CMPUT-615: Project Report

3D Model Based Tracking

Islam Ali
Department of Computing Science
University of Alberta
iaali@ualberta.ca

Junaid Ahmad
Department of Computing Science
University of Alberta
jahmad@ualberta.ca

Abstract

Abstract Goes Here ...

1 Problem Definition

Object tracking is one of the main corner stones in a wide spectrum of robotics-related applications. This is true for both regular 2D tracking, and 3D object tracking which allow robots to perform more tasks. In this project, we study the problem of 3D Model-Based Tracking, which is tracking 3D object with the knowledge of its 3D structure in prior. We study one of the main systems that most of modern trackers are based on, which is RAPiD [1]. We focus on the study of the system structure and how the task can be achieved, pinpointing main problems with RAPiD tracking and proposing solutions when applicable. The system will be tested on synthetic data as a starting point with the purpose of having it tested with real-world data in the near future. The main performance metric is the tracking accuracy for the time being, with plans to focus on memory and complexity aspects as the system evolves in the future.

2 Literature Review

2.1 Taxonomy of 3D Tracking Systems

3D tracking systems can differ in a wide range of aspects and can have similarities as well. However, in order to have a high-level picture on how to classify a 3D tracking system, we needed to develop a taxonomy of such systems based on our readings from the literature. Specially, the work done in [2] which highlights a wide range of systems and give a bird's eye view of their structure and principle of operation. This can also open doors to new combinations of systems and can result in more generic system that are better performing. The following diagram was developed with the for the aforementioned purpose. Such a taxonomy can provide some pointers on how to make fair comparisons between algorithms as well and how to generalize them in order to work in a wider spectrum of scenarios.

Figure 1: Taxonomy of 3D Tracking System as Observed From the Literature

2.2 Performance Evaluation

The three corner stones of evaluation of any computer-based system are:

1. Accuracy.
2. Time Complexity.
3. Memory Requirements.

The main metric that is considered the main objective is accuracy [3] [4] [5]. However, we were not able to find a unified definition of accuracy when coming across tracking tasks. This stems from the fact that accuracy is hard to measure and compare across different systems where their structure differ and also their experimental setup [2]. Another side objective is to enhance the time complexity of the system [2] [6], and possibly enable it to run in real-time, similar to the main objective set in RAPiD [1], where the real-time performance was put first, resulting in a room for accuracy enhancement for those who come after it. It worth mentioning in this context that this metric is the easiest to measure, and also to decompose w.r.t. system component. On the other hand, we did not encounter anyone reporting the memory requirements for their system, although the great importance we have for this metric in systems deployed on limited resources platforms such as drones and small UAVs.
On the other side of the spectrum, other metrics are reported in order give a better sense of the system performance or the enhancement achieved. However, in many cases the definition is loose and is not provided along the work description. One of these reported metrics is the system convergence [3], which is defined by the time the system takes for its least squares step to converge to an acceptable residual error. This becomes important in determination of the system response time, and, in fact, is considered highly related to the time complexity metric mentioned above.

3 System Design

As mentioned earlier, the system is based on the famous 3D Model Based Real-time tracking developed by Harris et. al.[1]. The following is a block diagram of the full system illustrating the different components and their interaction with each other.

