6.866 - Machine Vision

Class Project

Massachusetts Institute of Technology
Fall 2011/2012

Pedro Santana (psantana@mit.edu), MIT ID 926275295

December 13^{th} 2011

Efficient Real-Time 3D Trajectory Reconstruction Through Visual Tracking

1 Introduction

The future of robotics moves towards releasing robots from mostly repetitive tasks and bringing them to close and fluid interaction with humans. In this context, the MERS group at CSAIL explores a futuristic robotic manufacturing environment in which humans and robots collaborate together as peers within a semi-structured factory and leverage their different abilities in order to perform elaborate tasks robustly and efficiently. In this scenario, one key capability is to be able to track objects and agents in real-time while they move in the environment so as to plan actions accordingly and detect failures as soon as possible.

Due to the precision and current availability of laser range finders and similar sensors for reasonably low prices, point cloud-based methods for 3D environment reconstruction have become increasingly popular during the past couple of years. However, treating the overwhelming amount of information provided by these sensors is still an issue in terms of computing, specially when the processing hardware is not particularly powerful. In order to overcome this problem, this project proposes[1] an efficient method for 3D trajectory reconstruction based on the real-time tracking of a very limited Region of Interest (ROI) around the image of an object, which is in turn used to limit the number of samples considered in subsequent point cloud-based algorithms. The ROI's dimensions are kept small by anticipating its motion on the image plane using optical flow information within the region where the object is most likely to be. The theory behind this method is described in Section 3, followed by the real-world experimental results in Section 4 that were obtained on the robotic test bed described in the next section. Finally, Section 5 presents the concluding remarks and future work avenues.

2 System Overview

The autonomous manufacturing test bed currently available in the MERS group at CSAIL consists of a PR2 robot and two Barrett Whole Arm Manipulators (henceforth referred only as WAM arms) equipped with three-fingered Barrett Hands, besides a number of Microsoft Kinect sensors used for gathering additional information about the environment (see Figure 1). The collaborative manufacturing task currently undertaken by the robots consists of generating different kinds of assemblies involving colored foam balls about 20 cm in diameter. In this context, this project focuses its attention on a subsystem of the robotic manufacturing test bed, namely the monitoring module that keeps track of the actions performed during shared manipulation tasks in order to detect faults and trigger recovery routines as early as possible.

One problem involved in the monitoring of manufacturing tasks is to track the position of manipulated objects in real-time and check whether they are following their desired trajectories or have been dropped or taken by some other agent (human, possibly) that interfered with the task.

[1] I would like to make it clear that I am not stating that the visual tracking method presented here configures a novel scientific contribution. I am just not aware of any similar strategy, which might be due to my limited knowledge about the field of computer vision and point cloud processing methods.

(a) Barrett Whole Arm Manipulator (WAM).

(b) Willow Garage PR2.

(c) Microsoft Kinect.

Figure 1: Components of the robotic manufacturing test bed available at the MERS group (not drawn to scale).

In this situation, one cannot rely on the assumption that, once an object has been grasped by the robot, it will remain physically attached to its end effector ("hand") during the whole movement. In this project, we seek to efficiently reconstruct the 3D trajectory of a foam ball manipulated by a WAM arm by using data from a Kinect sensor mounted on the floor. The resulting algorithms will then be adapted to the PR2 sensor architecture in order to enable task coordination between these two different kinds of robots.

The whole manufacturing test bed runs on top of Willow Garage's Robot Operating System (ROS, `http://www.ros.org`), which offers a number of tools for integrating different robots and sensors in a data-exchange network, in addition to libraries for performing the most common tasks found in robotics projects. Figure 2 illustrates how the designed visual tracking system[2] is organized within the ROS framework. As can be seen in Figure 2, the *Visual Tracker* node takes in the camera image from the Kinect sensing the manufacturing test bed and outputs a Region of Interest (ROI), which is a very restrict subset of the image plane where objects should be looked for. This ROI, which is obtained through the steps detailed in Section 3, is then sent to the *3D Tracker* node, where it is used as a tool for improving computational performance by reducing the number of samples from the point cloud considered when extracting the spheres' spatial coordinates. Next, the 3D information is used by the *WAM arm subsystem* module (it comprises several different nodes) in order to plan the trajectory for grasping and executing the motion. Section 3 explains the internal details of the *Visual Tracker* node and how it interacts with the *3D Tracker*. Experimental results involving all components shown in Figure 2 are presented in Section 4.

