Pattern Recognition 60 (2016) 499–514

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Object recognition using local invariant features for robotic applications: A survey

Patricio Loncomilla, Javier Ruiz-del-Solar*, Luz Martínez

Advanced Mining Technology Center & Dept. of Electrical Engineering, Universidad de Chile, Chile

ARTICLE INFO

Article history:
Received 2 February 2016
Received in revised form
15 March 2016
Accepted 11 May 2016
Available online 24 May 2016

Keywords:
Local invariant features
Object recognition
Local descriptors
Local interest points

ABSTRACT

The main goal of this survey is to present a complete analysis of object recognition methods based on local invariant features from a robotics perspective; a summary which can be used by developers of robot vision applications in the selection and development of object recognition systems. The survey includes a brief description of the main approaches reported in the literature, with more specific analyses of local interest point computation methods, local descriptor computation and matching methods, and geometric verification methods. Different methods are analyzed by considering the main requirements of robotics applications, such as real-time operation with limited on-board computational resources, and constrained observational conditions derived from the robot geometry (e.g. limited camera resolution). In addition, various object recognition systems are evaluated in a service-robot domestic environment, where the final task to be performed by a service robot is the manipulation of objects. It can be concluded from the results reported that (i) the most suitable keypoint detectors are ORB, BRISK, Fast Hessian, and DoG, (ii) the most suitable descriptors are ORB, BRISK, SIFT, and SURF, (iii) the final performance of object recognition systems using local invariant features under real-world conditions depends strongly on the geometric verification methods being used, and (iv) the best performing object recognition systems are built using ORB–ORB and DoG–SIFT keypoint–descriptor combinations. ORB–ORB based systems are faster, while DoG–SIFT are more robust to real-world conditions.

1. Introduction

The recognition of objects under uncontrolled, real-world conditions is of paramount importance in robotics. Object recognition is an essential ability for building object-based representations of the environment, and for the manipulation of objects. In this work, object recognition refers to the recognition of a specific object instance (e.g. my cup), instead of a generic object class/category (e.g. a cup), which is usually called object categorization or generic object recognition. Both object recognition and object categorization are important abilities in robotics, and they are used for solving different tasks. This survey is focused on object recognition, and then the analysis of object categorization techniques (Bag of Visual Words [65], VLAD [21], FLAIR [59], cascaded ensembles of randomized decision trees [4], unsupervised segmentation of unknown objects [5]) is beyond its scope.

In recent years, several approaches to object recognition have been developed. They are usually based on global and/or local descriptions of the objects. Global description based methods model the appearance of an object as a whole, while local description based methods represent objects as a set of local interest points (keypoints), each of them represented by a local invariant feature[1] (or descriptor). Methods based on local features have advantages, such as not needing object segmentation, robustness against occlusions and against changes in the viewpoint (rotation and scale change), and having a near real-time recognition frame rate.

Object recognition methods based on the use of local invariant features have been developed mostly within the computer vision community, and then transferred to the robotics community. However, robot vision applications have different requirements than standard computer vision applications, such as the requirement of real-time operation with limited on-board computational resources, and the constrained observational conditions derived from the robot geometry, limited camera resolution, and sensor/object relative pose. In addition, in many cases the developers of robot vision applications adapt computer vision modules (e.g. the ones available in OpenCV [73]) to their robotics applications,

* Corresponding author.
E-mail addresses: ploncomi@ing.uchile.cl (P. Loncomilla), jruizd@ing.uchile.cl (J. Ruiz-del-Solar), luz.martinez@amtc.cl (L. Martínez).

[1] In this work, the focus is on appearance-based keypoints and features; 3D feature descriptors such as feature histograms obtained from range images are not considered.

http://dx.doi.org/10.1016/j.patcog.2016.05.021
0031-3203/© 2016 Elsevier Ltd. All rights reserved.

without analyzing the specific characteristics of the methods and the applications' requirements carefully. Only general criteria, such as "SURF is faster than SIFT", are applied.

In this context, the main motivation of this survey is to present a complete analysis of object recognition methods based on local invariant features from a robotics perspective, which can be used by developers of robot vision applications in the selection and development of object recognition systems. The survey analyzes the main functionalities of popular methods (local interest point computation, local descriptors computation and matching, geometric verification), and presents evaluations in terms of accuracy, robustness and efficiency.

