Semiautomatic Learning of 3D Objects from Video Streams

Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro

ISTI-CNR
Via G. Moruzzi 1
56124 Pisa - Italy
{fabio.carrara,fabrizio.falchi,claudio.gennaro}@isti.cnr.it

Abstract. Object detection and recognition are classical problems in computer vision, but are still challenging without a priori knowledge of objects and with a limited user interaction. In this work, a semiautomatic system for visual object learning from video stream is presented. The system detects movable foreground objects relying on FAST interest points. Once a view of an object has been segmented, the system relies on ORB features to create its descriptor, store it and compare it with descriptors of previously seen views. To this end, a visual similarity function based on geometry consistency of the local features is used. The system groups together similar views of the same object into clusters relying on the transitivity of similarity among them. Each cluster identifies a 3D object and the system learn to autonomously recognize a particular view assessing its cluster membership. When ambiguities arise, the user is asked to validate the membership assignments. Experiments have demonstrated the ability of the system to group together unlabeled views, reducing the labeling work of the user.

1 Introduction

In this work, a user assisted clustering system for online visual object recognition is presented. Our approach enables a single smart camera to learn and recognize objects exploiting change detection in the scene: given the evolution of the scene during time, the system incrementally builds a knowledge that can be exploited for the subsequent recognitions of the object when reappear on the scene. The user is queried when ambiguities cannot be automatically resolved.

Object detection is carried out by a local feature based background subtraction method [1] which distinguishes the foreground local features of the image from the background ones and segments new objects in the scene relying on FAST interest points. Each detected object, together with its extracted ORB local features, is maintained in a local database forming the knowledge base for object recognition. All the views of detected objects are incrementally organized in clusters based on the similarity among them. A similarity function between two object views is defined relying on local features matching and geometry constraints on their positions. The main goal of the system is to maintain gathered

views in clusters where each cluster contains only views of the same 3D object, even if it has been observed under different poses or illuminations (see Figure 1). Clusters can be labeled anytime by the user and object recognition is performed assessing the membership of a view to a particular cluster.

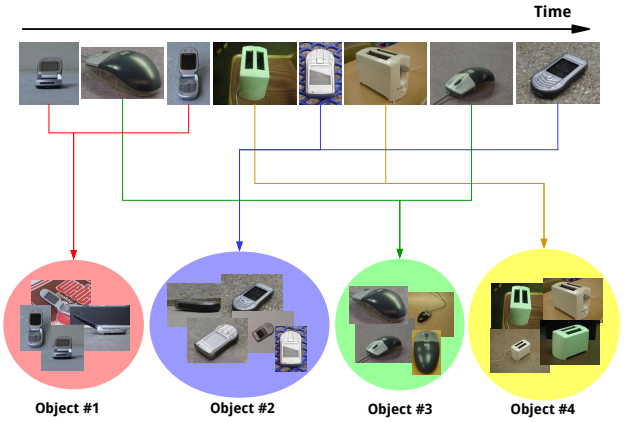


Fig. 1: Visualization of the system goal: online clustering of detected objects as recognition task.

The system has not been designed for a particular smart camera platform in mind, but it has been tested on the Raspberry Pi platform equipped with a Pi Camera module. Experiments have been made using the Stanford 3D Objects [1] [9] public available dataset in order to evaluate the ability to build a knowledge base for object recognition.

The rest of the paper is organized as follows: Section 2 describes the main features of some studied object recognition methods. Section 3 presents our method, describing the similarity function we have defined between detected objects. Section 4 describes the strategy used for similar object clustering. Section 5 reports the experiments performed and the metrics used to evaluate our method. Conclusive remarks are addressed at the end of this paper.

2 Related Work

Many solutions have been proposed to the problem of 3d object model learning for recognition task, starting from different 2d views of the object of interest.

Murase and Nayar [7] model each object as a manifold in the eigenspace obtained compressing the training image set for that object. Given an unknown input image, it is projected on the eigenspace and labeled relying on the manifold it lies on. Moreover, the exact point of the projection gives a pose estimation of the input object. However, only batch training is possible and the training set must be composed by a large number of normalized images with different poses and illuminantion and uncluttered background.

[1] `http://cvgl.stanford.edu/resources.html`

More recent studies address object modelling relying on local features of training images.