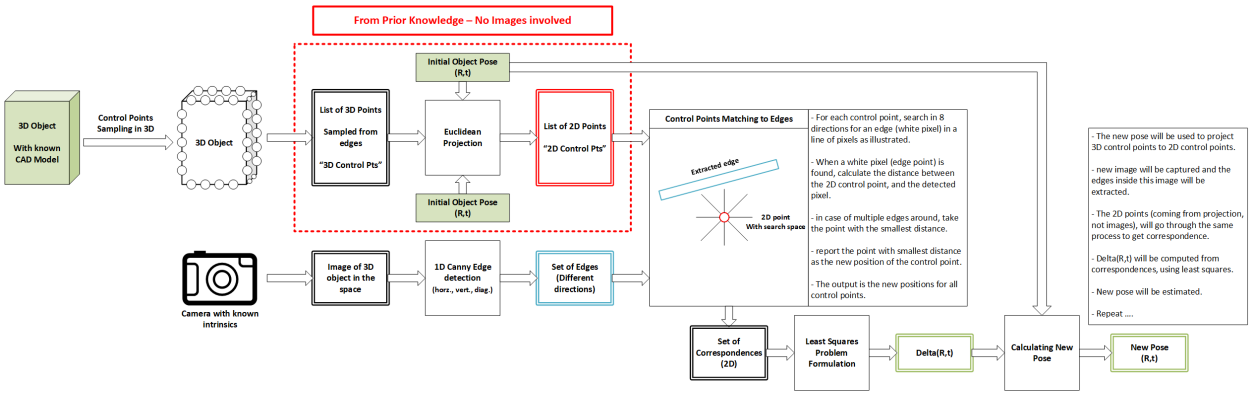


Figure 2: Full System Block Diagram

From an abstract point of view, the idea for RAPiD[1] is based on tracking high-contrast edges which we call control edges. However, edges are not dealt with directly but through a sampling step where these edges are samples to a number of points along each edge. These points are then called control points and they act as the basis for next steps. Next, we should have an estimate of the 3D object pose, which at the beginning can come from calibration, and as we progress can have the value for the previous estimated pose at the previous time step. The process is mainly project the 3D control points onto the image using the object pose and camera calibration parameters. Then search for the closest edge to each one of these points. After finding the corresponding 2D points, we construct a least squares equation in order to infer the values

for the motion in terms of translation and rotation. The pose of the object is then updated based on the inferred motion values. In the rest of this section, we provide the detailed design of the system as well as the mathematical representation of the motion screw and how it was obtained. We also show how the least squares formulation in our case was a bit different from the original least squares format provided in [1] due to the fact that the camera in our case does not align with neither the object frame nor with the world frame.
From a detailed point of view, the system goes through two different states which are:

1. Bootstrapping
2. Motion estimation loop

3.1 Bootstrapping

Bootstrapping is considered the system initialization phase and has the objective of having an initial state for the following items:

1. Camera calibration parameters (both intrinsic and extrinsic).
2. 3D model for the object to be tracked with proper representation.
3. 6D pose of the object w.r.t. the world coordinate frame.

Due to having a synthetic data set, we needed to extract these parameters from the blender model. In real-world setup, the system should undergo formal calibration procedure, formal measurements of pose of camera in the world frame, and proper measurement of both the object dimensions and complete pose.

3.2 Motion Estimation Loop

One the system is initialized and the aforementioned information are available and are accurate enough, the system tries to estimate the pose of the 3D object in the world coordinate frame using the following algorithm:

Algorithm 1: RAPiD 3D Model-Based Tracking

```
Result: 6D object pose
Determine the initial 6D pose of the object;
Perform camera calibration (intrinsic and extrinsic);
3D control points = sampling (high contrast 3D edges) ;
while Image Sequence Not End do
    2D control points = Projection of 3D points using pose info and camera calibration;
    sobel image = extract image edges using Sobel operator;
    for Each control point do
        l = length from 2D projected point to closest edge;
        corresponding 2D point = point with shortest length to edge;
        Add corresponding point to correspondence vector;
    end
    Filter correspondences and keep only strong ones;
    Δp = least squares on correspondences ;
    Update object pose: p = p + Δp;
end
```

3.2.1 Control Points Projection

Having a set of 3D control points sampled from high contrast object edges, we are able to project these 3D points onto 2D image plane with the knowledge of the camera calibration matrices and the object pose in

the world coordinate system. The projection is applied to the homogeneous form of the 3D points and takes the following form:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = [K][Rt]_{camera}[Rt]_{object} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

Then the 2D projection of each point is retained by normalizing the result as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \end{bmatrix} \tag{2}$$

3.2.2 Edge Detection and Extraction

Extraction of image edges is crucial as we need to find correspondences for projected points in each image. These correspondences will allow us to deduce the motion screw of the 3D object. A wide range of methods are available for edge detection and extraction. In our case, we tried both Canny edge detector [7] and Sobel-Feldman operator [8]. In our case and due to the lighting conditions of the synthetic scene, Sobel-Feldman was selected as our edge detector of choice.
Sobel-Feldman operator[8] is a gradient based edge detection method that computes the first order derivative separately for each of the image dimensions x and y. The derivative is calculated by performing convolution of the image with a two 3x3 kernels, one for each dimension, which computes approximation for the gradient. This gradient is able to highlight high frequencies inside images which are usually produced by edges. The Sobel-Feldman kernels are as follows:

$$kernel_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad kernel_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

To suppress weak edges and emphasize strong ones, image thresholding is applied to the result of the Sobel-Feldman convolution and the threshold value is tuned to be $I_{threshold} = 127$ which remove some of the shadows and emphasize real images.

3.2.3 Corresponding Edges Extraction

The original RAPiD [1] assumes that, if the motion done by the object being tracked is small enough, the control edge (projected), and the image edge (extracted from image) should be parallel to each other. Therefore, a 1D search can be done in the direction perpendicular to the control edge from the control point for image intensities. Once an edge pixel is detected, it is declared as the correspondence for this control point. For simplicity, and to cover multiple motion profiles of the object, the search is done in 8 directions around each control point which are: horizontal, vertical, and diagonal.

