Hand-to-Target Guidance Using Multi-Object Tracking

Elise Newman
Department of Computer Science
University of California, San Diego
La Jolla, CA 92093
eanewman@ucsd.edu

Abstract

The GroZi project at UCSD is working toward a complete system that can guide a visually impaired person around a store to collect the items on their shopping list by solving a wide array of computer vision problems. The system described by this paper should simultaneously track both the product and the hand in order to produce either sonic or haptic guidance toward the targeted product, assuming that the person using the system is (a) in the correct aisle, approximately one meter away from the desired product, and (b) looking for an item that is on a list of previously selected items. The tracking begins with a frame zero that contains (a) a bounding box around the targeted product, and (b) the outstretched hand. This system will be integrated with other GroZi components, including text recognition "in the wild" which will aid aisle sign processing, and product detection.

1. Overview

The objective of this project is to simultaneously track a shopper's hand and the targeted item in order to periodically update sonic or haptic guidance. This paper describes the first steps taken toward this goal, including what worked, what didn't work, and some of the main problems encountered and solutions to them. It is intended to aid whoever works on this component of the system next. The website for this project is available at *http://httguidance.blogspot.com/*

The goals of the project include addressing the following questions:

- How can two trackers be best combined into one smooth tracker?
- How can I effectively guide the hand to the item with information obtained from the joint trackers?
- How often should I update the feedback?

The current task is to modify Kalal's OpenTLD tracker, which mitigates common causes of tracker failure by employing pipeline stages of tracking, learning, and detecting [4], [5]. This tracker can be used to simultaneously track both the hand and the object it reaches for. The next task will be to determine effective modes of feedback, starting with the most basic (perhaps a quad-directional text-to-speech output), and culminating in finer-grained feedback.

2. Camshift

One approach to hand detection is to use Camshift [2], an adaptation of the mean shift algorithm that employs continuously adaptive probability distributions, as a starting point for the system. Hewitt provides a Camshift wrapper [4] that serves as an interface to OpenCV's implementation of the algorithm. The results obtained from running this software on a simple piece of footage of a hand reaching toward a solid-colored square against a blank white background are far from ideal. The parameters constantly need tweaking depending on lighting conditions and the tracking is inconsistent. This can be attributed to the histogram-based nature of the tracker, which harvests the various hues of the tracked object in the initial frame of the footage. Since this histogram is only calculated once, changes in lighting render tracking performance less reliable, as the hue of the tracked object starts to vary. Post-occlusion recovery is also unreliable. Kalal offers

OpenTLD, a much more robust approach nicknamed "Predator" because of its relentless tracking ability under many different lighting conditions, transformations and scaling.

3. OpenTLD

Kalal uses a Lucas-Kanade-based tracker in order to address occlusions, a major problem area for Camshift. "TLD" stands for Tracking, Learning, Detection, the three main stages of the algorithm executed during each iteration. The learning part of the tracker involves gathering positive examples of the tracked object, and negative examples which are nearby but sufficiently distinct from the positive examples. This aids detection in later iterations of the tracker, constantly increasing the ability to recover from being "sidetracked" or from the object leaving the field of view [5]. The TLD detector is novel in that it is constructed and updated online [4]; it doesn't rely on a large amount amount of previously accrued data like AdaBoost, the state-of-the-art offline detector, does. This paradigm for trackers, which packages online learning with tracking to increase reliability, allows for a more general-purpose tracker. This type of tracker is therefore appropriate in the context of the GroZi project for tracking both the hand that reaches for the product and the product itself. The following sequence shows the tracker's ability to recover from occlusion:

Figure 1. Bounding box around product just before occlusion (negative examples in column along left side, positive examples in column on right side)