Weber et al. [10, 5] developed a method to learn object class models from unlabeled and unsegmented cluttered scenes. The authors combine appereance and shape in the object model using the constellation model. In this model, objects are represented as flexible constellations of rigid parts (features). Most robust parts are automatically identified applying a clustering algorithm to parts detected from the training set. An expectation-maximization algorithm is applied to tune the parameters of a joint probability density function (pdf) on the shape of the constellation and the output of part detectors. An enhanced version of this method using Bayesian parameter estimation is proposed by Fei-Fei et al. [4] for the purpose of scale-invariant object categorization capable of both batch and incremental training. However, this approach performs poorly with few training images and is more suitable to the modeling of classes of objects rather than individual 3D objects. Moreover it does not cope with multi-pose 3D object recognition.

The work in this paper follows the approach of Lowe [6], who addresses the problem of view clustering for 3D object recognition using a training set of images with uncluttered background. Each view is described by its SIFT local features and adjacent views are clustered together relying on feature matching and similarity transformation error. The presented method relies on a different geometric consistency check based on homographies which is capable of relating views under different perspectives with low false positive rates.

3 Object Extraction and Matching

A specialized local feature based background subtraction method [1] has been implemented to segment stable foreground objects from a video stream relying on their FAST keypoints. A 2-level background model is created and updated using temporal statistics on the positions of the keypoints. The first level is trained to segment keypoints in background and foreground, while the second level segments the foreground keypoints in moving or stationary keypoints. Stationary foreground keypoints are used to extract views of stable foreground objects from the video removing the part of the image containing keypoints coming from the cluttered background.

The system ciclycally a) updates the background model untils it is steady, b) waits for stable new objects to be detected in the scene, c) extracts the view of the detected object, d) compares it to already collected views and e) organizes cluster of views.

Each extracted view o_i is described by a) K_i, the set of the positions of its local features (keypoints) and b) D_i, the set of their extracted ORB descriptors. ORB is a rotation invariant version of the BRIEF binary descriptor based on binary tests between pixels of the smoothed image patch [8]. It is suitable for realtime applications since it is faster than both SURF and SIFT but it has similar matching performance and is even less affected by image noise [2].

3.1 Observations Matching

A similarity function $S : (o_1, o_2) \rightarrow [0, 1]$ is defined on a pair of object views (o_1, o_2), representing the quality of the visual match between them. The similarity value among two views is computed in steps shown in Figure 2 and described below.

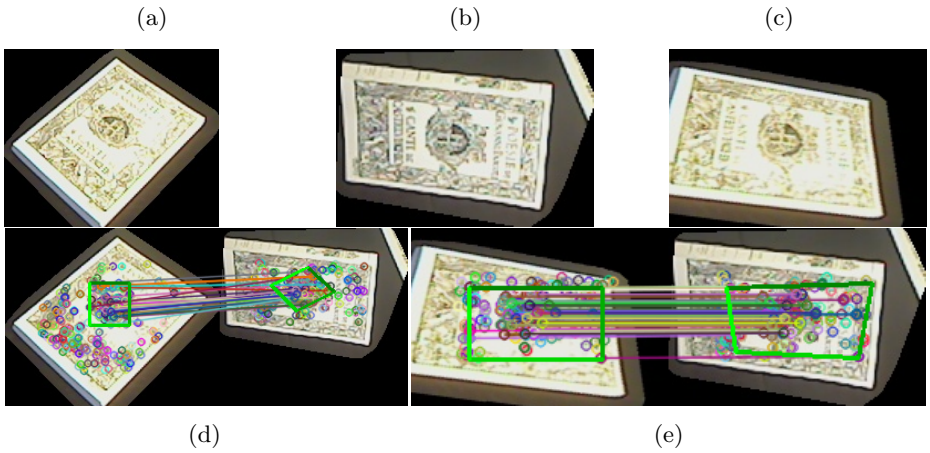


Fig. 2: Example of similarity computation among two different views (a) and (b) of the same 3D object. The homography relating the matching keypoints is shown in (d). The view (a) is transformed using the found homography in (c) and all matching steps are reapplied: the second homography found (e) confirms the match. The computed similarity value is 0.48*.*

Feature Matching Let K_1, K_2 be the sets of keypoints of the compared views and D_1, D_2 their sets of corresponding descriptors.

A preliminary list L of descriptor matches is created finding for each descriptor in D_1 its nearest neighbor in D_2 using the bruteforce method. Distances between descriptors are computed using the method suggested by the authors of the descriptor. In case of ORB, Hamming distance between the binary representation of the descriptors is used.

