Efficient object recognition using sampling of keypoint triples and keygraph structure

Estephan Dazzi*, Teofilo de Campos†‡, Adrian Hilton‡ and Roberto M. Cesar - Jr*
*Instituto de Matemática e Estatística - IME
Universidade de São Paulo - USP, São Paulo - SP, Brasil
Email: e.dazzi@ime.usp.br, rmcesar@usp.br
†Universidade de Brasília - Faculdade Gama. Gama, DF, Brasil
‡Centre for Vision, Speech and Signal Processing - CVSSP
University of Surrey, Guildford, UK
Email: {t.decampos, a.hilton}@surrey.ac.uk

***Abstract*—We present an object matching method that employs matches of local graphs of keypoints, called keygraphs, instead of simple keypoint matches. For a keygraph match to be valid, vertex (keypoint) descriptors must be similar and both keygraphs must satisfy structural properties concerning keypoints orientation, scale, relative position and cyclic ordering; as a result, the large majority of initial incorrect keypoint matches is correctly filtered out. We introduce a novel approach to sample keypoint triples (i.e. keygraphs) in a query image, based on complementary Delaunay triangulations; this generates a linear number of triples with relation to the number of keypoints. Query keygraphs are then matched against the indexed model keypoints; each established keygraph match is used to evaluate a candidate pose (an affine transformation). The proposed method has been evaluated for object recognition and pose estimation, achieving a better performance in comparison to state-of-the-art methods.**

***Keywords*-Local image feature matching; semi-local graph matching; graph topological properties.**

I. Introduction

Many problems in computer vision involve matching an image against a large database of images. For instance, the problem of 3D object recognition can be cast as one of image matching, by storing images of a range of views for each model object and then treating a test image as a query.

One of the most succesful approaches for image matching is the local feature framework [1], which extracts interest points (keypoints) from images. Currently, most methods on the literature follow the strategy of establishing correspondences between individual keypoints (*e.g.* [2], [3], [4], [5]). However, establishing simple one-to-one keypoint correspondences disregards structural aspects contained in neighborhoods of keypoints in images. Recent work on the literature has attempted to use such a structural information (*e.g.* [6], [7], [3]), but in limited ways.

This paper presents an improved approach to deal with structural organisations of keypoints in images. We substitute matches between keypoints by matches between *local graphs of keypoints*. Each image is represented by a set of local, small graphs, whose vertices are keypoints and whose edges contain structural information; these local keypoint graphs are called *keygraphs*. Correspondences are then established between keygraphs in the query image and keygraphs in the model images. Establishing a match between graphs encompasses not only keypoint descriptor information (vertex matches) but also requires both graphs to satisfy structural properties (see Figure 1). We consider keygraphs composed of three vertices, due to the favorable structural and topological properties of keypoint triples.

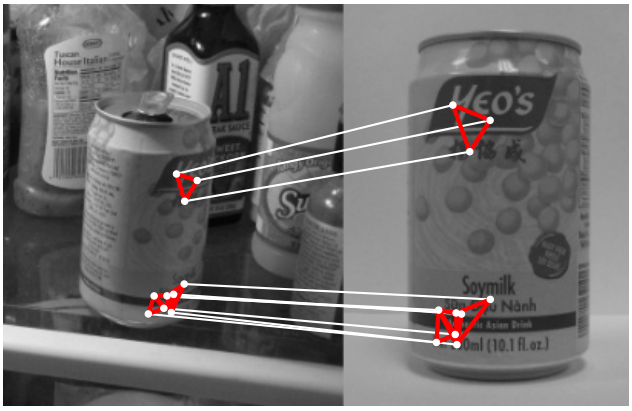

Figure 1. Keygraph (*i.e.* keypoint triplet) matches between a query image (left) and a model image (right). Keygraphs are sampled in the query image by using complementary Delaunay triangulations and then matched against the model keypoints. A valid keygraph match must satisfy structural properties concerning keypoints orientation, scale, relative position and cyclic ordering.

Our first main contribution is a novel set of graph structural properties which are used to filter out incorrect candidate keygraph matches. Keygraph structure concerns keypoints orientation, scale, relative position and cyclic ordering. The filtering stage correctly rejects more than 99% of the initial candidate keygraph matches, which improves efficiency and performance in the final pose estimation stage.

Our second main contribution is an algorithm to sample

triples of keypoints (*i.e.* keygraphs) in a query image. Initially, we sample a subset of keypoints such that no two keypoints are too close. Then, a Delaunay triangulation is constructed, producing triples of neighbouring keypoints. This process is repeated T times; keypoint subsets are selected in a pairwise complementary way. For a query image with n keypoints, running the algorithm has complexity $O(Tn \log n)$ and produces $O(Tn)$ keypoint triples.

Keygraphs in the query image are matched against indexed [8] model keypoints; each established keygraph match generates a candidate object pose (an affine transformation). Thus the number of pose evaluations grows *linearly* with the number of keypoint matches between a pair of images.

Keygraphs constitute an intermediate level feature, above the keypoints. Keygraphs can be used with any keypoint detector that assigns scale and orientation to every keypoint. In this paper, we employ SIFT [2] features.

We evaluated our method for object recognition and localisation. A better performance was achieved in comparison to state-of-the-art methods [2], [3], [4], particularly in a scalable scenario, where many model images were used.