3.2.4 Correspondences Filtering

In order to avoid having noise in the estimation in the least squares step, correspondences must be filtered to ensure consistency. The simplest method is to filter based on the euclidean distance to nearest edge. In the current setup, this threshold is set to be $d_{threshold} = 50$, meaning that, if the closest edge is $50 pixels$ or more far away from the point, the corresponding point is considered to be at infinity, and this control point is then excluded from the vector of correspondences sent to least squares. Other filtering can be done such as using RANSAC[9] algorithm to ensure consistency over correspondences before deducing motion screw based on them.

3.2.5 Least Squares Formulation

This step is concerned about finding the motion increment that the 3D object made in the 3D world. It is represented by 6DoFs which are 3D translation and 3D rotation. The derivation for this least squares formula will differ from the derivation mentioned in [1] [2] due to the difference in assumptions, as such,

they assume that the relation between the object and the camera is just translation denoted as T and they assume that there is no rotation involved. However, in our case, we do have rotation and translation between the camera frame and the world frame and therefore between the object frame and the camera frame. We believed that the formula derived in this section is more generic to a wider spectrum of situations.
Given a 3D point P in the world coordinate frame, and a camera that has the relation to the world frame governed by the rotation matrix R and the translation vector T. Point P can be represented in the camera coordinate system as:

$$M = RP + T \tag{4}$$

Assume that the object performed a small motion that is represented by a change of position Δt and a change in rotation δr. The point projection in the camera frame after this small motion is given by:

$$M^{'} = R(\Delta r.P + \delta t) + T \tag{5}$$

Harris[1] assumed that the rotation Δr can be linearized to be:

$$\Delta r = I + \delta\Omega \tag{6}$$

Where $\delta\Omega$ is a skew symmetric matrix of the 3D rotation vector, and is given by:

$$\delta\Omega = \begin{bmatrix} 0 & -\omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \tag{7}$$

Providing the formula $M^{'}$ in terms of M is given by:

$$M^{'} = R.P + R.\delta\Omega.P + R.\delta t + T \tag{8}$$

Given the formula: $M = RP + T$, we can re-write the above equation as:

$$M^{'} = M + R.\delta\Omega P + R\delta t \tag{9}$$

To represent this equation in terms of the image projection, both M and $M^{'}$ should be projected to the image plane using the intrinsic camera matrix:

$$m = KM \qquad m^{'} = kM^{'} \tag{10}$$

However, the inputs to our system is image correspondences m and $m^{'}$, therefore, these input points needs to be normalized by the intrinsic camera matrix.

$$\begin{bmatrix} u \\ v \end{bmatrix} = K^{-1}m = M \qquad \begin{bmatrix} u^{'} \\ v^{'} \end{bmatrix} = K^{-1}m^{'} = M^{'} \tag{11}$$

Expanding the above equations and omitting all nonlinear terms, will result:

$$u^{'} = u + \frac{1}{T_z + P_z}(\delta t_x + \delta\Omega_y P_z - \delta\Omega_z P_y - u(\delta t_z + \delta\Omega_x P_y - \delta\Omega_y P_x)) \tag{12}$$

$$v^{'} = v + \frac{1}{T_z + P_z}(\delta t_y + \delta\Omega_z P_x - \delta\Omega_x P_z - v(\delta t_z + \delta\Omega_x P_y - \delta\Omega_y P_x)) \tag{13}$$

Which can be re-written in the following matrix format:

$$\begin{bmatrix} u^{'} - u \\ v^{'} - v \end{bmatrix} = \begin{bmatrix} -uP_y & P_z + uP_x & -P_y & 1 & 0 & -u \\ -P_z - vP_y & vP_x & P_x & 0 & 1 & -v \end{bmatrix} \begin{bmatrix} \delta\Omega_x \\ \delta\Omega_y \\ \delta\Omega_z \\ \delta t_x \\ \delta t_y \\ \delta t_z \end{bmatrix} \tag{14}$$

Given that the distance between the projected 2d point and the corresponding 2d point is mainly a vector and is given by:

$$l = k^{-1}(m^{'} - m) \tag{15}$$

The above matrix can be re-written to be:

$$l = W\delta p \tag{16}$$

Which is a least squares formulation that can be solved alliteratively, or by the closed form given as:

$$\delta p = (W^T W)^{-1} W^T l \tag{17}$$

The resulting motion screw δp must be represented in the world frame, and this can be done by multiplication to the inverse of the rotation matrix between the camera and the world frame:

$$\delta p_{world} = R^{-1}\delta p \tag{18}$$

By finding this quantity, we are able to update the object pose in the world frame successfully.