Matches in L are then filtered keeping only the ones having distance between descriptors below T_m. We chose $T_m = 64$ as suggested by Rublee et al. [8] despite not being the most stringent value to filter bad matches, but we preferred high recall of matches rather than high precision at this step.

If there are less than 4 matches left in the list, the following steps cannot be applied and the similarity value is set to 0.

RANSAC Filtering of Matches Bad matches in L are filtered out checking whether the points that match are geometrically consistent.

Two images of the same planar surface in space are related by a *homography* [3]. A homography is a invertible transformation represented by a 3×3 real matrix that maps the 2D coordinates of points in a image plane into the 2D coordinates in another plane.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Let $K_1^\star, K_2^\star$ be the sets of keypoints corresponding to the descriptors belonging to L. In order to find the homography that relates correctly the most of the points in $K_1^\star$ and $K_2^\star$, RANSAC is applied [3]. RANSAC is an non-deterministic algorithm to estimate parameters of a mathematical model from a set of observed data which contains outliers. RANSAC algorithms iteratevly executes the following steps:

1. takes 4 matches (couples of points) at random from $K_1^\star$ and $K_2^\star$,
2. computes the homography H relating those points,
3. counts the number of other matches that are correctly related by H (inliers).

After a certain number of iterations, the matrix H which gave the maximum number of inliers is returned.

Using the homography found by the RANSAC algorithm (Figure 2d), we can further filter the matches in L, keeping only the inliers of the perspective transformation.

Quasi-degenerate and flipping homographies can be detected analizing the homography matrix. Three checks are done:

- flipping homographies can be discarded checking if $det(H) < 0$.
- very skewed or prospective homographies can be discarded if $det(H)$ is too small or too big: given a parameter N, H is discarded if $det(H) > N$ or $det(H) < \frac{1}{N}$.
- homographies transforming the matching keypoints bounding box in a concave polygon can be filtered out with a convexity check.

In those cases, it is very unlikely that the views under analysis are really related by this perspective transformation, therefore the system assumes there is no similarity between them and returns a similarity value of 0.

Second Stage RANSAC Some views may pass the homography matrix check even if the perspective transform described by H is very unlikely to be observed. In order to filter out false positives homography matrices, the image of the first view o_1 is transformed in $\hat{o_1}$ using the homography to be validated (Figure 2c) and the similarity computation steps are repeated considering the views $\hat{o_1}$ and o_2. Features are re-detected and re-extracted from $\hat{o_1}$, matched with o_2 and a

second RANSAC is executed to estimate a new homography $\hat{H}$ describing the prospective transformation among $\hat{o_1}$ and o_2 4. If the original views o_1 and o_2 were really different views of the same object, $\hat{H}$ should be very near to the identity transformation (Figure 2e), otherwise the similarity between o_1 and o_2 is set to 0.

Similarity Output After the system found a good homography relating the views, the ratios $\hat{r_1}, r_2$ among the number of inliers and the total number of detected features are computed for each view:

$$\hat{r_1} = \frac{I}{|\hat{K_1}|}, \qquad r_2 = \frac{I}{|K_2|}$$

where I are the number of inliers of the homography estimated between views $\hat{o_1}$ and o_2, $|\hat{K_1}|$ and $|K_2|$ are respectively the number of detected keypoints in $\hat{o_1}$ and in o_2. The similarity value among original views under analysis $S(o_1, o_2)$ is defined as the harmonic mean between $\hat{r_1}$ and r_2 (Figure 3):

$$S(o_1, o_2) = \frac{2\hat{r_1} r_2}{\hat{r_1} + r_2}$$

	A	B	C	D
A	1	0	0	0.63
B	0	1	0.57	0
C	0	0.62	1	0
D	0.68	0	0	1

(a) (b) (c) (d) (e)

Fig. 3: Values of similarity among object views (a-d) reported in table (e).

4 Online Object Clustering

Everytime a new view of an object is gathered from the video stream, the system a) assigns it to a cluster and b) maintains clusters of views that potentially represent the same 3D object (Figure 4).

Each cluster is identified by a label assigned to views. The system puts a new view in a cluster relying on the similarity it has with other already clustered views, following an agglomerative clustering approach. The new view can bring informations useful to cluster reorganization: for example, let c_1 and c_2 be two clusters of views representing the same 3D object viewed from two different poses. An intermediate view of the 3D object could suggest the system to merge c_1 and c_2 in a unique cluster (see Figure 5).

