Joint 3D object recognition and tracking

Orizio Riccardo,
Rizzini Mattia,
Zucchelli Maurizio

University of Brescia

25 july 2013

Introduction

The project goal is to realize a joint system of recognition and tracking of some features of a previously chosen object which is identified in a video. These features are called LABELS.
The sofware is aimed for execution on low performance devices such as the iPad, therefore the tracking is necessary due to the high computational cost of a pure recognition.

Introduction - Overall structure

The software is designed for concurrent execution.
There is an object of class MANAGER which, given every frame captured by a webcam or in a video, elaborates it in some way and retuns an object of class OBJECT which contains the actual LABEL positions.

Outline

Manager

The MANAGER handles two objects of class RECOGNIZER and TRACKER. The first object runs a *thread*.
Every N frames elaborated, the MANAGER asks the RECOGNIZER to perform a full recognition. In the meantime the TRACKER keeps tracking the LABELs starting from the last OBJECT given by the RECOGNIZER.
When the RECOGNIZER has finished it returns an OBJECT which contains the positions of the LABELs in the moment in which the recognition started. These positions are then *actualized* by the TRACKER.

Outline

Recognition

To perform recognition, *SIFT* descriptors are used since, in our tests, they seemed to perform better than the *SURF* ones without any computational disadvantage.

Recognition - Database

The recognition bases itself on a DATABASE.
This DATABASE supports serialization of the structures used for recognition. At the program start, a DATABASE name is provided: if this name corresponds to a serialized DATABASE, the DATABASE is loaded; otherwise it is trained.

Recognition - Database

The DATABASE training is performed through the analysis of a group of sample images depicting the object from various points of view.
To every image is also associated a text file which contains the absolute positions of the various LABELs in the referenced sample.
Every sample image is loaded, from every image *SIFT* features are extracted, descriptors computed and label positions loaded.
Everything is then saved in three structures and serialized.
At the end of the training or load phase, a *FLANN* based matcher is trained based on all the samples descriptors.

Recognition - Matching

To perform a match on a frame, the following steps are executed:

1. The SIFT features and descriptors are extracted from the frame.
2. The descriptors are matched using the *FLANN* matcher with a *knnMatch* method.
3. The matches are searched to identify the sample which has got the maximum number of matches. The only informations considered in the database are now those regarding this sample.
4. The matches of that sample are extracted and filtered using the *NNDR* criteria.

Recognition - Matching

5. If the remaining matches are less than 20 (estimated practically) then the match ends with nothing found (empty OBJECT is given back).
6. Otherwise the keypoints related to the matches in the database and the frame are used to estimate an homography with *RANSAC*.
7. If it is valid (the inliers ratio over the total points used by RANSAC is at least 50%), the homography is used to remap the positions of the labels in the frame. Those positions are added to an OBJECT which is passed back to the MANAGER.

Outline

Tracking

The tracking of the object is divided in two principal parts:

1. The effective tracking of the OBJECT between two successive frames.
2. The *actualization* of the OBJECT received by the RECOGNIZER.

Tracking

To track an object, *GFTT* (Good Features To Track) are used: these features are extremely fast to calculate and provide optimal results in tracking application.
The algorithm works on the whole frame: the features are associated to the new frame using Lucas-Kanade's method for *optical flow* estimation, then, for every LABEL, the nearest features to it are calculated using a *FLANN* based matcher and their movement between the frames is mediated and added to the LABEL's position.

Tracking - Actualization

The *actualization* process updates the OBJECT received from the RECOGNIZER to its position in the current frame. This is necessary because of the long time required by it.
To allow an easy and fast *actualization*, the TRACKER calculates and saves the features of the frame used for the recognition and starts using them internally for the tracking of the other frames. This way, when the OBJECT is received, the only operation to be done is the tracking from the saved features to the current features.

Tracking - Optimization

The heaviest part of the tracking algorithm is the calculation of the *optical flow*. To reduce the time required, the frame is decimated by a factor of 2; since the complexity of the algorithm is more than linear, this reduces the computational cost to less than 25% of the original time.
Also, to furtherly reduce the complexity, the whole tracker has been adapted to work with decimated frames. This can be done without damaging the results because both the features and the tracking algorithm are robust to scaling.

Outline

Conclusions

The software works very well in tracking simple shaped objects with lots of details (mostly because of the way SIFT works).
The combination of a tracking and recognizing system achieved pretty well the goal of creating a lightwight software. It is to notice that to achieve this, some sacrifice has been made, particularly in the tracking part which has the purpouse to supply at the slowliness of the recognizer. The biggest consequence of this is the inaccuracy in tracking objects in fast motion.
Since the requirements specified the tracking of slow motion, we are pretty good with that flaw.