II. Related work

The concept of keygraphs was introduced by Hashimoto *et al.* [9] and used for indoors self-localisation using sign boards by Morimitsu *et al.* [10]. Fourier coefficients of keygraph edges were used as local descriptors, providing a high efficiency. However, in comparison to our method, Morimitsu *et al.* only employed cyclic ordering for match filtering and used a single Delaunay triangulation to generate query keygraphs. Also, all the model keygraphs must be stored in memory during application time, which imposes a limit on the number of model images. On the other hand, we propose a keygraph based method that uses a large set of model images.

Philbin *et al.* [5] propose an image retrieval method which uses affine-invariant keypoints [1]. This allows a single *keypoint* match to carry enough information to instantiate a candidate pose, while we use a single *keygraph* match for that. There are two main advantages of using keygraph matches instead of keypoint matches. First, more robust candidate object poses are generated, since three image regions are employed for pose generation. Second, rich structural information can be used for match filtering, which is not possible in case only one point is matched.

Hsiao *et al.* [4] present a 3D object recognition method which uses 3D object models built using structure-from-motion. Keypoints are extracted from many artificially distorted model images and then triangulated into the 3D model, which augments the set of model keypoints.

Sattler *et al.* [3] explore structural information in keypoint neighbourhoods. The idea is that a correct keypoint match usually occurs close to other correct keypoint matches in the image. Thus each keypoint match (p, q) is required to have some neighbour matches around p (in the query image) and around q (in the model image) agreeing about the matched image; otherwise, (p, q) is filtered out. We also explore the fact that a correct keypoint match is likely to be close to other correct keypoint matches in the image. A keypoint triple in the query image is composed of neighbour keypoints, which increases the chance that all the three keypoint matches are simultaneously correct.

Info / Method	[3]	[6]	[7]	[11]	[10]	Ours
Cyclic ordering			X		X	X
Keypoints scale		X	X	X		X
Keypoint orientation				X		X
Keypoint relative position		X	X			X
Keypoint neighbourhood	X	X			X	X

Table I
STRUCTURAL AND TOPOLOGICAL INFORMATION USED IN MATCHING

Hao *et al.* [6] present a 3D object recognition method. First, a filtering stage, similar to the method by Sattler *et al.* [3], is employed. Then, another structural property is evaluated: for every pair of matches, the method requires agreement between distances in the 3D model and keypoint scales. However, in comparison to our method, [6] neither employs cyclic ordering nor keypoint orientation for keypoint match filtering, which decreases efficiency.

Hao *et al.* [7] employ triples of keypoint matches, similarly to the proposed method. For keypoint match filtering, the authors use structural information concerning keypoints scale, relative position and cyclic ordering. Before application time, the method in [7] explicitly stores a large number of model keypoint triples, which are then matched against the keypoints in a query image. The method in [7] differs from ours in three main aspects. First, we explore the use of local neighborhood of keypoints. Second, our method additionally uses keypoint orientation for match filtering. Third, in [7], a large memory space is needed to store a sufficiently diverse set of model keypoint triples. On the other hand, in the proposed method, every keygraph is efficiently generated during execution time.

Jégou *et al.* [11] presented an image retrieval method which uses structural information for keypoint match filtering, by requiring consistency in keypoints scale and orientation. However, the method neither uses cyclic ordering nor verify the consistency of keypoints scale or orientation with their relative position.

Table II summarizes the previous discussion concerning the structural information used for keypoint match filtering. Our method employs a more heterogeneous set of information, which provides a better performance in filtering.

III. Proposed method

Object recognition is carried out by finding affine transformations mapping the query image to one or more model images. Before application time, the model keypoints are

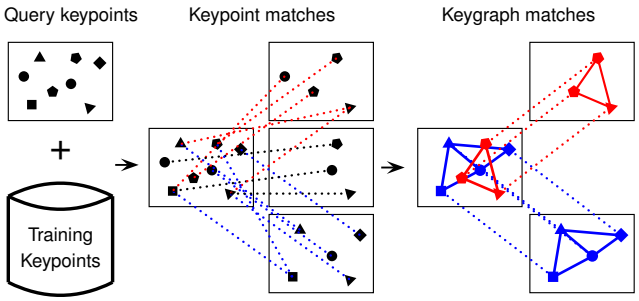

Figure 2. Establishing keygraph matches between a query image and model images. Query keypoints are matched against indexed model keypoints, establishing keypoint matches. Then, keypoint matches are transformed into keygraph matches, which correctly eliminates the large majority of the initial keypoint matches.

extracted and indexed [8]. During application time, our method follows this pipeline:

1) *Keypoint matching*: Each keypoint in the query image runs through a hierarchical k-means tree [8], matching keypoints in many model images (see Figure 2).
2) *Keygraph matching*: Keypoint triples are sampled in the query image (see Figure 3) and then matched against the model images, by using the previously obtained keypoint matches (see Figure 2). This generates candidate keygraph matches. Each candidate keygraph match must satisfy the keygraph structural properties in order to be considered as valid (see Figure 4).
3) *Pose estimation*: The remaining keygraph matches are used to search for affine transformations mapping query and model images. Each keygraph match generates a candidate transformation that is evaluated by counting the number of keygraph vertex matches (*i.e.* filtered keypoint matches) that agree with it.

A. First stage: keypoint matching

We employ a hierarchical k-means tree [8] in order to index model keypoints. The tree is built by recursivally applying k-means to the descriptor space; each leaf node stores a set of (less than k) model keypoints.

During application time, each query keypoint p runs through the tree in order to match stored model keypoints. p is compared to a stored keypoint q by calculating the Euclidean distance between their SIFT descriptors [2]. We let p establish up to *one* match with a keypoint in *each* model image, namely the match with the lowest Euclidean distance. The parameter L sets the number of keypoint comparisons performed between a query keypoint p and stored model keypoints.