3 Methodology

As described before in Section 2, the only objects currently handled by the robots in this initial stage of development of the autonomous manufacturing test bed are colored foam balls. The reasons

[2]Only the *Visual Tracker* and the *3D Tracker* nodes were developed during this project

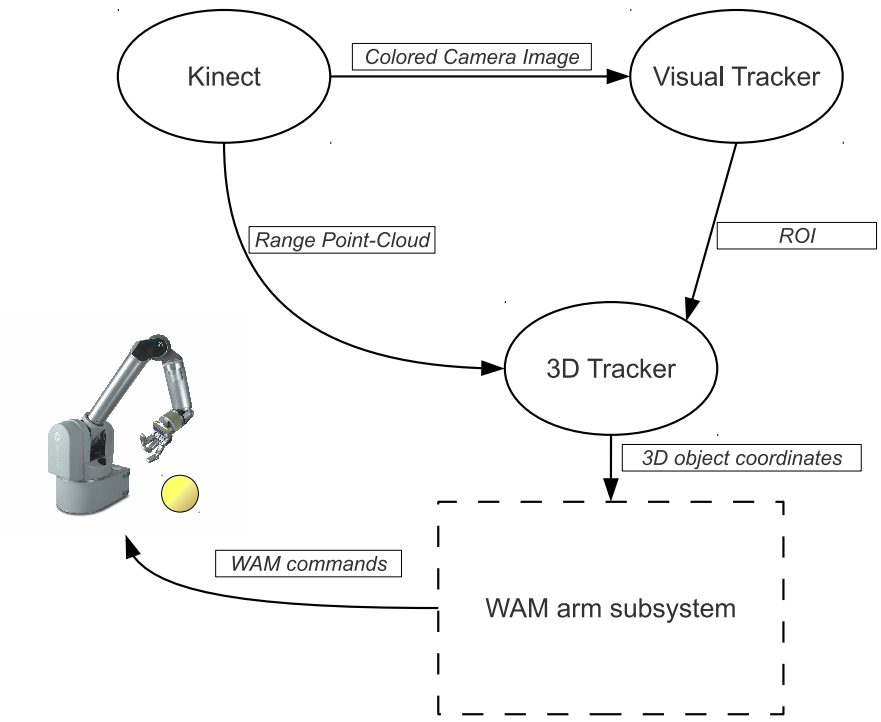

Figure 2: Visual tracking system in ROS. Ovals denote ROS nodes, while text surrounded by solid boxes represent exchanged data. The *WAM arm subsystem* box abstracts a set of components that form the WAM arms' interface with ROS.

behind this choice are twofold: first of all, both the PR2 and the WAM arms have limited dexterity at their end effectors, which limits the geometrical complexity of objects that can be manipulated. Second, the previous version of the test bed did not comprise any sensing capability other than the WAM arms' encoders, which forced all objects to be placed at pre-specified positions and orientations so the robots could reach them. Therefore, choosing to manipulate balls greatly simplifies the experimental setups involving collaborative task execution, since their spherical symmetry eliminates the degrees of freedom associated to orientation. In this context, the problem of measuring object positions in the image space boils down to the detection of circles of different radii. One efficient way of accomplishing that is to use the Hough transform applied to circle detection using edge gradient information [1], whose implementation details can be found in [2] and are briefly explained in Section 3.1.

The aforementioned circle detector does provide useful information about the positions of different circles in the image, provided that there is enough contrast so edges can be correctly extracted and used by the Hough circle transform (Section 4 presents a possible solution in the case when the contrast assumption is violated). However, its correct operation depends on fine tuning a number of detection thresholds that are strongly dependent on the environment, making it virtually impossible to guarantee correct object detection at all times. Moreover, the foam balls are partially occluded when being actively manipulated and carried around by the robotic WAM arm, further hindering the circle detector's performance. Therefore, it is necessary to deal with the problem of where to look for the ball in the image once it has not been correctly detected.