3.2.6 Object Pose Update

After the transformation of the motion screw to the world coordinate system, we can update 3D control points by the following equation:

$$P_{new} = P_{old} + [[\delta\Omega]_\times | \delta t] P_{old} \tag{19}$$

Where $[\delta\Omega]_\times$ is the skew symmetric matrix of the vector of rotation increment.

4 Results and Discussion

4.1 Experimental Setup

For the sake of having full control over the environment, a synthetic scene was rendered using blender 2.8 that includes a cereal box-like object that does simple movements in the 3 axes, followed by rotation around the 3 axes, and finally performing composite motions that include both translation and rotation around one or multiple axes at a time. The sequence yielded 800 images with 9 high contrast edges visible during the whole sequence. The sequence was used for testing and evaluation of different system blocks. The objective is to also test the system against real video sequence in order to confirm the generalization of the solution and to pinpoint possible enhancements to the pipeline to allow it to be a suitable solution for real-world tasks.

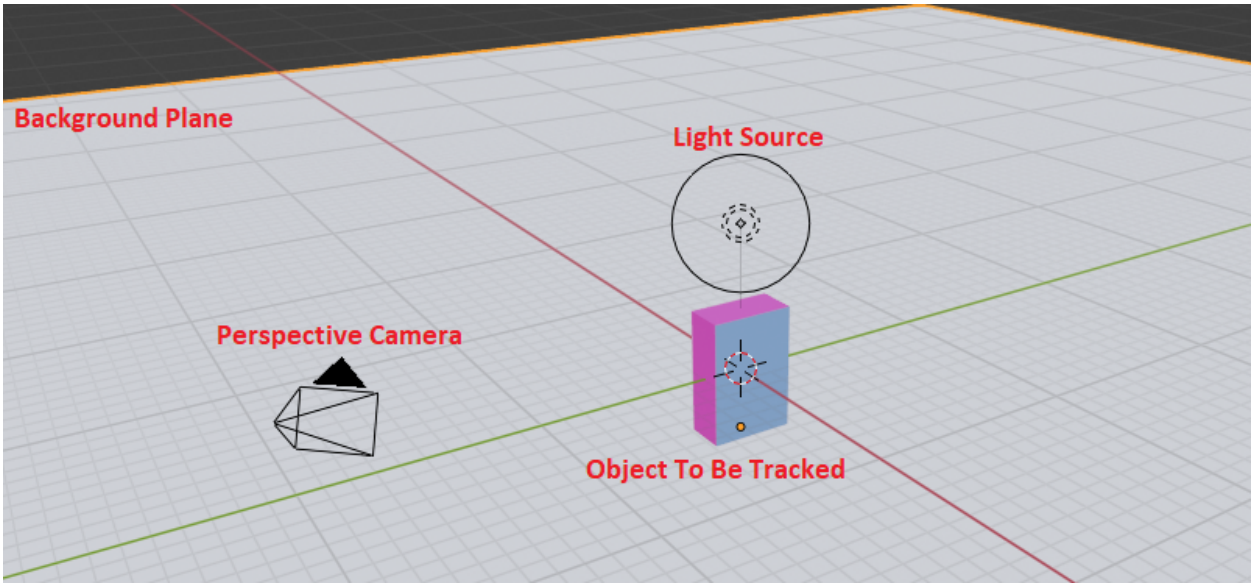
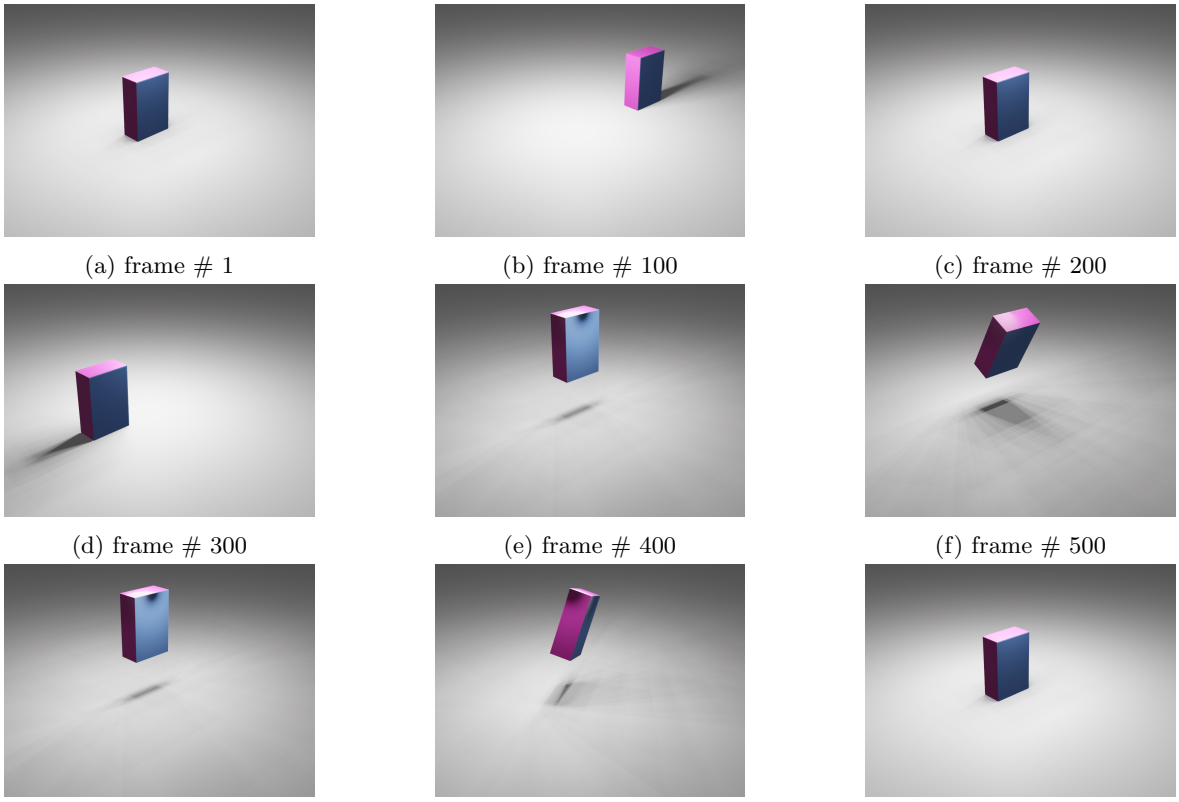