B. Second stage: keygraph matching

In the second stage, keygraphs are obtained from the query image and then matched againt the model images.

A keygraph is defined as a graph $G = (V, E)$, where the vertex set V is composed of keypoints extracted from the same image, and E is the set of graph edges. Every keygraph G has κ vertices and consists is an *oriented circuit in the counter-clockwise direction*, $G = (v_1, v_2, \ldots, v_\kappa)$ [10].

We employ keygraphs with $\kappa = 3$ vertices, which allows the use of cyclic ordering and makes *one* keygraph match be sufficient to instantiate an affine transformation.

1) Keygraph sampling in a query image: Obtaining keygraphs from a query image involves sampling from the set of all possible triples of keypoints. The proposed method follows a strategy based on using complementary Delaunay triangulations, which produces a *linear* number of triples. In order to avoid using keygraphs with very short edges, which would increase sensitivity to noise, the method samples a keypoint subset $\mathcal{S} \subseteq \mathcal{P}$ from the original set $\mathcal{P}$ of query keypoints such that no keypoints in $\mathcal{S}$ are very close to each other in the image. As $|\mathcal{S}|$ can possibly be much smaller than $|\mathcal{P}|$, we use T different subsets, $\mathcal{S}_1, \ldots, \mathcal{S}_T$. In order to select the first keypoint subset $\mathcal{S}_1 \subseteq \mathcal{P}$, we start with $\mathcal{S}_1$ as an empty set. Then, keypoints p_j are randomly accessed in $\mathcal{P}$, and a tentative is made to include each p_j in the set $\mathcal{S}_1$ being constructed. The construction of $\mathcal{S}_1$ stops after all the keypoints in $\mathcal{P}$ have been accessed. To include a keypoint p_j in $\mathcal{S}_1$, the ℓ_∞ distance, in pixels, between p_j and every keypoint already included in $\mathcal{S}_1$ must be above $\delta = 8$ pixels.

Random selection of keypoints is implemented by creating an array A of keypoints from $\mathcal{P}$, and then randomly permuting the elements in A. Then, for selecting the first keypoint subset $\mathcal{S}_1$, the sequence of accessed keypoints is obtained by starting from the *first* position of the array A and then increasing one position at a time until the last element of the array is accessed. Each time an array position j is accessed, the method tries to include the keypoint p_j stored in this position into the set $\mathcal{S}_1$ being constructed.

After the creation of the first keypoint subset $\mathcal{S}_1 \subseteq \mathcal{P}$, the next subset $\mathcal{S}_2 \subseteq \mathcal{P}$ is constructed. The same array of permuted keypoints A that was used for $\mathcal{S}_1$ is also used for $\mathcal{S}_2$, but, for $\mathcal{S}_2$, the elements of A are accessed starting from the *last* position of the array and then decreasing one position at a time until the first position of A is achieved.

The complexity of selecting a keypoint subset $\mathcal{S}_i$ is $O(n)$, where n is the number of keypoints in the query image. Permuting the positions of a keypoint array A has complexity $O(n)$. Then, keypoints from A are included into $\mathcal{S}_i$; trying to include a keypoint has complexity $O(1)$, thanks to the use of a grid of squares with cell length $\delta = 8$ pixels[1].

To obtain T keypoint subsets, the procedure described to obtain $\mathcal{S}_1$ and $\mathcal{S}_2$ is simply repeated. Each pair of subsets $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$ employs the same array A_i of permuted keypoints.

Finally, each keypoint subset $\mathcal{S}_i$ is used to generate a Delaunay triangulation. $\mathcal{S}_i$ produces the set of keypoint triples $\mathcal{D}_i = \{(p_1, p_2, p_3), (r_1, r_2, r_3), \ldots\}$, with each triple in $\mathcal{D}_i$

[1]Including a keypoint p in a grid cell demands checking if this cell is already occupied by a keypoint and then checking the ℓ_∞ distance between p and a (possible) keypoint in each of the eight neighbouring grid cells.

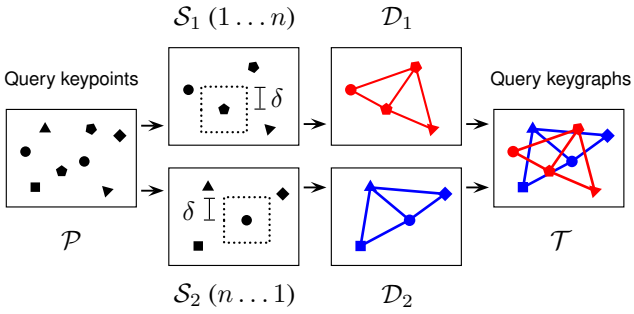

Figure 3. Keypoint triple sampling in a query image. First, query keypoints $\mathcal{P}$ are detected. Then, complementary keypoint subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ are obtained (using a minimum ℓ_∞ distance between keypoints of δ pixels) and employed to calculate Delaunay triangulations $\mathcal{D}_1$ and $\mathcal{D}_2$. The triples $\mathcal{T} = \mathcal{D}_1 \cup \mathcal{D}_2$ are returned.

being a keygraph. Calculating the Delaunay triangulation $\mathcal{D}_i$ has complexity $O(n \log n)$ and produces $O(n)$ triples.

The set of all keypoint triples is given by $\mathcal{T} = \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_T$. Every keypoint triple in $\mathcal{T}$ is included in a hash table for checking in $O(1)$ time whether that triple is repeated.

