Real Time Pose Estimation for Model-based Rigid Object

Guan-Horng Liu
Carnegie Mellon University
Robotics Institute
guanhorl@andrew.cmu.edu

Hsueh-Lin Huang
Carnegie Mellon University
Mechanical Engineering
hsuehlih@andrew.cmu.edu

Abstract

Object detection and tracking stands as a fundamental but crucial problem in robotics field. Applications in various scenarios such as manipulation, autonomous driving, and humanrobot interaction have been widely investigated. In this project, we aim to focus on tracking and 6-DoF pose estimating of a model-based rigid object using low-cost video stream device. More specifically, we would like to introduce a real-time framework of model-based object recognition and 6-DoF pose tracking in the applications of manipulation and learning-by-showing scenarios.

1. Introduction

Tracking an object in a video sequence means continuously identifying its location when either the object or the camera are moving. There are a variety of approaches, depending on the type of object, the degrees of freedom of the object and the camera, and the target application.

Two dimensional tracking typically aims at following the image projection of objects or parts of objects whose 3D displacement results in a motion that can be modeled as a 2D transformation. Two dimensional tracking only tracks features in two dimensional space, without any concern to camera movement or distortion. An adaptive model is then required to handle appearance changes due to perspective effects or to deformation.

Three dimensional tracking aims at continuously recovering all six degrees of freedom that define the camera position and orientation relative to the scene, or, equivalently, the 3D displacement of an object relative to the camera. Estimating and tracking the 6-DOF (three translation and three rotation) pose of rigid objects is critical for robotic applications, involving object grasping and manipulation. Common Techniques for inferring depth information for 3D reconstruction, including 3D modelbased approach and RGB-D image based method, such as [5] and [3]. [4] provides us the different techniques for tracking of complex 3D objects, i.e. natural features such as edges or surface texture.

In this project, we can roughly separate the whole idea to three parts, tracking, recognition, and learning. For tracking, we use edge detection instead of pure SIFT. The reason is that using SIFT tracking will typically require 3D texture model. The method we introduced would like to generalize to cases where textures are not necessary to provide. For example, the objects are not easy to extract textures or even a textureless object, or the texture of object is occluded by some other objects. Moreover, SIFT is relatively computation costly. Running each single iteration with SIFT takes too much effort and time for tracking. Therefore, using edge detection here is a more reasonable option. For the other two parts, the learning node learns the SIFT of the object and pass it to the recognition node. Notice that learning node is only activated when human decide the keyframes for the robot to learn. When the tracking node lose track of the object, it calls recognition node for reinitializing the pose.

We have also compared the result from BLORT with OpenCV, which iteratively tracks the object with its texture. Using Oriented FAST and Rotated BRIEF (ORB) to extract the 3D and 2D feature descriptors, OpenCV then generates 3D/2D correspondence points with flann-based matcher. RANSAC based perspective-n-points (PnP) algorithm is then implemented for pose estimation. In the end, OpenCV uses Kalman filter to filter out the bad poses. Unlike BLORT's interacting system between recognition and tracking nodes, OpenCV extracts ORB features of each frame and runs PnP algorithm each iteration.

2. Problem Description

We considered a scenario where human placed a new object (e.g. a Pringle bottle or a black tea box bought from supermarket) on kitchen table. Then he asked a manipulation robot to first detect object, then registered its texture information as human demonstrated different views of the object to robot. The robot should register texture on object and estimated its 6-DoF pose $x = [R|t]$ at the same time in order to register texture information corresponding with its relative 3D position. Fig.1 shows how human actually

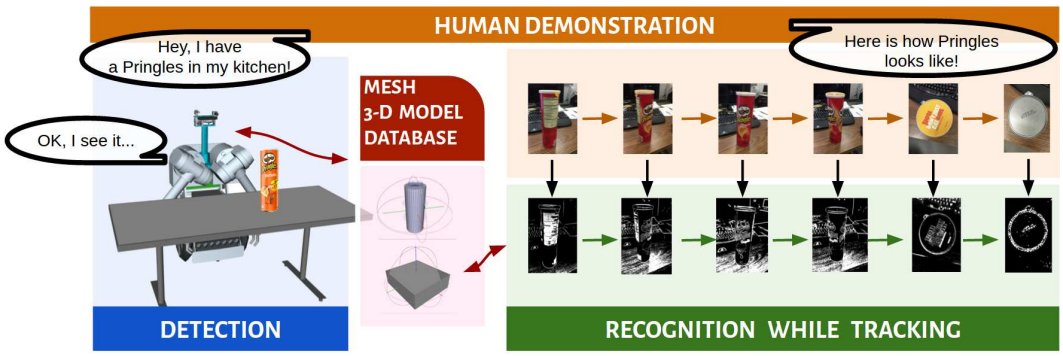

Figure 1. Interaction between human and robot under our specific scenario.

interact with the robot.