One simple way to recover from losing track of an object is to look for it in the whole image. This strategy, however, is not computationally efficient and, most important, completely ignores the readily available information of where the object was right before it was lost. In the light of these two observations, it makes sense to look for the object within a ROI, as explained in Section 3.2. Choosing the size of the ROI configures a trade-off between computational complexity and the stability of the sphere detector. Smaller windows tend to reduce the time required for image processing and allow for significant performance improvements of the point cloud-based 3D position extraction algorithms mentioned in Section 3.4. Nevertheless, it makes the circle detector more "brittle", since it is likely that the sphere might move outside of a very tight ROI between two consecutive images. On the other hand, increasing the ROI's size will help find the sphere's new position, but at the expense of increased computational effort. In this context, Section 3.2 explains how the sphere's motion in the image is anticipated by means of the measured optical flow around its most likely position. In Section 3.3, this velocity information, along with the position estimates from Section 3.1, are then used by a pair of Kalman filters that dynamically resize and move the ROI in order to keep it around the sphere's most likely position for a given level of confidence. Finally, once a circle has been correctly detected within this dynamic ROI, the resulting frame is used by a point cloud processing algorithm that extracts the ball's radius and center coordinates in 3D space.

3.1 Measuring image position

The algorithm presented in [1] and implemented in the OpenCV library is a more computationally efficient version of the original circle detection method introduced in [3], where the Hough transform [4] is directly applied to image processing. While [3] detects circles on the image plane by means of a three-dimensional accumulator volume in parameter space, where the axes are the x and y positions of the circles' centers and their radii r, the authors in [1] propose an algorithm with considerably reduced memory requirements based on an edge-detection pre-processing step followed by gradient information extraction, a procedure that is schematically depicted in Figure 3. In the particular implementation available in the OpenCV library, the Canny edge detector [5] is used. Once edges are extracted, the local gradient for every non-zero pixel in the resulting binary image is calculated, as represented by ∇E in Figure 3, and the weight of all points in the direction of ∇E within a minimum (d_{min}) and a maximum (d_{max}) distances is incremented. The centers of the circles will be the points with maximum weights after this operation has been carried out for all edge points. For every circle center selected in the previous step, the radius is chosen as the distance that maximizes the number of edge points in support of that center.

3.2 Measuring image velocity

Consider the ROI shown in Figure 4 and the auxiliary windows W_1 and W_2. Our goal is to obtain an estimate of the circle's motion on the image plane by means of the optical flow vectors calculated for each one of its pixels using the Horn-Schunk method [6]. Assuming that the sphere's motion can be reasonably approximated by an uniform translation, the resulting velocity vector can be obtained by taking the average of the horizontal (u) and vertical (v) optical flow measurements within W_2, which is tightly drawn around the tracked circle in order to correctly approximate its

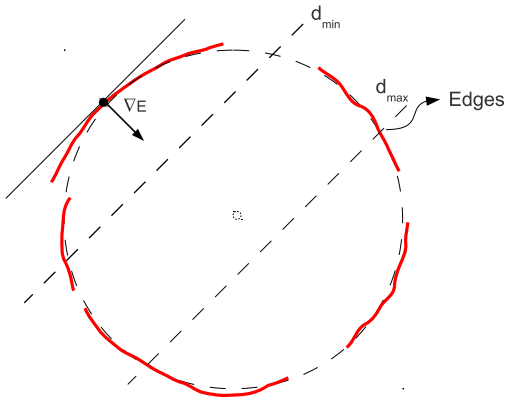

Figure 3: Diagram illustrating the use of the Hough transform for circle detection.

motion in the image. What is the role of W_1, then? If the optical flow measurements were to be extracted solely from W_2, they would suffer from the *aperture problem*, given that only a very narrow window would be used to detect the motion of brightness patterns on the image. In order to overcome this issue, it is within W_1, which is a strict superset of W_2, that the optical flow iterations are performed. The reason why optical flow is not calculated within the whole ROI is that the latter can be resized by the Kalman filters and occupy the whole image for reasons that will become apparent in the next section. Besides the considerable computational cost of calculating the optical flow vectors for every single pixel, it would not make sense to assume that the sphere's motion could be correctly approximated by the average optical flow obtained for the whole image frame. Figure 5 shows the frames from Figure 4 superimposed on a real image from the robotic test bed explained in Section 2.