Figure 2. Product still detected accurately with partial occlusion

Figure 3. No detection during complete occlusion

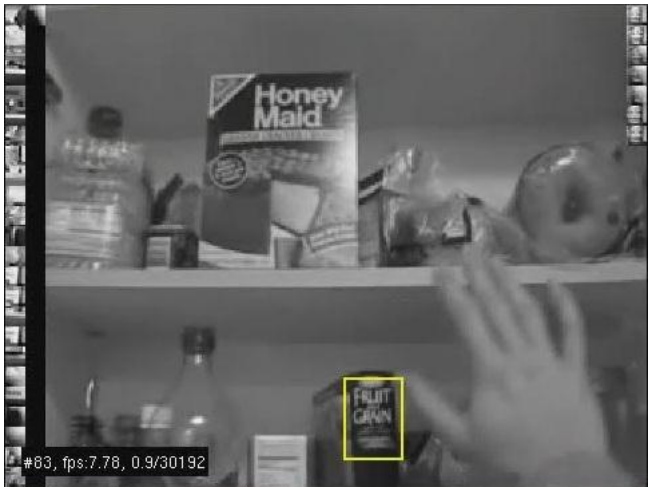

Figure 4. Product re-detected after occlusion using positive examples

4. Combining Trackers

This section addresses how to combine two trackers to process a single input sequence simultaneously.

OpenTLD is currently implemented in Matlab and calls OpenCV routines via the MEX interface. The Matlab compiler, mcc, is required to call Matlab from C++ or vice versa. Since mcc is not compatible Student Version of Matlab, OpenTLD was modified to support multiple trackers. OpenTLD does not allow for easy instantiation of multiple trackers operating on a single piece of footage. This could be because the learning already takes a substantial amount of time, even with just one tracker. On a Core i3 CPU running at 2.13 GHz with 4G RAM, running Ubuntu 10.04, OpenTLD ranges from processing between 1 and 12 frames per second with a single tracker, but with two trackers, the performance deteriorates and hovers around 1 to 2 frames per second. Once the implementation's migration from Matlab to C is complete this will likely be improved.

The process of merging the two trackers is nontrivial because the OpenTLD framework is not conducive to employing multiple trackers on the same piece of footage. Since writing a C++ application that utilizes OpenTLD code is not an option due to the lack of compatible compiler (mcc), the next best option is to find the minimal set of entry points into the tracking system necessary to instantiate and update multiple trackers operating on the same frame, while leaving the algorithmic details of the tracker abstracted. This produces a tracker in which theoretically, absent of restrictions due to limited time and resources, n items can be tracked in a single frame.

The top-level entry point is the m-file Run_TLD_demo.m. This function calls tld_Demo.m, which then calls the other components that perform tracking, learning, and detecting. The main variable that is updated within most of the functions is a struct called "tld" which contains all of the information about a single tracker. In order to extend the project to support multiple trackers, a struct of trackers was used instead. The trackers were each initialized with their own respective bounding boxes and then iterated through during each step of each TLD iteration. After each iteration of tracking, detection, selection between tracking and detection, and learning, a single frame was updated with the new bounding boxes produced by each tracker. Problems I encountered with TLD can be found in section 6.

In order to test the multi-object tracker a library of videos was generated. A series of 5-second movies of blindfolded subjects reaching for an item on their shopping list was captured from over the shoulder opposite of their reaching hand. Vocal direction acted as a stand in for the auditory or haptic feedback that will eventually be built into the system. Shown below are two frames from some footage that has been processed by the multi-tracker version of OpenTLD.

Figure 5. Hand as it might approach a product

Figure 6. Hand, after successfully guided to the product

5. Possible Improvements

A couple of OpenTLD's strengths are its ability to continue to recognize an object as it is covered or begins to disappear from the frame, and to re-recognize it after it has been completely hidden and becomes *completely visible* again. However, if a tracked object is completely covered and then only partially reappears, the item it is not usually re-

recognized. This may be due to the aggregation of "positive" examples which almost solely contain the object causing the occlusion. As these examples begin to outweigh the previously collected positive examples of the sought object, the object in the foreground becomes the new target. The following sequence highlights this unintended artifact that the learning aspect of the tracker introduces.