In summary, calculating the set $\mathcal{T}$ of keygraphs in a query image has complexity $O(Tn \log n)$. This produces a number of keygraphs (keypoint triples) $|\mathcal{T}| = O(Tn)$.

2) Matching query and model keygraphs: First, we calculate potential keygraph matches that may occur, based on the initial keypoint matches. Then, (incorrect) keygraph matches are filtered out by using structural information.

Obtaining initial candidate keygraph matches. A keygraph in the query image can establish up to *one* match with each model image. Let $G = (v_1, v_2, v_3)$ be a query keygraph; G can be matched in all images where all its vertices have reported correspondences. During the keypoint matching stage, assume that the keypoint matches (v_1, u_1), (v_2, u_2) and (v_3, u_3) are established between the query image and a model image I. Thus the candidate keygraph match between G and the image I is (G, H), where $H = (u_1, u_2, u_3)$ is a candidate keygraph in the model image I. Therefore, the keygraph match (G, H) is associated to the keypoint matches $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$.

Implementing this initial stage involves checking, for each query keygraph $G = (v_1, v_2, v_3)$, which model images have keypoint matches with v_1, v_2 and v_3. To calculate this, we use, for each query keypoint p, words of 64 bits indicating which model images are matched by p. Thus the images initially matched to $G = (v_1, v_2, v_3)$ are obtained by using a AND binary operation between the words for v_1, v_2 and v_3; the 1's in the resulting binary words can be accessed by using count-leading-zeros machine instructions.

Keygraph match filtering using structural information. Every keygraph match (G, H) must satisfy the graph structural properties in order to be considered as a valid match.

Cyclic ordering is a topological property, invariant to perspective transformation in 2D images. Every keygraph G in a query image is a circuit (a triple) oriented in the counter-clockwise direction. Since it is assumed that mirroring is not a possible distortion of the query image, the circuit of a matched keygraph H in a model image has to be oriented in the counter-clockwise direction as well.

Four other attributes are calculated for each keygraph:

- *Edges length*: A keygraph edge is a straight line connecting two vertices with length expressed in pixels. A keygraph $U = (h_1, h_2, h_3)$ has three edges, $e_{12} = (h_1, h_2)$, $e_{23} = (h_2, h_3)$ and $e_{31} = (h_3, h_1)$, whose lengths are l^U_{12}, l^U_{23} and l^U_{31} (see Figure 4-a).
- *Keypoints scale*: Each keypoint in $U = (h_1, h_2, h_3)$ has a scale assigned by SIFT. They are denoted as s^U_1, s^U_2 and s^U_3, respectively (see Figure 4-b).
- *Edges orientation*: Each keygraph edge forms an angle with the x axis of the image (a keygraph edge is treated as a regular vector in 2D space). The orientations of the edges of a keygraph $U = (h_1, h_2, h_3)$ are denoted as α^U_{12}, α^U_{23} and α^U_{31} (see Figure 4-c).
- *Keypoints orientation*: Each keypoint in $U = (h_1, h_2, h_3)$ has an orientation assigned by SIFT. They are denoted as o^U_1, o^U_2 and o^U_3 (see Figure 4-d).

For each candidate keygraph match (G, H), our method calculates the changes in orientations, lengths and scales from G to H. A valid keygraph match must present a similar change in edges orientation and keypoints orientation, as well as a similar change in edges length and keypoints scale.

Let the candidate keygraph match (G, H) between $G = (v_1, v_2, v_3)$ and $H = (u_1, u_2, u_3)$ be associated to the keypoint matches $\mathcal{M} = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$, as illustrated by Figure 4-e. Attribute changes are calculated:

- *Changes in edges length*: Three ratios between the length of corresponding edges in the match (G, H): $\Phi_{12} = l^G_{12}/l^H_{12}$, $\Phi_{23} = l^G_{23}/l^H_{23}$, $\Phi_{31} = l^G_{31}/l^H_{31}$;
- *Changes in keypoints scale*: Three ratios between the scale of corresponding keypoints in the match (G, H): $\Phi_1 = s^G_1/s^H_1$, $\Phi_2 = s^G_2/s^H_2$, $\Phi_3 = s^G_3/s^H_3$;
- *Changes in edges orientation*: Three changes in orientation of corresponding edges in the match (G, H): $\Delta_{12} = \alpha^G_{12} - \alpha^H_{12}$, $\Delta_{23} = \alpha^G_{23} - \alpha^H_{23}$, $\Delta_{31} = \alpha^G_{31} - \alpha^H_{31}$;
- *Changes in keypoints orientation*: Three changes in orientation of corresponding keypoints in (G, H): $\Delta_1 = o^G_1 - o^H_1$, $\Delta_2 = o^G_2 - o^H_2$, $\Delta_3 = o^G_3 - o^H_3$.