Previous studies have analyzed the performance of object recognition approaches based on local features without considering the full requirements of robotics applications. For instance, in [68] the authors focused on the analysis of the precision of the methods regarding viewpoint angle, scale and affine transformations, but without considering the main robot vision requirements, such as real-time operation. In [50], several interest point detectors are compared, and their runtime and accuracy are evaluated for several image resolutions. However, real world problems, such as changes in illumination, background and partial occlusions are not analyzed. In [36], six object recognition algorithms based on local descriptors are compared in an object recognition task (recognizing objects on a table). Real-world conditions are included in that comparison. The results obtained are included in this survey, and the experiments they performed are extended.

This survey includes a brief description of the main approaches described in the literature, with specific analyses of local interest point computation methods, local descriptor computation and matching methods, and geometric verification methods. In addition, comparisons of the applicability of the methods in robotic applications, based on their accuracy, robustness, and efficiency, are presented.

The use of RGB-D sensors is very popular in the robotics community, and it could be wrongly assumed that the use of local invariant visual features is less relevant than the use of 3D range derived features. This assumption is really incorrect, because (i) the use of local invariant visual features is complementary to the use of 3D range features, (ii) standard RGB-D sensors do not work properly in outdoors and/or when observing black surfaces, restricting their applicability, and (iii) 3D range features require a much higher resolution than visual features to recognize objects, therefore their use impose constraints in robotics applications, e.g. objects can be recognized only at short distances (see the detailed analysis in [36]).

It is also important to explain why this survey does not include object methods based on deep learning. Although object recognition based on the use of the deep learning paradigm is a hot topic in the computer vision community, and its use in robotics applications will increase in the near future, still most of these methods are not able to fulfill the main requirements of robotics applications (real-time operation with limited on-board computational resources). Certainly, the use of both object recognition paradigms (local invariant features and deep-learning) will complement each other in the near future.

This paper is organized as follows: in Section 2, the paradigm of object recognition through the use of local features is presented. In Section 3, several interest point detection algorithms including corner-based and blob-based variants are described. In Section 4, algorithms for computing local descriptors and the standard procedure for matching descriptors are explained. In Section 5, algorithms for finding geometric verification of the matched features are described. In Section 6, a comparison of several object recognition systems in a real robot application is presented. Finally, in Section 7 some conclusions are drawn.

This survey intends to be a guide for developers of object recognition systems for robotics applications. The reader interested in having a practical guide for the use/application of the different algorithms/methods, and not just a description of them, is referred to (sub) Sections 2, 3.4, 4.2, 4.3, 5.3 and 6.

2. Object recognition using local invariant features

A local feature is "an image pattern which differs from its immediate neighborhood" [68]. A local interest point, also called a keypoint, defines the position of a local feature, and a descriptor describes/represents its image pattern. Therefore, the interest points are first searched for in the image under analysis, and then the regions around the interest points are described by the descriptors.

In general terms, object recognition based on local invariant features works according to the following principle: (i) keypoints are extracted independently from both a test image and a reference image (model), and characterized using invariant descriptors, and (ii) the invariant descriptors (features) are matched with each other. Afterwards, (iii) geometric verifications of the matched features are carried out using different procedures. For instance, whether or not the matched features satisfy a similarity or an affine transformation is tested.

The object recognition pipeline includes the following stages (see Fig. 1):

- Object segmentation (optional): In case a depth image I_D is available, objects that are on a planar surface, such as a table or the floor, can be isolated/segmented, and the object recognition method can be applied to only the segmented area.
- Local Interest Point Computation: Interest points (keypoints) are computed from the image I_{RGB} under analysis.
- Descriptors Computation: Local image descriptors are computed around each keypoint. In some methods more than one descriptor can be computed for each keypoint, depending on the local gradients' characteristics.
- Matching: Local descriptors belonging to the image under analysis I_{RGB} and to reference images (training descriptors) are

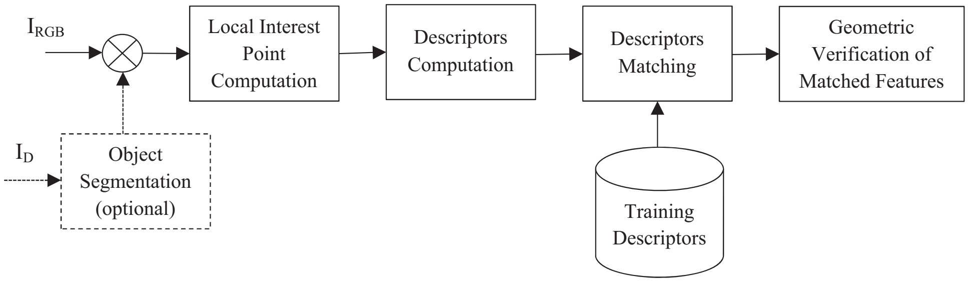
I_RGB
Local Interest Point Computation
Descriptors Computation
Descriptors Matching
Geometric Verification of Matched Features
I_D
Object Segmentation (optional)
Training Descriptors

Fig. 1. Pipeline used for Recognizing Objects by using Local Invariant Features. The object can be segmented by using the depth image. Then, interest points and descriptors are computed and compared against those in a database. Extra verifications can be performed for rejecting incorrect detections.

matched. Matches between descriptors are found by searching descriptors that belong to different images and are the most similar.