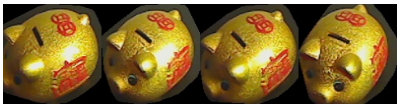
Given a new view $\hat{o}$, a list L_s of similar views is generated scanning the local database. For each object view o_i the similarity value $s_i = S(\hat{o}, o_i)$ is computed and if it is above a similarity threshold T_s, o_i is inserted in L_s.

When trying to label $\hat{o}$, the following scenarios can occur:

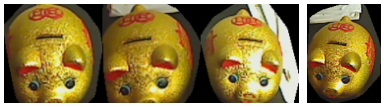
(a) table calendar

(b) poetry book

Fig. 4: Example of two object view clusters (a) and (b).



(a) side view cluster



(b) frontal view cluster (c)

Fig. 5: Example of cluster merging*: The new view (c) is similar to both clusters and can lead to a cluster merge.*

1. $\hat{o}$ does not match with any views, hence a new cluster is created and a new label is assigned to $\hat{o}$.
2. $\hat{o}$ matches with one or more views all belonging to the same cluster, hence the system assigns the corresponding cluster label to $\hat{o}$.
3. $\hat{o}$ matches with more than two views beloging to different clusters. Many actions may be taken by the system in this situation:
   (a) the clusters containing the views similar to the new one are merged together in a unique bigger cluster to which the new view will belong (Figure 5).
   (b) the new view is inserted into only one among the candidates clusters.
   (c) a new cluster is created containing only the new view.

Up to now, the system does not decide automatically in the third scenario and asks the user which action should be taken. Interaction between multiple cameras and similarity values between views and clusters may be exploited to take the correct action automatically, but are not discussed in this paper and are left to future work.

In the case a new view is incorrectly put in a new cluster instead of being grouped with the other views representing the same object, the agglomerative cluster algorithm can eventually build a unique cluster if intermediate views of the same object will be collected by the system.

5 Experiments

The presented system autonomously groups object views into clusters without knowing their labels, but cannot recognize them before the user labels at least some of them, hence the system cannot be compared with traditional trained classifiers. Instead the ability of the system to build good and easy to label clusters is measured.

To do so, the publicly available *3D Objects* dataset [9] has been used. This dataset is composed by images of 10 object categories. For each category, 9-10

Fig. 6: Object learning and recognition in a test video sequence: objects are added, moved and removed from the scene. The system segments objects from the background and incrementally creates clusters of similar object views. An object is recognized assessing the membership of its current view to a pre-existent cluster.

objects are present and for each object, several images are reported in which the specific object is shown in different poses. Each image comes with a foreground mask which denotes exactly in which part of the image the object is located. Images are taken from 8 different angles using 3 different scales and 3 different heigths for the camera, leading to around 5500 labeled images of 100 specific objects (see Figure 7).

Category	**Object**	**Views**
cellphone	cellphone_1	 ...
	...	...
	cellphone_9	 ...
mouse	mouse_1	 ...
	...	...
toaster	toaster_1	 ...
	...	...

Fig. 7: Excerpt from the Stanford "3D Objects" dataset: only some views of some objects of some classes are reported.

Let $O = \{(o_1, l_1), (o_2, l_2), \ldots\}$ the set of labeled views. The entire dataset O is randomly shuffled and splitted in training set O_{train} (90%) and testing set O_{test} (10%): training views are presented to the system as coming from the output of the foreground extraction stage. The system builds clusters of views while they are processed. In the case a supervised clustering is needed, the test code uses the groundtruth labels of involved views to simulate user interaction applying Algorithm 1.

Once the clusters are built, they must be labeled to produce a labeled training set. Since the user usually does not want to waste time in cleaning clusters or labeling singular objects, the test code simulates a labeling technique based on *major voting*: an entire cluster is labeled with the label of the most frequent object present in it.