Figure 7. Tracked object and simultaneously tracked hand, approaching from the left

Figure 8. Occluded product, still recognized

Figure 9. Occluded product mostly visible, but detection is now focused on the jacket cuff due to a new crop of positive examples containing the jacket cuff

A proposed solution to this would be to subsample several smaller regions within the larger region containing the currently tracked object and to use these smaller images to re-detect the object post-occlusion. Since OpenTLD "learns" from positive examples, there were enough positive examples generated by the cuff for the tracker to start tracking it and leave the poptarts undetected even when they reappeared. For this particular application, since the product will not likely be changing much, this effect could be mitigated by having distinct parameters for the product tracker and for the hand tracker. The product tracker's positive examples gathered toward the beginning of the tracking should be given higher weights, whereas since hands are very dexterous and constantly change appearance, the hand tracker should have parameters similar to those currently used in OpenTLD. This weighting system should reduce confusion over what the tracker should be aiming to re-detect.

6. Encountered Problems and Solutions

In undertaking this project several technical issues were encountered, mostly derived from the utilization of pre-existing code due to timing constraints. Often when compiling others' code, such as in the cases of the Camshift wrapper and OpenTLD, the message "undefined reference to…" would appear. This means that there was a linking error in the compiling process. At least one library required for the variable to be "recognized" was missing from the project and must be added. When attempting to run a MEX file included in OpenTLD (precompiled), the following message might occur:

```
??? invalid MEX-file
'/path/to/MEX/file:'
/path/to/Matlab/bin/glnx86/../../sys/os/glnx86/lib
stdc++.so.6:
Version 'GLIBCXX_3.4.11' not found (required by
/usr/local/lib/libcvaux.so.4).
```

Matlab has its own version of the C++ libraries necessary for the execution of the MEX file, which it automatically uses without ensuring that it is the correct version for the particular MEX file to execute. In order to correct this problem one solution is to make a symbolic link between Matlab's variable and the correct C++ library which resides in /usr/lib/ in Ubuntu 10.04:

```
cd /path/to/Matlab/sys/os/glnx86/
ln –s /usr/lib/libstdc++.so.6.0.13
~/path/to/Matlab/sys/os/glnx86/libstdc++.so.6
```

OpenTLD was created specifically for one tracker to operate on a frame upon each iteration. The main variable that gets passed to most of the functions, "tld," represents the singleton tracker that contains all information associated with relevant processing. Since most functions in the project expect this, an array of trackers was created to iterate through during each iteration of TLD. In order to pre-specify the desired dimensions, the elements of the array were set to null ([] in Matlab), but when the array was passed to other functions the dimensions were not preserved (the dimensions were 1x0). A struct of trackers was used instead. To iterate through them, the construct fieldnames(trackers) is used, which returns a cell array of field names that is iterable.

7. Future Work

This dual tracker can be used in order to determine the direction of the hand relative to the product sought by the GroZi user. A function of the lateral distance between the hand and the product, treated as if they existed on a 2-dimensional plane, can be used to determine the feedback delivered to the user. This feedback would consist of both directional and distance data. At first this feedback will likely be auditory, but eventually it will migrate to haptic feedback, or perhaps a combination of both, depending on which aspects of each prove to be most useful in guiding the user to his or her intended target.

References

[1] J. G. Allen, R. Y. D. Xu, and J. S. Jin, Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces. *Proceedings of the Pan-Sydney area workshop on Visual Information Processing*, ACM International Conference Proceeding Series Vol. 100 (Australian Computer Society, Inc., Darlinghurst, Australia, 2004).

[2] G. R. Bradski, Computer Vision Face Tracking for use in a Perceptual User Interface. Intel Technology Journal, 2nd Quarter, 1998.

[3] R. Hewitt, Seeing With OpenCV, Part 3: Follow That Face. *SERVO Magazine*, 2007.

[4] Z. Kalal, and K. Mikolajczyk, Online Learking of Robust Object Detectors During Unstable Tracking. On-line Learking for Computer Vision Workshop, 2009.

[5] Z. Kalal, K Mikolajczyk, and J. Matas, Forward-Backard Error: Automatic Detection of Tracking Failures. International Conference on Pattern Recognition, 2010, pp. 23-26.

[6] R. Stolkin, I. Florescu, and G. Kamberov, An Adaptive Background Model for Camshift Tracking with a Moving Camera, in Advances in Pattern Recognition, pp147-151, 2007.