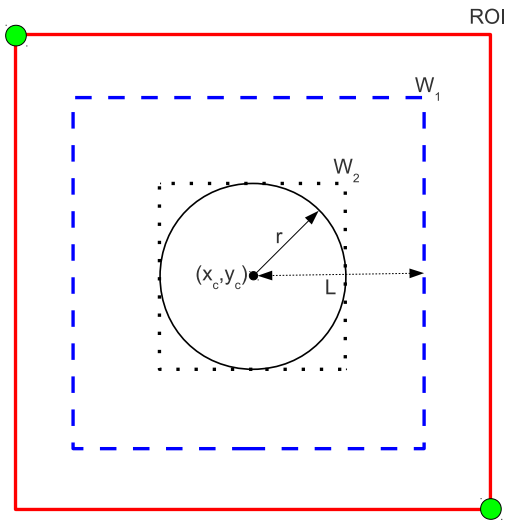

Figure 4: Region of Interest (ROI) and auxiliary windows for calculating optical flow.

Two reasons explain why the Horn-Schunk method was chosen over the currently more popular Lucas-Kanade [7] algorithm. First of all, one of this project's goals was to demonstrate the author's ability to use the concepts learned during the course in order to solve real-world problems. Second,

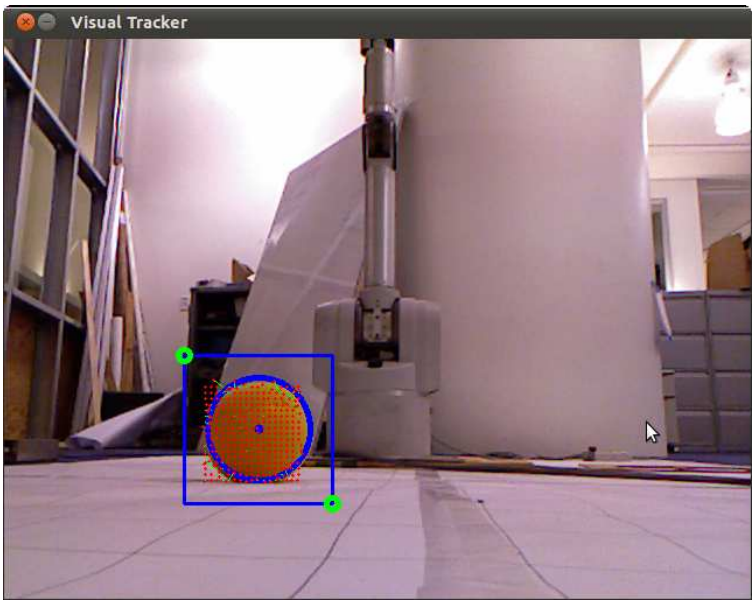

Figure 5: ROI, detected circle, and W_2 window superimposed on a real image of the robotic test bed. The optical flow vectors are shown as green vectors with rounded red tips.

the Lucas-Kanade algorithm depends on the specification of good visual features for estimating optical flow, which are not easily obtained from a tight image window surrounding a uniformly-colored foam ball.

3.3 Kalman filtering in image space

This section details how the position information from Section 3.1 and the velocity information from Section 3.2 are fused together by means of a pair of Kalman filters [8] in order estimate the sphere's position on the image plane and also determine the size of the ROI. Defining

$$x_k = \begin{bmatrix} x_{c,k} & y_{c,k} & u_{c,k} & v_{c,k} & r_k \end{bmatrix}^T \tag{1}$$

as the state vector at the k-th sample instant, where r_k is the circle's radius and $x_{c,k}$, $y_{c,k}$, $u_{c,k}$, and $v_{c,k}$ are, respectively, the X and Y coordinates of the circle's center and their corresponding velocities, we have the dynamic model

$$\begin{aligned} x_k &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_{k-1} + w_k = Ax_{k-1} + w_k, \\ w_k &\sim N(0, Q_k), \quad Q_k = \mathrm{diag}(\sigma_{x_c}^2, \sigma_{y_c}^2, \sigma_{u_c}^2, \sigma_{v_c}^2, \sigma_r^2). \end{aligned} \tag{2}$$