The training set thus labeled is used for training a k-NN classifier. The *cluster* k-NN classifier finds the k most similar views (the ones with the higher value

Algorithm 1 Clustering algorithm simulating user interaction used for evaluation tests

for all $(o_i, l_i) \in O_{train}$ **do**
  find the set O_s of views similar to o_i, $O_s = \{o_j \in Database : S(o_i, o_j) > T_s\}$
  if $|O_s| = 0$ **then**
    put o_i in a new cluster
  else if $|O_s| = 1$ **or** (all views in O_s belong to the same cluster) **then**
    put o_i in the cluster of the similar views
  else ▷ simulate user interaction
    find set C_s of all clusters to which the similar views belong
    for all $c \in C_s$ **do**
      find the majority groundtruth label of c (the label appearing the most in the cluster)
    end for
    create a new cluster merging together all clusters having its majority label equal to l_i (the label of o_i)
    put o_i in the newly created cluster
  end if
end for

of similarity S) and assigns a score for each label of those views. The winning label is assigned to the processed test view. Another k-NN classifier is trained using the training set with correct labels and another labeling of the test set is generated in the same way.

Test set labelings are evaluated extracting precision, recall and F-score for each 3D object and then aggregating them using macro- and micro-averaging techniques defined as follows:

	micro-avgd	**macro-avgd**
precision	$p_{micro} = \frac{\sum_{i=1}^{n} \mathrm{TP}_i}{\sum_{i=1}^{n} (\mathrm{TP}_i + \mathrm{FP}_i)}$	$p_{macro} = \frac{\sum_{i=1}^{n} p_i}{n}$
recall	$r_{micro} = \frac{\sum_{i=1}^{n} \mathrm{TP}_i}{\sum_{i=1}^{n} \mathrm{P}_i}$	$r_{macro} = \frac{\sum_{i=1}^{n} r_i}{n}$
f-score	$F_{micro} = \frac{2 p_{micro} r_{micro}}{p_{micro} + r_{micro}}$	$F_{macro} = \frac{2 p_{macro} r_{macro}}{p_{macro} + r_{macro}}$

where n is the number of object classes, TP_i the true positives, FP_i the false positives, P_i the total number of views, p_i the precision, r_i the recall and F_i the F-score of the object i.

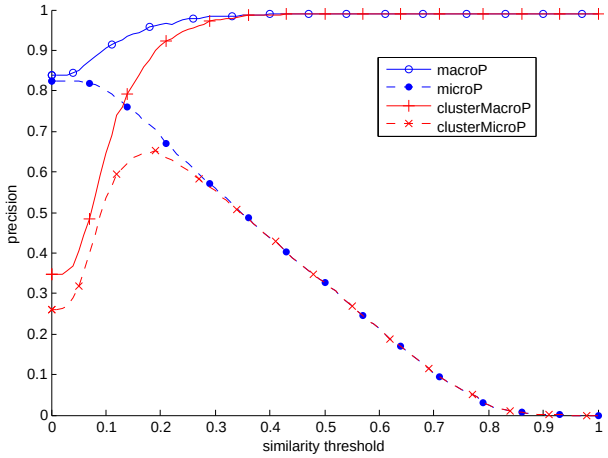
Macro-averaged metrics tends to give the same weight to each class, while micro-averages metrics takes into account possible biases introduced by each class and gives a more accurate global performance index. Another measured metric is the number of interactions the system must have with the user in order to label the training set: the *groundtruth* k-NN classifier needs the user to label each training view individually, which corresponds to a number of query to the user equal to the number of views in the training set. The *cluster* k-NN

classifier needs to interact with the user a) when a cluster merging can not be resolved automatically during the online clustering and b) when a cluster has to be labeled.

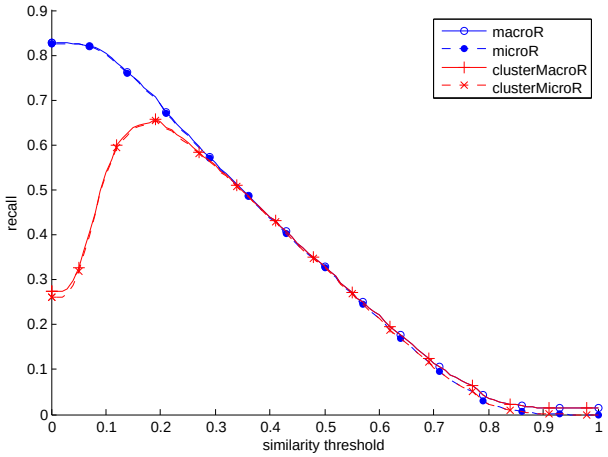
6 Conclusions

In Figure 8, the performance of the two classifiers for various similarity thresholds T_s are reported.