Here we make two fair and reasonable assumptions for this specific scenario. First, we restrict the object with basic geometric shape (e.g. cuboid and cylinder). This assumption is acceptable since most of the rigid objects in kitchen, such as cans, bottles, and food box, can be simplified as basic-shape geometric objects. Second, we assume the robot already acquired a textureless 3D mesh model of the object he is asked to recognize, which can be either acquired on-line by robot itself, or off-line by manual specified. Since the objects are in basic shape, specifying the geometric dimension of object to robot is not a painful task for human.

The whole scenario can be roughly separated as two subtasks: (a) detection of a new object based on its textureless 3D model, and (b) recognition the object, incremental registration of its texture information, and estimation of its 6-DoF pose all at the same time during human demonstration. In this project, we focus on the latter one problem. That is to say, we assume we have the initial pose of the object with only pure geometric shape mesh model.

3. Platform and Device

The whole system is built on Robotics Operation System (ROS), an open-source distribution control system which is widely-use in robotics community. More specifically, Blocks World Robotics Vision Toolbox (BLORT [5]) is utilized for our framework. For hardware device, we use a low-cost RGB-based webcam (1.3Mpx iBuffalo BSW13K10H) along with gstreamer [2] on ROS in order to extract images from video stream. Camera calibration is done manually by using MATLAB Camera Calibration Toolbox.

4. Framework Demonstration

A brief framework is shown in Fig.2. With respect to functionality, we can basically divide the software architecture into two nodes: Detection/Recognition and Tracking. Given the initial pose from recognition node along with geometric model, the object was tracked based on monte carlo particle filter to generate a 6-DoF pose that maximize edge correlation between current image and the rendered image from its 3D model. While tracking, user could manually trigger learning node, which extracted texture informations from current 2D image and then mapped them onto 3D surface model given the estimated pose from the tracker. Under this framework, scale invariant feature transformation (SIFT) with GPU acceleration is chosen to encode the interest points of texture image. These informative descriptors are aimed to gradually refine the textureless geometric model such that a more accurate initial pose can be found when tracking was reported failed and need to be re-initialized. Note that SIFTs matching only occurred within Recognition node. The whole pose estimation process is a pure edge-based 6-DoF tracking. This is a reasonable design since finding key points and matching their descriptors could be computationally expensive to implement in real time.

The detailed process inside tracking node as well as its interaction with recognition node are shown in Fig.3, and the link to demo video is also provided in Section 8.1. In the first stage (a.k.a. Image Processing stage in Fig.3), the edge image of the incoming image I_C is detected. In the meantime, a colour texture image I_S is rendered by projecting every edge inside a convex region, which is defined by the previous weighted mean pose of tracking object and the geometric model. Then in the second stage (a.k.a. Particle Filtering stage in Fig.3), a number of particles are generated based on Gaussian noise, where each of the particle represents a 6-DoF hypothesis. The confidence level c^i of each particle x^i is computed based on the edge correlation response with I_C using the below formula:

$$c^i = \sum_{u,v} \phi^i(u,v)/A \tag{1}$$

where ϕ^i is the correlation edge image, and A is the normalization term. Since the confidence levels can be thought as

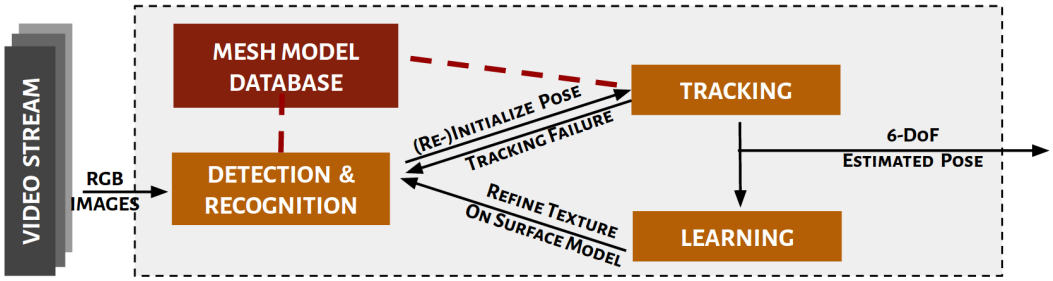

Figure 2. Brief description of Tracking and Recognition/Detection framework.