- Geometric Verification of Matched Features: The matches corresponding to an image pair are analyzed by searching sets of keypoints from one image that are then mapped onto the other image via a common transformation. In this process, more than one transformation, some of them wrong, can be found; so, several tests are carried out in order to reject the wrong transformations. The tests enable rejecting transformations with a low probability of representing a real detection, transformations that are numerically unstable, or those that generate a great amount of distortion when mapping images.

It is important to mention that when the local invariant features are used for object categorization, instead of matching the descriptors, visual words are computed from them. This object recognition paradigm is called *Bag of Visual Words* (BoVW), but it is beyond the scope of this survey which does not cover applications of local invariant features for object categorization.

Several basic algorithms can be used as building blocks for implementing the described object recognition pipeline. A summary of these basic algorithms is given in Table 1.

It must be noted that some of the most popular object recognition methods, which include most of the described stages, are known by the name of the descriptor computation algorithm that they use. This is the case for the SIFT and SURF methods.

Table 2 shows examples of papers on object recognition for robotics applications based on local invariant features. In each case the interest points, descriptors, and geometric verification algorithms used are identified. In addition, whether or not the implemented recognition systems use object segmentation based on RGB-D cameras is included.

3. Local interest point computation

Interest points are normally discontinuities in the image space or in the image scale space. There are two main families of interest point detectors: corner detectors, which are obtained by computing the maximal values of a so-called *cornerness* function, and blob detectors, which are obtained by detecting structures that are maximal on some scales of the image scale-space.

The detection of interest points must be precise and repeatable [18], and whenever possible, scale invariant (the same interest points are obtained independently of the image scale). Different computational implementations of the same interest point detector and descriptor can generate very different results [17]. This fact weakens the comparison of the accuracy and runtime of the methods. For a detailed description of interest point detectors, the reader can refer to [68] and [18].

3.1. Corner detectors

3.1.1. Moravec detector

The use of interest points of images started with the work of Moravec [42]. In that work, corners in images are detected by comparing image patches (centered in x,y) against nearby ones (centered in $x+\Delta x$, $y+\Delta y$). Regions corresponding to uniform areas are similar to all nearby patches; regions corresponding to edges are similar only to regions along the direction of the edge; and regions corresponding to corners are different from nearby patches in all of the directions. The patch has a circular shape, and four different directions Δx,Δy are selected for comparing patches. Differences between patches are computed by using a sum of squared differences (SSD) between the patches, which represents the energy of the difference between them. The Moravec energy function associated with a displacement (Δx, Δy) is computed as follows.

Table 1
Summary of Basic Algorithms used as Building Blocks in Object Recognition Methods.

Algorithm	Functionality (Pipeline stage)	Paper
Harris keypoints	Local Interest Point Computation	Harris and Stephens, 1988 [20]
SUSAN keypoints	Local Interest Point Computation	Smith and Brady, 1997 [66]
Harris–Laplace keypoints	Local Interest Point Computation	Lindeberg, 1998 [29]
MSER regions	Local Interest Point Computation	Matas et al. 2002 [37]
Harris–Affine keypoints, Hessian–Affine keypoints	Local Interest Point Computation	Mikolajczyk and Schmid, 2004 [39]
DoG keypoints, SIFT descriptor, probability test	Local Interest Point Computation & Descriptors computation & Transformation Computation & Extra Verification	Lowe, 2004 [32]
FAST keypoints	Local Interest Point Computation	Rosten and Drummond, 2006 [52]
FAST+CSLBP descriptors	Local Interest Point Computation & Descriptors computation	Lu et al. [33]
Fast Hessian keypoints, SURF descriptors	Local Interest Point Computation & Descriptors computation	Bay et al., 2008 [8]
Rank order LoG	Local Interest Point Computation	Miao et al. [38]
PCA–SIFT descriptor	Descriptors computation	Ke and Sukthankar, 2004 [24]
ASIFT descriptor	Descriptors computation	Morel and Guoshen, 2009 [43]
BRIEF descriptor	Descriptors computation	Calonder et al., 2010 [10]
ORB descriptor	Descriptors computation	Rublee et al., 2011 [55]
BRISK descriptor	Descriptors computation	Leutenegger et al., [27]
FREAK descriptor	Descriptors computation	Alahi et al. [2]
LIOP descriptor	Descriptors computation	Wang et al. [70]
MROGH descriptor	Descriptors computation	Fan et al. [14]
MRRID descriptor	Descriptors computation	Fan et al. [14]
MDGMH–SURF descriptor	Descriptors computation	Kang et al. [23]
Hough Transform for Transformation Computation Verification	Transformation Computation	Ballard, 1981 [7]
RANSAC algorithm	Transformation Computation	Fishler and Bolles, 1981 [15]
HEASK	Transformation Computation & Extra verifications	Yan et al. [71]
Wide baseline matching using local descriptors, semi-local constraints	Transformation Computation & Extra Verifications	Schmid and Mohr, 1997 [60]
Clique descriptor	Descriptors computation, Transformation Computation	Shin et al. [64]
L&R hypothesis verification stages: linear correlation test, fast probability, affine distortion test, pixel correlation test	Extra Verifications	Ruiz-del-Solar and Loncomilla, 2009 [57]

