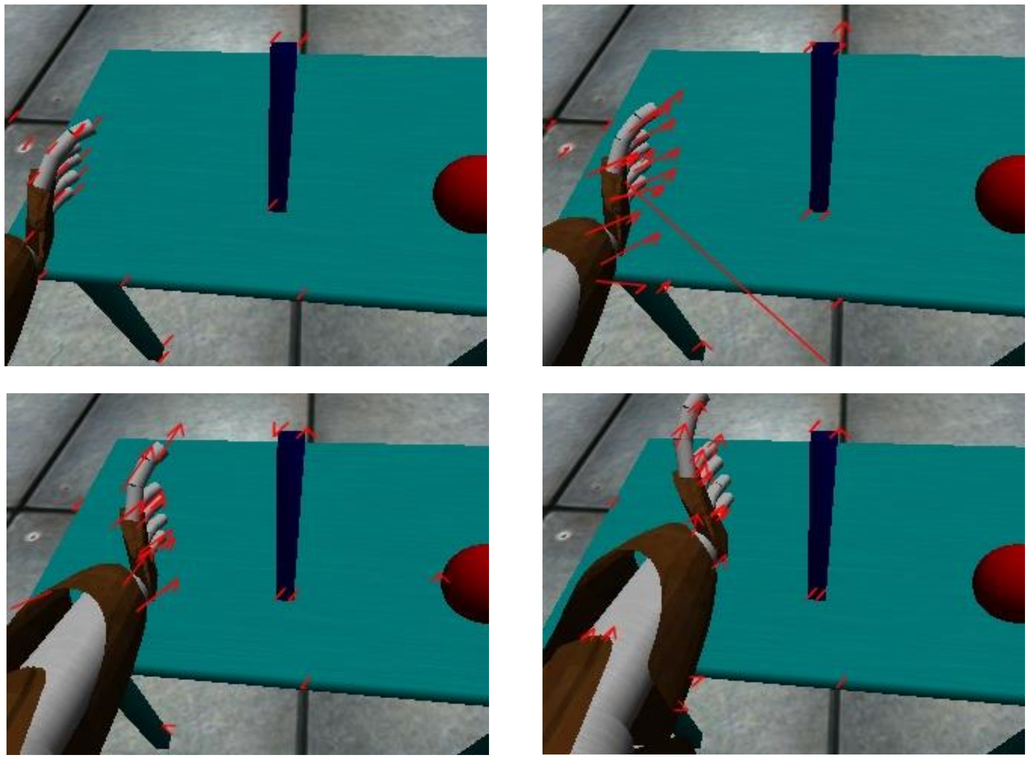


Figure 12: Optical flow vectors on hand movement during reaching process

However, this simple Lucas-Kanade method also includes the effect on the image from the head movement, which is inevitable during the process of reaching. Reaching requires not only control of the arm, but also control of the torso, and the head must move accordingly to compensate for the torso movement. It is worth pointing out that those error vectors from self-movement tend to be small compared with the hand's movement. Hence a solution of pyramids is proposed. By applying the Lucas-Kanade method for multiple times, we can remove small movements and reduce large movements to smaller ones.

After we extract the hand's movement vector from the optical flow, the module calculates the minimum difference between these moving features and the contours of the object, giving an estimation of the hand’s distance vector from the ideal position. With the object’s depth and current end-effector position known, we calculate the new 3D coordinates for the end-effector by using the camera’s geometry.

4. Conclusion and Future Work

During the experiment running on the iCub simulator, we have encountered several problems.

1. The iCub Cartesian control based on the minimum-jerk model produces a non-linear trajectory with its original intention of mimicking human behavior. However, in the case of reaching, it is possible that iCub's arm or hand collides with some other physical parts.
2. Without a priori knowledge of table height, it is difficult to even try to grasp some objects on the table, unless the objects have a considerable height. Because of errors introduced in both the stereo disparity part and motor control part, the end-effector often collides with the table and loses control due to lack of feedback sensors.
3. The usage of color segmentation to detect objects does offer faster computation. However, it has limitations. For example, the algorithm can only be used in a lab environment where we know no other objects of similar color exist. In the iCub simulation, the robot mistakenly thought the sky was an object and tried to grasp it.
4. The hard-coded hand sequences are not adaptable and can only be used to grasp certain objects like a cube and a soda can.

Notice that this application is a naïve model of reaching and grasping with all necessary components. Each module has its own space of improvement.

1) Motor control and kinematics: The current approach assumes a priori knowledge and perfect execution in the real world. However, a lot of errors might occur and hence an autonomous learning method would be more practical and perform better.
2) Object recognition: The current approach assumes uniform shape, simple environment and strong color difference. However in the real world, the environment and object are usually

much more complicated, leading to a failure of recognition. This problem is researched extensively in computer vision with many real-time solutions having been proposed.

3) Stereo-vision depth map: The current approach uses the simplest model. The advanced block selection algorithm might lead to a large decrease in computation time; using a better block comparing algorithm has the potential of more accurate result.

References

[1] H. Moravec, *Mind Children: The Future of Robot and Human Intelligence*, Cambridge: Harvard University Press, 1988.

[2] R. Clifton, M. Muir and D. Ashmead, "Is visually guided reaching in early infancy a myth?," *Child Dev.,* vol. 64, no. 4, pp. 1099-110, 1993.

[3] D. Ashmead, M. McCarty, L. Lucas and M. Belvedere, "Visual guidance in infants' reaching toward suddenly displaced targets," *Child Dev.,* vol. 64, no. 4, pp. 1111-27, 1993.

[4] J. M. Hollerbach and C. W. Wampler, "The calibration index and taxonomy for robot kinematic calibration methods," *International Journal of Robotics Research,* vol. 15, no. 6, pp. 573-591, 1996.

[5] R. Tsai and R. Lenz, "Real time versatile robotics hand/eye calibration using 3D machine vision," in *International Conference on Robotics and Automation*, 1988.

[6] G. Metta, G. Sandini, D. Vernon, L. Natale and F. Nori, "The iCub humanoid robot: An open platform for research in embodied cognition," in *Proceedings of the 8th workshop on performance metrics for intelligent systems*, 2008.

[7] G. Metta, P. Fitzpatrick and L. Natale, "Yarp: Yet another robot platform," *International Journal of Advanced Robotic Systems,* vol. 3, no. 1, p. 8, 2006.

[8] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale and F. Nori, "An open-source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, 2008.

[9] "ICubForwardKinematics," 27 February 2014. [Online]. Available: http://wiki.icub.org/wiki/ICubForwardKinematics. [Accessed 21 April 2017].

[10] A. Roncone, U. Pattacini, G. Metta and L. Natale, "A Cartesian 6-DoF gaze controller for humanoid robots," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, 2016.

[11] U. Pattacini, F. Nori, L. Natale, G. Metta and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*, 2010.

[12] A. Wächter and L. T. Biegler, "On the implementation of a Primal-Dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming,* vol. 106, no. 1, pp. 25-57, 2006.

[13] D. Bullock and S. Grossberg, "Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation," *Psychological Review,* vol. 95, no. 1, pp. 49-90, 1988.

[14] A. S. Deo and I. D. Walker, "Robot subtask performance with singularity robustness using optimal damped least-squares," in *IEEE International Conference on Robotics and Automation*, 1992.

[15] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981.

[16] J. Shi and C. Tomasi, "Good features to track," in *9th IEEE Conference on Computer Vision and Pattern Recognition*, Springer, 1994.