Concerning the measurement model, two cases are possible. If the Hough transform from Section 3.1 was able to detect a circle within the ROI, we have the position and radius measurements

$y_{p,k} = [\bar{x}_{c,k}, \bar{y}_{c,k}, \bar{r}_k]^T$ along with the optical flow measurements $y_{v,k} = [\bar{u}_{c,k}, \bar{v}_{c,k}]^T$. If not, we only have $y_{v,k}$, giving rise to the measurement models

$$y_k = \begin{bmatrix} \bar{x}_{c,k} \\ \bar{y}_{c,k} \\ \bar{u}_{c,k} \\ \bar{v}_{c,k} \\ \bar{r}_k \end{bmatrix} = \mathbb{I} x_k + v_k^{full},$$
$$v_k^{full} \sim N(0, R_k^{full}), \;\; R_k^{full} = \text{diag}(\sigma^2_{\bar{x}_c}, \sigma^2_{\bar{y}_c}, \sigma^2_{\bar{u}_c}, \sigma^2_{\bar{v}_c}, \sigma^2_{\bar{r}}), \tag{3}$$

and

$$y_k = y_{v,k} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} x_k + v_k^{vel},$$
$$v_k^{vel} \sim N(0, R_k^{vel}), \;\; R_k^{vel} = \text{diag}(\sigma^2_{\bar{u}_c}, \sigma^2_{\bar{v}_c}), \tag{4}$$

where $\mathbb{I}$ is the identity matrix of appropriate dimensions. By combining a Kalman filter for (2) and (3) and another one for (2) and (4), we get $\hat{x}_k$ and $\hat{P}_k$, respectively the estimate of (1) and its associated covariance matrix.

The last step is to make the ROI's dimensions in Figure 4 proportional to the uncertainty associated with the estimates $\hat{x}_{c,k}$ and $\hat{y}_{c,k}$, so it is automatically increased when the sphere's position cannot be precisely pinpointed. In the case of this project, the ROI's width is equal to the estimated circle radius $\hat{r}$ plus $3\sigma_{\hat{x}_c}$. Analogously, the ROI's height is equal to $\hat{r}$ plus $3\sigma_{\hat{y}_c}$. Therefore, if the visual tracker cannot find the sphere within the current ROI, it will only be able to correct its velocity measurements by means of (4), which causes the uncertainty of $\hat{x}_c$ and $\hat{y}_c$ to grow and, consequently, expand the ROI to a larger image region. In the worst case where the sphere is not detected for an extended amount of time, the ROI eventually expands to a size that comprises the whole image (the rate of expansion depends on the uncertainty associated with x_c and y_c in Q_k).

3.4 Extracting 3D position from a point cloud

The sphere's radius and 3D center coordinates are obtained from the Kinect's range sensor data using functions from the PCL library (`http://pointclouds.org/`), whose implementation details are outside the scope of this machine vision project. However, it is in this brief section that the most significant improvement to the performance of the robotic test bed described in Section 2 is found. In the previous version of the test bed's object tracking system, the complete point cloud coming from the Kinect's range sensor was processed in order to detect a sphere and reconstruct its 3D position and dimensions. This, in turn, entailed a computational cost that rendered the machine running the tracking algorithm unable to follow moving objects, since the estimates' settling time was in the order of a few seconds. The tracking system was still useful for mapping spheres scattered around the robotic arm, but could not be used for real-time motion tracking. Thus, the greatest contribution coming from the visual tracking system described in the previous sections is providing the point cloud algorithm with a small ROI around the tracked object, which defines a restricted subset of the point cloud that needs to be considered for 3D information extraction. Even though

Section 4 only brings experimental results concerning the tracking system described in this report, the differences in performance between the previous and the current tracking methods can be understood by watching the video available at the following link.

`http://people.csail.mit.edu/psantana/public/videos/2011_12_13_santana_visual_tracker_evolution.mp4`

4 Experimental Results

The first trials involving the visual tracking system described in Section 3 were carried out successfully using a grayscale version of the Kinect's camera image for both circle detection and optical flow estimation. However, the poor lighting conditions within the robotic test bed area prevented the Hough circle transform from Section 3.1 to correctly detect the sphere in the scene, since the small amount of contrast between objects impaired the edge-detection step. In order to overcome this obstacle, the solution adopted was converting the colored image coming from the Kinect into a binary image using the sphere's color as a threshold, as can be seen in Figure 6.