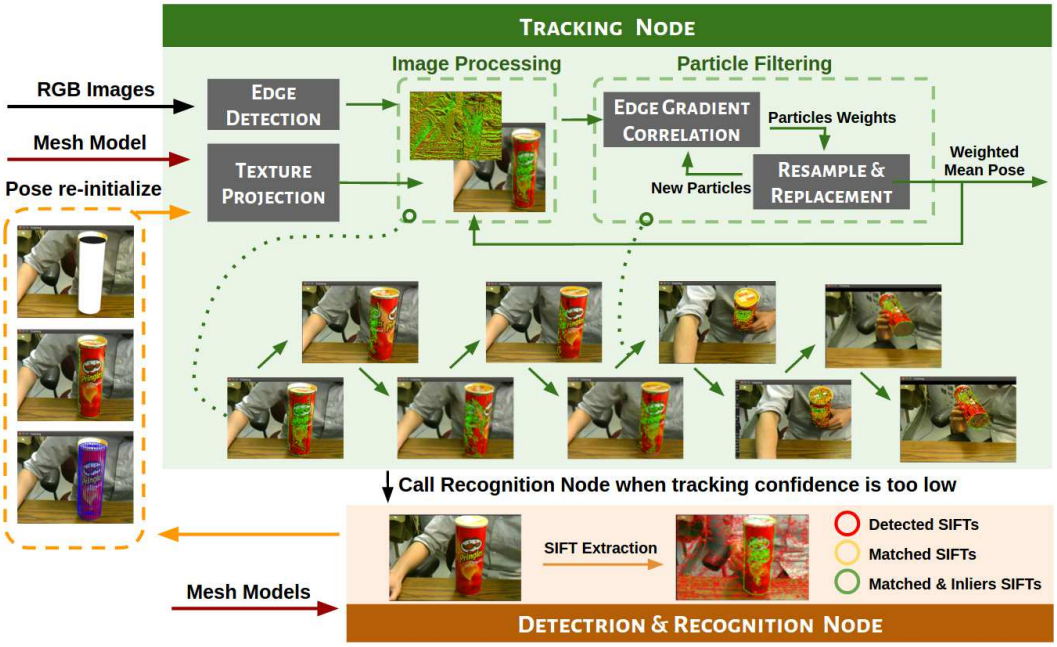

Figure 3. Detailed processes inside Tracking and Detection/Recognition.

probability of pose hypothesis matching with actual pose, new particles is resampled based on previous particles with weighted distribution. Then the whole correlation evaluation process repeats again for several iterations depending on the computational power. Finally, the weighted mean pose is outputted as the estimated 6-DoF pose, and stored as input for image processing stage in the next iteration. In practical, we found out that 150 particles can generally give a satisfying estimation of tracking pose.

If the weighted confidence level of trackers output is below a threshold, tracker will report failure and call recognition node for re-initialization. The recognition node uses the previously registered SIFT codebook entries to find a good pose. If none of the SIFT features are registered, e.g. at the very first iteration where we start with totally textureless model, edge-based detection is used to search initial pose, which we mentioned in the last paragraph of Section 2 that it is not in the scope of this final project.

5. Experiments

5.1. Single Object Tracking

The full tracking process after completion of texture registration can be found in the link of domo video in Section 8.2. Screenshots of how tracking recovery from bad pose tracking and its interplay with recognition node are shown in Fig.4.When the tracking node was not confident enough with his pose, an exception would be thrown toward recognition node and waited until he found a new initial pose for tracking.

