# Joint 3D object recognition and tracking

Orizio Riccardo,
Rizzini Mattia,
Zucchelli Maurizio

Università degli Studi di Brescia

26 luglio 2013

# Introduction

The project goal is to realize a joint system of recognition and tracking of some features of an object which is identified in a video. These features are called LABEL.
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 *actualized* by the TRACKER.

# Outline



# Recognition

To perform recognition, *SURF* descriptors are used in place of *SIFT* ones because they are proved to be lighter in application.
The recognition bases itself on a DATABASE.
This DATABASE support 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 *SURF* features are extracted, descriptors computed and label positions loaded.
Everything is then saved in three structures and serialized.

# Recognition - Matching

At the end of the training or load phase, a *FLANN* based matcher is trained based on all the samples descriptors.
When the MANAGER asks the RECOGNIZER to perform a match on a frame, his *SURF* features are extracted and descriptors computed. The descriptors are then used to find a match in the *FLANN* index.
For every descriptor, the two best matches are given. Then the matches are filtered to exclude the ones in which the distance of the best match are too close to the distance of the second best match.

# Recognition - Matching

These good matches are analyzed to find out what is the sample who's got more of them. That sample is the only considered and his matching keypoints are extracted from the DATABASE.
Then an homography matrix is calculated between the sample keypoints and the ones in the frame. The matrix is used to map the positions of the LABELs in the frame. These LABELs are added to an OBJECT which is returned 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 algorithm is the calculation of the *optical flow*. To reduce the time required, the frame is decimated of a factor 2; since the complexity of the algorithm is higher than linear, this reduces the computational cost to less than 25% the original time.
Also, to furtherly simplify 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

It works, so shut up!