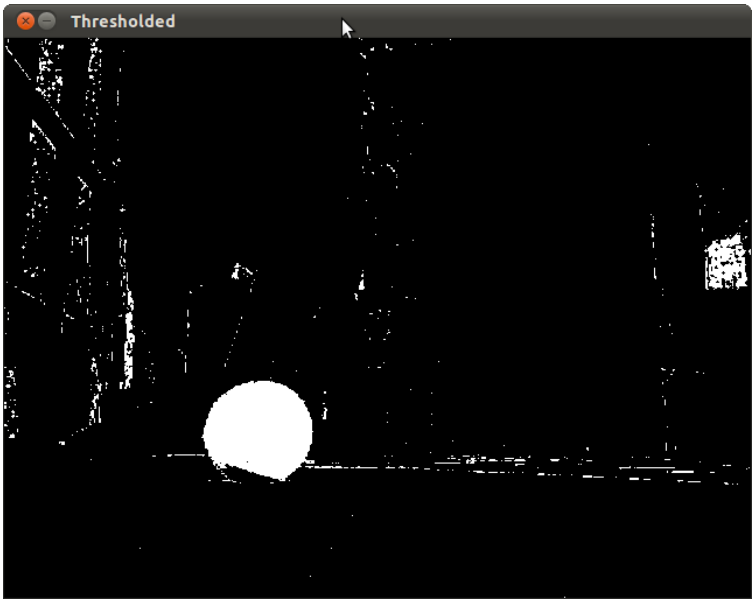

Figure 6: Binary image obtained by using the foam ball's color (yellow) as threshold. This is the same image shown in Figure 5.

Once all systems components were operating correctly, the experiments conducted were very similar to the video referenced in Section 3.4: guided by the visual system, the WAM arm picked up the ball at a random position within the grasping area and moved it to another randomly chosen destination. At all times, the WAM arm have no record of the ball's current or previous location and totally relies on the 3D position estimates coming from the visual tracking system in order to grab the ball and change its position. Once the WAM arm has the ball, it raises its end effector about 30 cm above the ground, moves on a planar trajectory trajectory until it reaches a point above the goal location, lowers the ball to the ground, and repeats the sequence. Figure 7 contains the position and velocity estimates generated by the Kalman filters tracking the circle

corresponding to the ball's image. Nine ball displacements were performed during the experiment, as can be seen by the level changes in Figure 7(a) and the peeks in Figure 7(b). It should be noticed that, since the origin for image position was located at the upper left corner of the image, the higher planar segments in Figure 7(b) actually correspond to the ball on the ground, whereas the lower peaks correspond to the ball being raised. Moreover, it should be noticed that the level changes in Figure 7(a) appear to be evenly distributed above and below $x_c \approx 300$, which is coherent with the fact that the WAM arm moved the ball to positions symmetrically distributed around the center of the 640×480 image. Another interesting point is to observe that sudden changes in Figure 7(a) correspond to peaks in Figure 7(c), since the latter correspond to horizontal velocity estimates of the circle's center. The same relationship exists between Figures 7(b) and (d).