5.2. Single Object Occlusion

Fig.5 shows the screenshots of occlusion effect on single object with the link of demo video in Section 8.3. Since

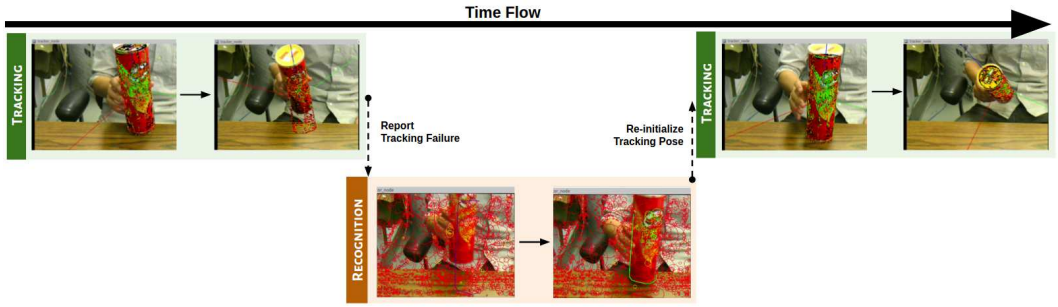

Figure 4. Screenshots of demo video in Section 8.2 demonstrating tracking failure report and recovery with re-initialization interaction.

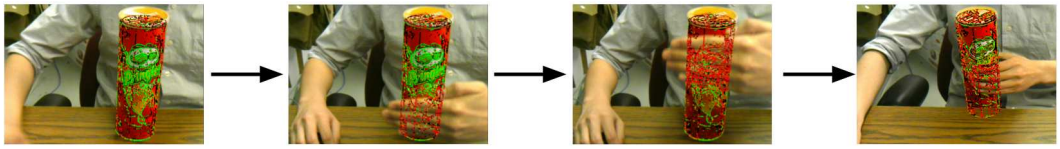

Figure 5. Screenshots of demo video in Section 8.3 demonstrating robust tracking of single object with occlusion.

tracking is based on edge correlation, it is very robust to the occlusion of the static object. In practical, we found out that it can successfully track the pose even if 60% of texture is occluded.

5.3. Multi Objects Tracking

Fig.6 shows the screenshots of multi objects tracking with the link of demo video in Section 8.4.No more modifications are needed to perform tracking from single to multi objects tracking unless 3D mesh models should be provided for each rigid objects. However, for multi objects tracking, it is more vulnerable to cluttered background, but would work fine with pure cleaner one as shown in Fig.6. Another thing we would like to mention is that the 3D model did not need to accurately match the object geometric shape. For example, the coke can in Fig.6 is simplified as cylinder with sightly dimension error but could still be tracked well.

5.4. Comparison with OpenCV Tutorial [1]

Feature matching is at the base of many computer vision problems, such as object recognition or structure from motion. In OpenCV example, we extract the feature with ORB, which is at two orders of magnitude faster than SIFT. With flann-based matcher, we can generate the 2D/3D correspondences and match them with the descriptors of each frame from the video stream. We then calculate the rotation matrix and translation matrix with the correspondence points with PnP algorithm. By updating the Kalman filter in each iteration by filtering out the bad pose, the tracking system works though sometimes loses track of the object if the inlier points didn't exceed the given threshold. We show the comparison of OpenCV and BLORT in Section 8.5.

Fig.7 shows the screenshots of experiment results from BLORT and OpenCV. The first pair of images shows that both the initial pose estimation of BLORT and OpenCV. However, if the object moves too fast, both trackers lose their track of the object. We can tell from the third and the fourth pair of images, BLORT can immediately reinitialize their pose from SIFT-based detector, while the OpenCV loses its track afterwards. This shows BLORTs fast recovery of new initial pose compared with OpenCV tutorial when losing track of the object.

6. Discussions

6.1. Tradeoff using pure edge-based tracking

Though edgebased tracking has been shown to be robust with occlusion and cluttered background if more textures are registered, we discovered the effect of small pose offset error in cases where too many textures are provided. The reason is that since the tracking algorithm makes no distinction between tracking geometric shape edges and tracking textures – i.e. they are all treated as edges with mean weights – the scaling and mapping errors occurred during texture registration stage would cause small offset of the tracking pose. See the right-upper two images of Fig.4 for example, on the left image where texture edges are much more than shape edges, the estimated pose tended to have small pitch offset in order to match all the textures on the Pringles side surface. However, on the right image where texture edges did not dominate the edge correlation process, better pose can be found and tracked.