Similar changes in keypoints scale and keygraph edges length. For a keygraph match (G, H), let the three ratios between the length of corresponding edges be Φ_{12}, Φ_{23} and Φ_{31}, while the three ratios between the scale of corresponding keypoints are Φ_1, Φ_2 and Φ_3. As illustrated in Figure 4-e, a valid keygraph match must satisfy the following property: for any pair Φ', Φ'' of those six ratios $\{\Phi_{12}, \Phi_{23}, \Phi_{31}, \Phi_1, \Phi_2, \Phi_3\}$, the largest ratio must be lowert

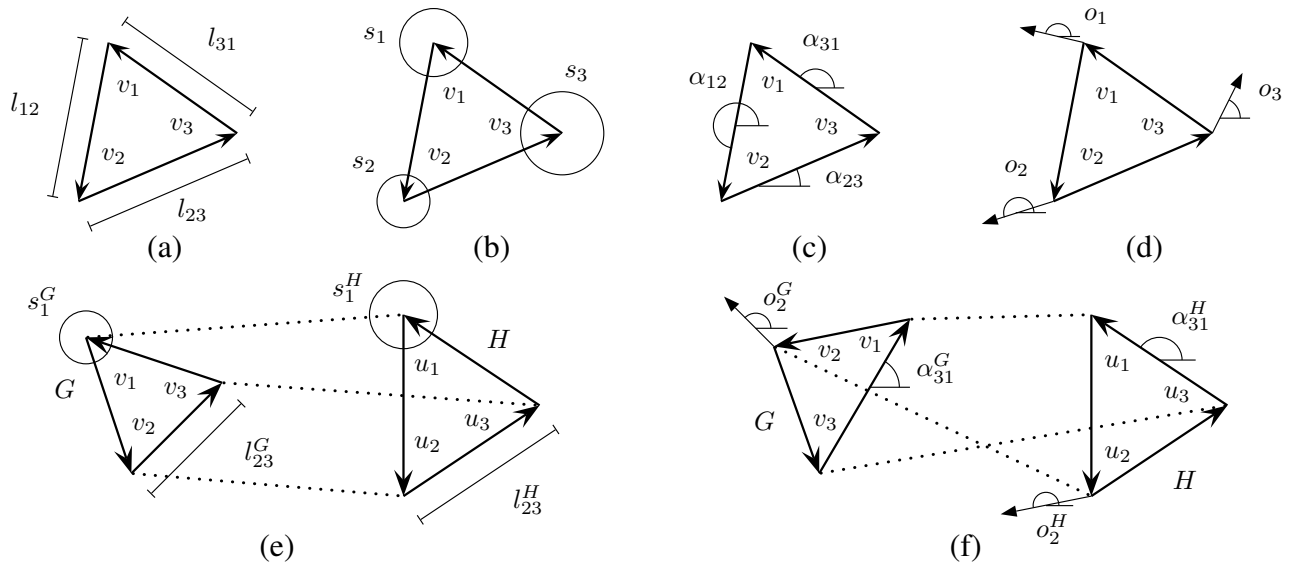

Figure 4. Structural information employed in keygraph match filtering. **(a)** Keygraph edges length: l_{12}, l_{23} and l_{31}. **(b)** Keypoints (SIFT) scale: s_1, s_2 and s_3. **(c)** Keygraph edges orientation: α_{12}, α_{23} and α_{31}. **(d)** Keypoints (SIFT) orientation: o_1, o_2 and o_3. **(e)** Checking whether changes in keypoints scale and keygraph edges length are similar. For the corresponding keypoints scale s_1^G (in a query keygraph G) and s_1^H (in a model keygraph H), the ratio $\Phi_1 = s_1^G/s_1^H$ is calculated, as well as the ratio between the length of corresponding edges $\Phi_{23} = l_{23}^G/l_{23}^H$. We check whether Φ_1 is similar to Φ_{23}: $0.5 \leq \Phi_1/\Phi_{23} \leq 2$. A similar check is done for every pair Φ', Φ'' from the set of ratios $\{\Phi_1, \Phi_2, \Phi_3, \Phi_{12}, \Phi_{23}, \Phi_{31}\}$. **(f)** Checking whether changes in keypoints orientation and keygraph edges orientation are similar. For the corresponding keypoints orientation o_2^G and o_2^H, the change in orientation $\Delta_2 = o_2^G - o_2^H$ is calculated, as well as the change in orientation of the corresponding edges $\Delta_{31} = \alpha_{31}^G - \alpha_{31}^H$. We check whether Δ_2 is similar to Δ_{31}: $\arccos(\Delta_2 - \Delta_{31}) \leq 60°$. A similar check is done for every pair Δ', Δ'' from the set of changes in orientation $\{\Delta_1, \Delta_2, \Delta_3, \Delta_{12}, \Delta_{23}, \Delta_{31}\}$.

than than twice the smaller ratio, *i.e.*[2]

$$0.5 \leq \frac{\Phi'}{\Phi''} \leq 2. \quad (1)$$

Similar changes in keypoints orientation and keygraph edges orientation. For the keygraph match (G, H), let the changes in keygraph edges orientation be Δ_{12}, Δ_{23} and Δ_{31}, while the changes in keypoints orientation are Δ_1, Δ_2 and Δ_3. As illustrated in Figure 4-f, a valid keygraph match must satisfy the following property: for any pair Δ', Δ'' of those six changes in orientation $\{\Delta_{12}, \Delta_{23}, \Delta_{31}, \Delta_1, \Delta_2, \Delta_3\}$, the angle between Δ' and Δ'' must be less than $60°$, *i.e.*[3],

$$\arccos(\Delta' - \Delta'') \leq 60°. \quad (2)$$

C. Third stage: pose estimation

In the third stage, the remaining keygraph matches are used to localise objects and estimate their pose.

[2]A perspective transformation changing the viewing angle in ψ degrees makes a unit circle become an ellipse whose longer and shorter axes have length 1 and $\cos\psi$, respectively [3]. SIFT features lose reliability when $\psi > 60°$ [2]; when $\psi = 60°$, the length of the transformed ellipse's shorter axe divided by the original circle's diameter is $\cos 60°/1 = 0.5$.

