# **3D Object Tracking and Reconstruction**

|  |  |
| --- | --- |
| Ishan Patil  Dept. of Electrical Engineering  Stanford University  iapatil@stanford.edu | Second Author  Institution2  First line of institution2 address  http://www.author.org/\_second |

**Abstract**

*The goal of this project is to try to track single objects in surveillance videos and then attempt to reconstruct the tracked objects as a 3D model in real time. Objects to be tracked will be selected by the user using manual bounding box selection, and then will be tracked using the techniques described in the paper by Feng et al.: On-line Object Reconstruction and Tracking for 3D Interaction [1]. We plan on implementing the components from [1] ourselves, only using existing libraries, such as OpenCV, for basic preprocessing steps or in situations where our implementations prove to be too slow for our needs.*

# 1. Introduction

The goal of this project is to try to **track** objects in surveillance video and then attempt to **reconstruct** the tracked objects as a 3D model in real time. Objects to be tracked will be selected by the user using manual bounding box selection, and then will be tracked using the techniques described in the paper by Feng et al.: *On-line Object Reconstruction and Tracking for 3D Interaction* [1]. We plan on implementing the components from [1] by ourselves, only using existing libraries, such as OpenCV, for basic preprocessing steps or in situations where our implementations prove to be too slow for our needs.

# 2. Problem Statement

The overall project is composed of several steps: detection, segmentation, tracking, and 3D reconstruction. In detection step, the user will select an object by creating a bounding box around the object of interest. Using this bounding box, we will find a set of Shi-Tomashi feature points. Using these feature points, segmentation is performed with Fast Local Kernel Density Estimation (FLKDE). Lastly, we will directly use the Lucas-Kanade Tracker in OpenCV to perform tracking in next keyframe. On the other hand, 3D reconstruction will be performed using features points collected from continuous keyframes.

We plan to use VISOR Dataset [2], which contains a large set of labeled videos captured from surveillance cameras. This dataset contains many different types of labeled data for various types of object recognition tasks, such as people recognition, face recognition, and indoor and outdoor objects tracking.

For evaluation, we plan to use several types of metrics following [3]: precision plot, success plot and robustness evaluation. Precision plot is the average Euclidean distance between the center of tracked object and the ground truth; successful plot is the ratio of the overlap of tracked.

# 3. Technical Approach

## 3.1. Feature Detection

## 3.2. Segmentation

* 1. Tracking

we outsourced the tracking component of this project by leveraging the OpenCV functions, using KLT technique.

* 1. 3D reconstruction

we will use the knowledge learnt in class on triangulation to perform this task. We will also combine with RANSAC to improve the result. We leave this implementation after this mid-term report.

# 4. Preliminary Results

4.1 Feature Detection

In order to perform segmentation and tracking, we need to find features in the image that are relatively interesting that can be tracked. To do this, we use the Shi-Tomashi corner detection algorithm to find feature points in the bounding box selected by the user. Since this is a relatively simple algorithm, we have implemented this ourselves and compared its performance against the OpenCV implementation, cv2.goodFeaturesToTrack.

As seen in figure 1, our implementation of the Shi-Tomashi algorithm performs relatively well when compared to the OpenCV implementation, with many of the same feature points detected by both implementations. The differences in the sets of feature points is likely due to mismatching parameters to the algorithm, as it’s not completely clear what some of the defaults in OpenCV are set to.

Although our implementation of Shi-Tomashi produces reasonable results compared to the OpenCV version of the algorithm, our version is much slower. OpenCV can find the feature points above in a fraction of a second, whereas our implementation takes over a minute to run. This is due to the fact that our implementation is done completely in python, while the OpenCV implementation is written in C++ and is parallelized.

Figure 1: Comparison of outputs from customized Shi-Tomashi implementation (top) in green and the OpenCV implementation (bottom) in purple.

4.2 Segmentation

4.3 Tracking

OpenCV does a good job in reading video data and tracking across frames and it will also convert keyframes into grayscale images. The function cv2.calcOpticalFlowPyrLK helps to calculate the new corresponding points in the new keyframe, using pyramidal KLT. The overall performance looks good.

By the time of the midterm report, we haven’t yet connected the dots between the above steps. So, we can’t report any evaluated metrics for now.

# References

1. [1] Feng, Y., Wu, Y., & Fan, L. (2012). On-line Object Reconstruction and Tracking for 3D Interaction.2012 IEEE International Conference on Multimedia and Expo. doi:10.1109/icme.2012.144
2. [2]: Vezzani, Roberto, and Rita Cucchiara. "Video surveillance online repository (visor): an integrated framework." Multimedia Tools and Applications 50.2 (2010): 359-380.
3. [3]: Yi, W., Jongwoo, L., Ming-Hsuan, Y. (2013). Online Object Tracking: A Benchmark. CVPR2013, Computer Vision Foundation.