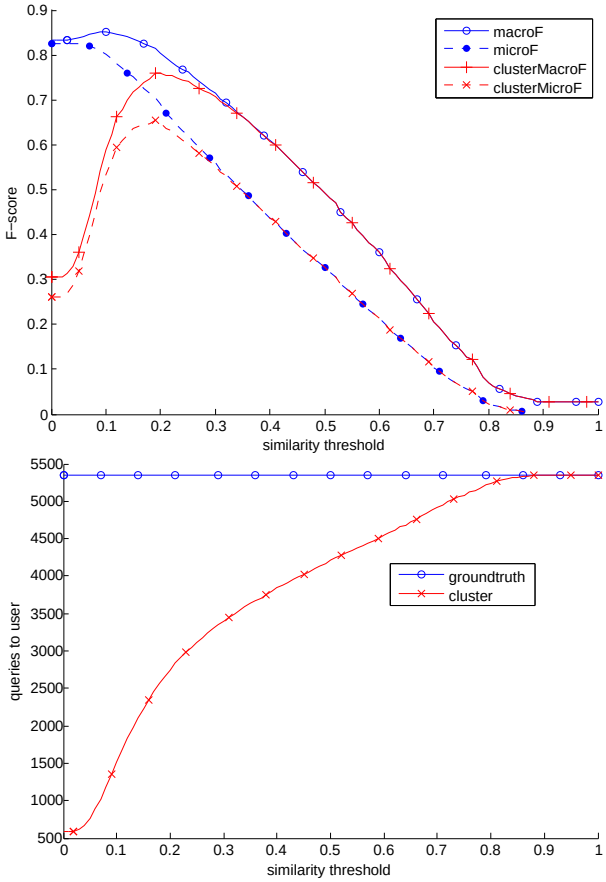
It can be seen that for T_s around 0.2, the *cluster* k-NN classifier has almost the same performance of the *groundtruth* k-NN classifier, having only around half the interactions with the user.



(a) Micro- and macro-averaged precision values when varying the similarity threshold



(b) Micro- and macro-averaged recall values when varying the similarity threshold



(c) Micro- and macro-averaged F1-score values when varying the similarity threshold

(d) The number of queries to the user when varying the similarity threshold

Fig. 8: Comparison of the performance of the recognition task, solved by a k-NN classifier trained with the groundtruth training set (blue lines with circle markers) and by a k-NN classifier with training set made by cluster labeling (red lines with cross markers). Solid and dashed lines indicate respectively micro- and macro-averaged metrics.

However, performance degradation of the *cluster* k-NN classifier is due to the fact that we simulated a unique user interaction after the training phase which used *major voting* paradigm to label all clusters at once. Since the system is incrementally building richer and richer clusters, this is not the best way to interact with the user asking for labels: user interaction may be proactively requested only when big homogeneous clusters are involved, maximizing the amount of information collected. Moreover, smarter techniques than major voting may be implemented to simulate a more precise user labeling session. In the performed tests, many singleton or small clusters are present at the end of the training phase, raising the number of queries to the user needed to label the entire training set.

Bibliography

[1] Carrara, F., Amato, G., Falchi, F., Gennaro, C.: Efficient foreground-background segmentation using local features for object detection. In: Proceedings of the International Conference on Distributed Smart Cameras, ICDSC '15, September 08 - 11, 2015, Seville, Spain (submitted for publication), available at `http://puma.isti.cnr.it/rmydownload.php?filename=cnr.isti/cnr.isti/2015-TR-012/2015-TR-012.pdf`

[2] De Beugher, S., Brône, G., Goedemé, T.: Automatic analysis of in-the-wild mobile eye-tracking experiments using object, face and person detection. In: Proceedings of the international conference on computer vision theory and applications (VISIGRAPP 2014). vol. 1, pp. 625–633 (2014)

[3] Dubrofsky, E.: Homography estimation. Ph.D. thesis, University of British Columbia (2009)

[4] Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding 106(1), 59–70 (2007)

[5] Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. vol. 2, pp. II–264. IEEE (2003)

[6] Lowe, D.G.: Local feature view clustering for 3d object recognition. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I–682. IEEE (2001)

[7] Murase, H., Nayar, S.K.: Visual learning and recognition of 3-d objects from appearance. International journal of computer vision 14(1), 5–24 (1995)

[8] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 2564–2571. IEEE (2011)

[9] Savarese, S., Li, F.F.: 3d generic object categorization, localization and pose estimation. In: ICCV. pp. 1–8 (2007)

[10] Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. Springer (2000)