[3]Rotating an image by θ degrees changes the orientation of every keypoint and keygraph edge in this image in the same θ degrees. On the other hand, when a perspective transformation is applied, not every keypoint and keygraph edge rotates in the same θ degrees, although very different changes in orientation cannot occur.

One keygraph match generates $\kappa = 3$ keypoint matches, which can be used to instantiate a candidate affine transformation mapping query and model image. Each candidate pose is then evaluated by counting the number of keypoint (vertex) matches that agree with it.

Affine transformations with agreeing keypoint matches. Let $\mathcal{G}$ be the set of keygraph matches between the query image and a model image I, $\mathcal{G} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{|\mathcal{G}|}\}$, in which $\mathcal{M}_i$ is a set of three keypoint matches (associated to a keygraph match). Thus the set of *keypoint* matches between those images is $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_{|\mathcal{G}|}$.

To evaluate the quality of a candidate affine transformation, the method counts the number of matches that agree with this transformation: for each keypoint match $(p, q) \in \mathcal{M}$, let x_p, y_p be the position in the query image as established by the match, and let $f(x_q), f(y_q)$ be the position in the query image as predicted by the affine transformation. If the ℓ_∞ distance between x_p, y_p and $f(x_q), f(y_q)$ is below *three* pixels, the keypoint match agrees with the candidate transformation. The confidence in a solution is given by such a number of keypoint match agreements.

Each affine transformation with at least one keypoint agreement is considered as a possible solution. Each possible solution is further refined using least-squares fit.

Dealing with multiple detections. The final stage involves projecting the objects and dealing with multiple detections.

First, the candidate affine transformations are sorted based on the number of agreeing keypoint matches (*i.e.* confidence). The most confident solution is returned as a correct pose, and the ground-truth of its matched model image is projected in the query image.

Then, the next more confident solution re-counts its agreeing keypoint matches, now discarding matches lying inside the projection of any previously returned solution. If at least one keypoint agreement remains, that solution is returned as a correct pose and the ground-truth of its model image is projected in the query image. The procedure continues by similarly examining the remaining elements in the sequence of produced possible solutions.

IV. Experiments and results

For performance evaluation, we used the CMU10 object recognition dataset made available by Hsiao *et al.* [4]. This dataset contains ten types of model objects, for a total of 250 model images and 500 query images.

We also evaluated the scalability of our method, by significantly increasing the number of model images. We randomly selected a subset of 4000 images from the PASCAL VOC 2007 dataset [12] and employed them as model images together with the CMU10 model images. Detecting a PASCAL image constitutes an error.

SIFT keypoints were extracted using the VLFeat [13] library. The CMU10 and CMU10+PASCAL dataset generated 2.5×10^5 and 7.5×10^6 model keypoints, respectively.[4] For each dataset, the keypoints were indexed in a hierarchical k-means tree (both using $k = 16$).

For each detection, we used the recovered affine transformation to project the model object's ground truth segmentation onto the query image and calculated the region R inside the convex hull. The region overlap criterion $(R \cap R_{\mathsf{gt}})/(R \cup R_{\mathsf{gt}}) > 0.4$ between the region R and the ground truth segmentation R_{gt} (in the query image) determines if an object is correctly detected.[5]

To evaluate the results we used the mean Average Precision (AP), computed over all ten object classes of this dataset.

A. Baseline systems

We evaluated our method against SCRAMSAC [3], Lowe's test [2] and the results reported by Hsiao *et al.* [4].

[4]For the CMU10 images, we used ground-truth segmentation to remove keypoints not belonging to the object. For the PASCAL images, ground-truth was not used, thus all parts of an image act as a model "object".

[5]Hsiao *et al.* [4] projected 3D models onto 2D query images; their 3D models included inferred object faces which are not actually visible in any model image. In order to make a more fair evaluation of our method, which is based on affine transformations, we modified the ground-truth segmentations in the query images by removing hidden faces of the object (e.g. the top of tins). Furthermore, we observed that our method enables the use of a smaller overlap threshold (0.4, instead of 0.5), giving more true positive matches without any increase in the number of false positives.

Initially, a hierarchical k-means tree is used to establish keypoint matches. For Lowe's test, each query keypoint p can only match its single nearest neighbor in the whole model set. For SCRAMSAC, each query keypoint p, after running through the tree, can match keypoints in different model images, up to one match per image. Then, for each query keypoint p, only the top N keypoint matches are retained (based on descriptor distance). Finally, keypoint matches without sufficient local support are removed [3].

In the final stage, RANSAC is used for object pose estimation. For Lowe's test and SCRAMSAC, each putative affine transformation is generated by randomly selecting *three* keypoint matches that occur with the same model image. For our method, each putative affine transformation is generated by randomly selecting *one* keygraph match.

B. Results

Figure 6 shows two sample object detection results of our method.

Figure 5-a shows the AP achieved by our method versus the number T of Delaunay triangulations used to sample keypoint triples. Each curve corresponds to a model dataset, CMU10 or CMU10+PASCAL; in both cases, in the initial keypoint matching stage, $L = 1000$ model keypoints were compared to each query keypoint.

As shown in Figure 5-a, employing a large number of Delaunay triangulations provided a significant gain in performance: the AP increased in almost 0.50 in comparison to using $T = 1$. For the CMU10 dataset, the AP plateaued at $T = 50$. Interestingly, when employing the more complex CMU10+PASCAL model dataset, the AP continued to increase for $T > 50$, going from 0.49 to 0.58 when using $T = 200$. This suggests that our sampling strategy was able to generate a diverse set of query keygraphs, as employing more triangulations partially compensated the decrease in accuracy due to the use of a larger number of model images.