Figure 3: Synthetic Environment Diagram Showing Its Components

4.2 3D Model of Tracked Object

The 3D model is a cereal box-like object with dimensions of 1 x 2 x 3 m located at the center of the world frame. The object vertexes are extracted from blender and was supplied to our code. The object is represented as a matrix of size 12x6, were each row represent an edge and each edge is represented by the two end points of this edge. The points locations are supplied in the world coordinate frame.



(a) frame # 1 (b) frame # 100 (c) frame # 200

(d) frame # 300 (e) frame # 400 (f) frame # 500

(g) frame # 600 (h) frame # 700 (i) frame # 800

Figure 4: Samples Frames from the Synthetic Data Set

4.3 Camera Calibration

Due to having a synthetic data set, the camera calibration parameters are extracted from blender. The code was used to generate the full camera matrix (having intrinsic and extrinsic parameters), extrinsic alone, and intrinsic alone. The results were then plugged into our integrated system in a configuration file. The following is the extracted camera information from the blender model.

Image Size	1024 x 768	Pixels
Focal Length	(1422.2, 1422.2)	mm
Principle Point	(512, 384)	Pixels
Distortion Factor	(0, 0)	%
Camera Translation (World Frame)	(28.3589, -24.9258, 16.9583)	m
Camera Rotation (World Frame)	(63.5593, 0, 46.6919)	Degrees

Table 1: Synthetic Camera Calibration Parameters

4.4 System Results

4.5 Discussion

5 Future Work

Section

Conclusion

References

[1] C. Harris and C. Stennett, "Rapid-a video rate object tracker.," in *BMVC*, pp. 1–6, 1990.

[2] V. Lepetit, P. Fua, *et al.*, "Monocular model-based 3d tracking of rigid objects: A survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005.

[3] D. Cobzas and M. Jagersand, "3d ssd tracking from uncalibrated video," in *Spatial Coherence for Visual Motion Analysis* (W. J. MacLean, ed.), (Berlin, Heidelberg), pp. 25–37, Springer Berlin Heidelberg, 2006.

[4] A. Tyagi, M. Keck, J. W. Davis, and G. Potamianos, "Kernel-based 3d tracking," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[5] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers, "Combined region and motion-based 3d tracking of rigid and articulated objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 402–415, March 2010.

[6] M. Haag and H.-H. Nagel, "Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences," *International Journal of Computer Vision*, vol. 35, no. 3, pp. 295–319, 1999.

[7] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[8] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," *a talk at the Stanford Artificial Project in*, pp. 271–272, 1968.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.