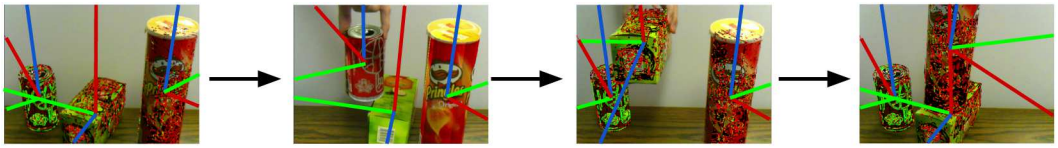

Figure 6. Screenshots of demo video in Section 8.4 demonstrating multi objects tracking in clearer background.

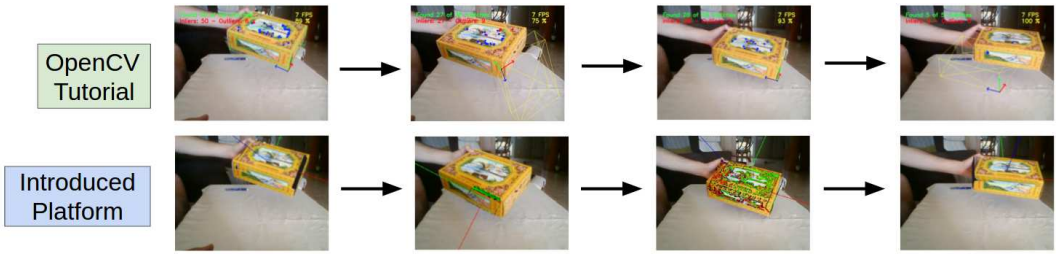

Figure 7. Screenshots of demo video in Section 8.5 and demonstrating that BLORT's performance is much better than OpenCV.

6.2. Effect of fast translational change and rotational change of pose

In practice, we found out that the particle filtering of tracker is relatively fragile in fast translational change of object pose compared with rotational change. One of the reasons is that the gaussian noise added for generating particles has tighter translational bounds with respect to rotational one. Another reason is that for this project we utilized a lowcost webcam with only 8 frames per second (FPS), while the average tracking process can be finished within 0.07sec. It can possibly be improved in future work if better webcam device with higher FPS is used.

7. Conclusion

We implemented a real-time framework of rigidbody pose tracking and recognition in the application of human-robotinteraction. More specifically, we focused on a scenario where robot is asked to detect new object in kitchen table and then registered its texture information from human demonstration. Combining fast edge-based tracking along with relatively slow but accurate recognition, experiments have been conducted using our own objects with a low-cost webcam. The performance of edge-based tracker increases when providing textures SIFT but still works if not provided. Single object tracking is robust with occlusion and cluttered background, while multi objects tracking can also be achieved under the same framework but with clearer background. The comparison of this framework with the similar work on OpenCV tutorial, which we treated as our baseline, clearly shows the effectiveness of the interaction mechanism – i.e. pose failure report and pose recovery between two separable processes.

8. Demo Videos

8.1. Training(Registering) Pringles

`https://youtu.be/_sVlaFlY3cI`

8.2. Single Object Tracking

`https://youtu.be/eU8BmNqziWo`

8.3. Single Object Occlusion

`http://ppt.cc/yJCkV`

8.4. Multi Objects Tracking

`http://ppt.cc/Ybia4`

8.5. OpenCV and BLORT comparison

`https://youtu.be/PcoYONoizQA`

9. Division of labor

Guan-Horng Liu (guanhorl) – Paper reviewing, ROS platform setup and experiment conduction w/ BLORT, Discussions.

Hsueh-Lin Huang (hsuehlih) – Paper reviewing, Webcam camera calibration, OpenCV tutorial comparison, Discussions.

References

[1] Opencv tutorial: Real time pose estimation of a textured object. See also URL `http://docs.opencv.org/3.1.0/dc/d2c/tutorial_rea`

[2] ROS, gscam package. See also URL `http://wiki.ros.org/gscam`.

[3] U. Asif, M. Bennamoun, and F. Sohel. Real-time pose estimation of rigid objects using rgb-d imagery., 2002. In: ICIEA. (2013).

[4] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey, in foundations and trends in computer graphics and vision, 2005. vol. 1, num. 1, p. 189.

[5] T. Mrwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. Blort the blocks world robotic vision toolbox best practice in 3d perception and modeling for mobile manipulation (in conjunction with icra 2010), 2010.