Figure 5-b shows that the number of keygraphs grows sub-linearly with the number of triangulations.

Figure 5-c shows the average number of keygraph matches between a query image and all the model images versus the threshold of the filtering stage in Equation 1. We changed the interval of accepted ratios, which is $[0.5, 2]$ in Equation 1. When a large interval of $[1/60, 60]$ is used, all candidates are accepted, thus filtering is performed only by the cyclic ordering requirement and the orientation based filtering stage (which uses a threshold of $60°$). Increasing the interval from $[0.5, 2]$ to $[1/60, 60]$ caused the number of established keygraph matches to increase by a factor of 100.

Figure 5-d is similar to Figure 5-c, but considers the orientation based filtering stage. Increasing the threshold from $60°$ (Equation 2) to $180°$ caused the number of established keygraph matches to increase by a factor of 10.

Using both Equations 1 and 2 for filtering, the average number of keygraph matches established per query image

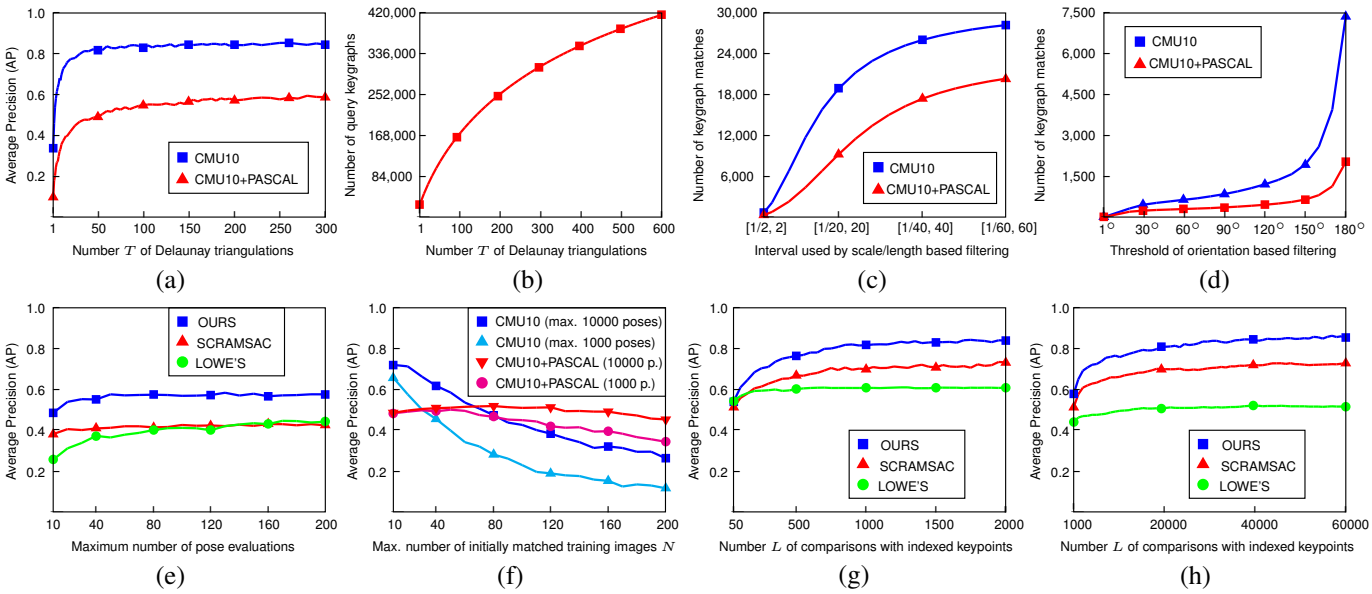

Figure 5. **First row**: evaluation of the effect of parameters in our method. **(a)** AP versus number T of Delaunay triangulations for the CMU10 or CMU10+PASCAL model datasets. In the latter dataset, PASCAL images were added to the model set to assess scalability and the effect of added negative samples, which is expected to reduce the AP as more mismatches may occur. **(b)** Average number of keygraphs sampled in a query image versus T. **(c)** and **(d)** Average number of keygraph matches between a query image and the model images versus the threshold used by the filtering stage based on (c) scale/length or (d) orientation. **Second row**: comparisons against other methods. **(e)** AP versus the maximum allowed number of pose evaluations of all model images for each query. **(f)** AP of SCRAMSAC versus the maximum allowed number N of model images matched by a query keypoint. **(g)** and **(h)** AP versus the number L of keypoint comparisons between query and indexed model keypoints, for (g) CMU10 or (h) CMU10+PASCAL.

was 600, for the CMU10 model images ($T = 50$), and 400, for the CMU10+PASCAL model images ($T = 200$).

Figure 5-e shows the AP versus the maximum allowed number of pose evaluations (affine transformations), employing the CMU10+PASCAL model dataset. Our method achieved a better performance than SCRAMSAC and Lowe's test, even for very few pose evaluations.

Figure 5-f shows the AP achived by SCRAMSAC versus the maximum allowed number of matches N between a query keypoint and the model images. We evaluated two cases: using a maximum of 1000 or 10000 pose evaluations. Using the CMU10 model dataset, the best results were achieved around $N = 10$, while for the more complex CMU10+PASCAL model dataset, the best results were achieved around $N = 50$; in both cases, using additional pose evaluations improved the performance. An advantage of the proposed keygraphs method is not having a parameter such as N, as our method does not limit the number of model images initially matched by a query keypoint.