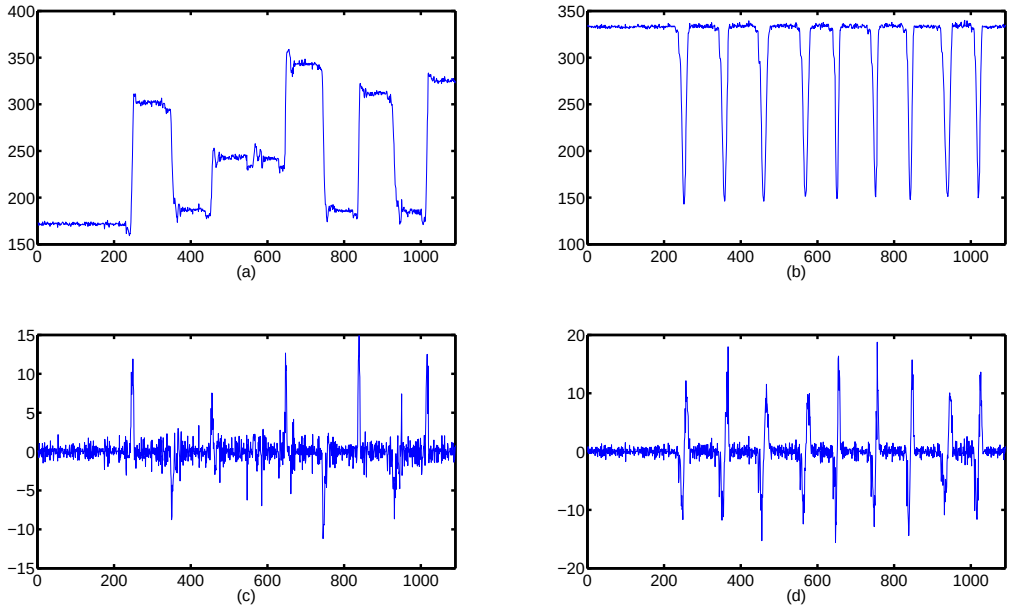

Figure 7: Kalman filter estimates on the image plane. (a) Estimates of x_c. (b) Estimates of y_c. (c) Estimates of u_c. (d) Estimates of v_c. All horizontal axes represent sample numbers, while the vertical axes are in units of pixels.

The ball's trajectory on the image plane is reconstructed in Figure 8, where once again the WAM arm's moving pattern can be clearly perceived. As in Figure 7(b), high values on the vertical axis correspond to the ball on the ground, while low values correspond to the ball being raised in the air by the robotic arm.

Finally, Figure 9 shows the reconstructed 3D trajectory of the ball for a longer set of repetitions (the experiment ran for approximately two hours). The three-dimensional plot shows the robot's perspective of the movement, i.e., the Kinect sensor is placed in the direction of decreasing values of Z. Except for a few outlier estimates, the rest of the points clearly reconstruct the Gaussian movement pattern undertaken by the WAM arm, as can be seen by the scattering of particles at ground level and the arcs generated by the WAM arm while rotating around its axis in order to move the ball to its new position. Therefore, we conclude that the designed visual tracking system

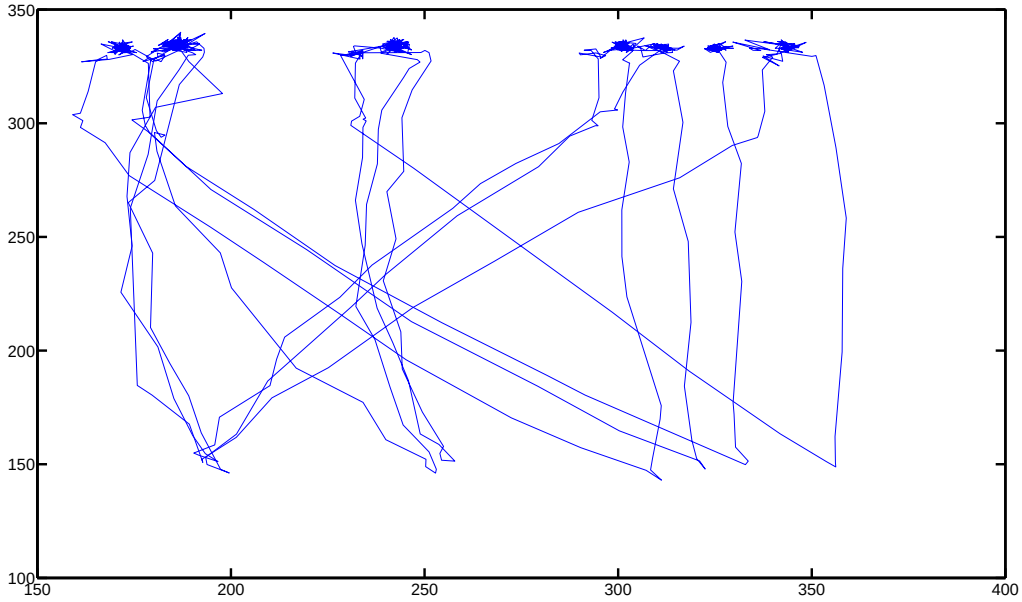

Figure 8: Reconstructed trajectory on the image plane.

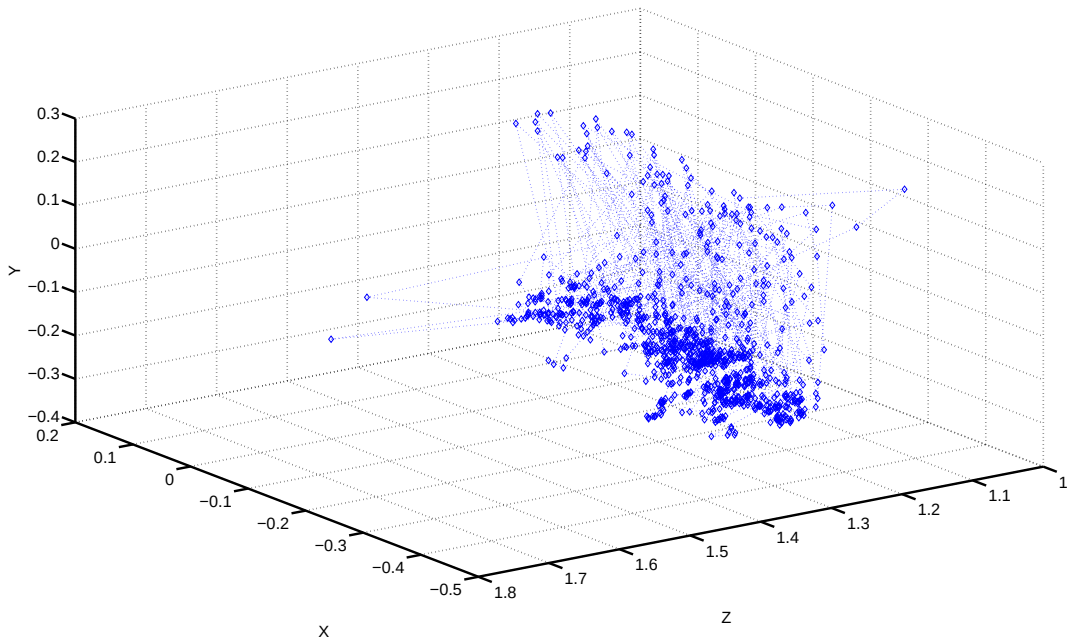

Figure 9: Reconstructed trajectory in 3D space.

is capable of being used as a task monitoring tool, since it is able to reconstruct in real-time the ball's 3D trajectory, which can in turn be compared to a nominal desired path in order to detect

failures.

5 Conclusions and future work

This project showed the viability of applying visual tracking as a means to increase the efficiency of recovering an object's 3D position from a range point cloud. By using optical flow information in order to anticipate movement, the visual tracker was able to keep a very tight ROI around the object being tracked, thus significantly reducing the number of points considered in the point cloud and their associated computational cost. The algorithm was successfully implemented in a real robotic test bed and had its usefulness as a real-time monitoring tool verified by a number of experimental graphs and videos.

Future work within the context of this project will extend the object recognition routines to a richer set of shapes other than just colored foam balls. For this purpose, the Extended Gaussian Image representation learned in class will be readily useful, given that the next step in the robotic manufacturing test bed will be to deal with three-dimensional rectangular blocks. Another important extension to this project will be to replicate the methods presented here to several moving objects at the same time. Since all the algorithms described and implemented could be easily scaled to any number of objects, the most important contribution would be to give the system the ability to differentiate between several similar objects during runtime.

Bibliography

[1] Carolyn Kimme and Dana Ballard. Finding circles by an array of accumulators. *Communications of the ACM*, 18(2):120–122, February 1975.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[3] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, January 1972.

[4] P.V.C. Hough and B.W. Powell. A method for faster analysis of bubble chamber photographs. *Il Nuovo Cimento (1955-1965)*, 18(6):1184–1191, 1960.

[5] J Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–98, June 1986.

[6] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, August 1981.

[7] BD Lucas and T Kanade. An iterative image registration technique with an application to stereo vision. *Image Understanding Workshop*, (x):674–679, 1981.

[8] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications, INC, New York, 1970.