Figure 5-g shows the AP versus the number L of indexed model keypoints which are compared to a query keypoint, for the CMU10 model dataset. Good results were achieved by employing $L = 1000$ (0.4% of the number of stored keypoints), for both SCRAMSAC and our method. For Lowe's test, setting L above 400 did not provide significant improvement.

Figure 5-h is similar to Figure 5-g, but considers the CMU10+PASCAL dataset. For our method and SCRAMSAC, employing $L = 30000$ allows achieving the same AP as when using $L = 1000$ with the CMU10 dataset. This is interesting, as the number of model descriptors, from CMU10 to CMU10+PASCAL, also increased by a factor of 30. On the other hand, Lowe's test was not able to similarly compensate the performance loss caused by the use of a larger number of model images.

In summary, for the CMU10 model dataset, the AP of our method, SCRAMSAC and Lowe's test was, respectively, 0.82, 0.70 and 0.61. For the CMU10+PASCAL model dataset the AP was: for $L = 1000$: 0.58, 0.51 and 0.44; for $L = 2000$: 0.65, 0.58 and 0.46; and for $L = 30000$: 0.84, 0.70 and 0.51.

The AP achieved by our method using the CMU10 model dataset, 0.82, was superior to the AP reported by Hsiao *et al.* [4], 0.78, even though Hsiao *et al.* employed 3D object models which allows to use matches to different model images and to use 3D-2D pose instantiation.

We evaluated the computation time demanded by our method (using a C implementation). For both the CMU10 and CMU10+PASCAL model dataset, when using $L = 1000$, $T = 50$, the time per query image was 1.4 ± 0.3 sec., with 55% of the time being spent on keypoint matching, 3% on keygraph sampling and 39% on keygraph matching. Using the CMU10+PASCAL model images and $T = 200$, the time per query image was, for $L = 1000$, 2.5 ± 0.7 sec.,

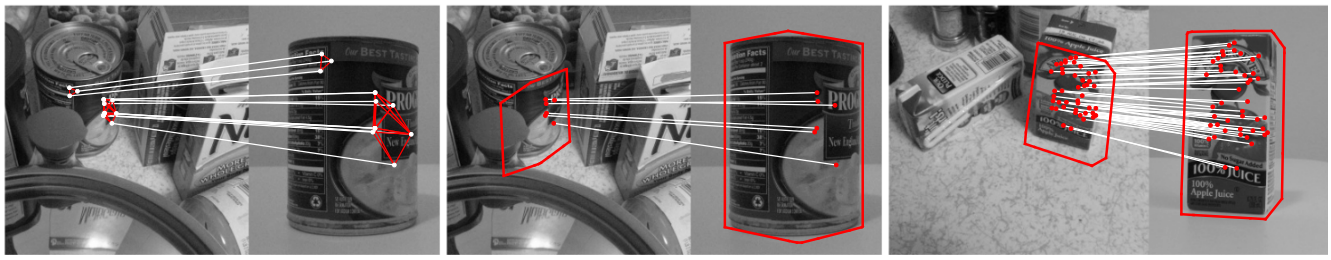

Figure 6. Keygraph and keypoint matches. The region overlaps achieved by the affine transformations were 0.61 and 0.73, respectively.

and for $L = 2000$, 6.8 ± 1.7 sec.; for $L = 1000$, 33% of the time was spent on keypoint matching, 9% on keygraph sampling and 57% on keygraph matching; for $L = 2000$ those values were 26%, 3% and 70%, respectively.

V. Conclusion

Keygraphs constitute an intermediate level feature, above the keypoints. Initially, keypoint matches are established by using an indexing tree. Then, keypoint matches are transformed into keygraph matches, which correctly removes the large majority of the incorrect initial keypoint matches.

This paper introduced structural properties used to filter out candidate keygraph matches. We considered an heterogeneous information set, concerning keypoints scale, orientation, image position and cyclic ordering, which allows to correctly filter out 99% of the candidate keygraph matches.

We also introduced an efficient algorithm to sample query keypoint triples, based on using complementary Delaunay triangulations. Each triangulation is done in a way that vertices are not too close to each other so the triangles generate more robust pose hypotheses.

At application, the processing time of our method depends almost exclusively on parameters that are pre-set by the user (triangulations T and keypoint comparisons L). We showed that a 30-fold increase in the number of model keypoints does not significantly deteriorate the performance of our method.

Acknowledgment

The authors are grateful to FAPESP grant # 2011/50761-2, # 14/50769-1, CNPq grant 502472/2014-6, CAPES, NAP eScience - PRP - USP.

References

[1] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] T. Sattler, B. Leibe, and L. Kobbelt, "SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter," in *Proceedings of the IEEE Conference on Computer Vision*, 2009, pp. 2090–2097.

[4] E. Hsiao, A. Collet, and M. Hebert, "Making specific features less discriminative to improve point-based 3D object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2653–2660.

[5] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, F. Wu, and Y. Rui, "Efficient 2D-to-3D correspondence filtering for scalable 3D object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 899–906.

[7] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, and F. Wu, "3D visual phrases for landmark recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3594–3601.

[8] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009, pp. 331–340.

[9] M. Hashimoto and R. M. Cesar Jr, "Object detection by keygraph classification," in *Graph-Based Representations in Pattern Recognition*. Springer, 2009, pp. 223–232.

[10] H. Morimitsu, R. B. Pimentel, M. Hashimoto, R. M. Cesar, and R. Hirata, "Wi-fi and keygraphs for localization with cell phones," in *Proc of ICCV Workshops*. IEEE, 2011, pp. 92–99.

[11] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *International Journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[13] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.