Table 2
Selected papers on object recognition for robotics applications based on local invariant features.

Paper	Interest points	Descriptors	Geometric verification	Segmentation with RGB-D Camera	Application
Lowe, 2004 [32]	DoG	SIFT	Hough, probability	No	Object recognition
Kragic, 2005 [26]	DoG	SIFT	RANSAC, M-estimators	Yes	Object recognition
Pillai, 2015 [48]	Dense + multi-view object proposals	SIFT + PCA	BOVW + VLAD, evidency accumulation over frames	Segmentation using modified ORB–SLAM	Object recognition
Ramisa, 2009 [50]	Fast Hessian, DoG, Harris–Affine, Harris–Laplace, Hessian–Affine, Hessian–Laplace, MSER	SIFT	RANSAC, Iterative reweighted least squares (IRLS), Heuristics filtering	No	Object recognition
Zickler, 2006 [72]	DoG	PCA–SIFT	Centroid, Clustering and Temporal Voting Space	No	Identify/localize objects
Alhwarin, 2008 [3]	DoG	SIFT	RANSAC	No	Object Recognition
Effendi, 2008 [13]	DoG	SIFT	Hough	Yes	Object Recognition
Azad, 2009 [6]	Harris	SIFT	Hough, RANSAC, and Least squares homography estimation	No	Object recognition and localization
Collet, 2009 [11]	DoG	SIFT	RANSAC and mean shift clustering, RANSAC and Levenberg–Marquardt	Yes	Object recognition and pose estimation
Danieletto, 2009 [12]	DoG	SIFT	Number of matches	No	Recognition smart objects
Kouskouridas, 2009 [25]	DoG	SIFT	Distances from keypoints to the center of mass of the features	Yes	Category Recognition and Pose estimation
Morel, 2009 [43]	DoG	ASIFT	None	No	Object Recognition
Ruiz-del-Solar, 2009 [57]	DoG	SIFT	Hough, L&R	No	Object recognition
Srinivasa, 2010 [67]	DoG	SIFT	Levenberg–Marquardt, clustering, and robust matching RANSAC and mean shift clustering	Yes	Object Recognition and Pose Estimation
Martinez, 2014 [36]	DoG	SIFT	RANSAC Levenberg–Marquardt	No	Object Recognition and Pose Estimation
Jia, 2011 [22]	DoG	SIFT	Feature codebook filtering	No	Object detection, Path Planning
Ramathan, 2011 [49]	DoG	SIFT	Bag-of-Words	No	Object Recognition
Saenko, 2011[58]	DoG	SIFT, HoG	Naive Bayes Nearest Neighbor	Yes	Object /Category Recognition
Seib, 2011 [62]	Fast Hessian	SURF	RANSAC	Yes	Object Recognition
Madry, 2012 [34]	DoG	SIFT, opponent SIFT, HoG	Bag-of-Words	Yes	Category Recognition
Piccinini, 2012 [47]	DoG	SIFT	RANSAC RANSAC clustered, Voting scheme Mean shift	Yes	Object Detection
Rigual, 2012 [51]	DoG	SIFT	RANSAC and mean shift clustering	Yes	Object Recognition
Han, 2013 [19]	Landmarks in contours with Radians	SIFT	EDT, AT, RANSAC with Procrustean distance	No	Object Recognition
Lopez, 2013 [31]	Hessian–Laplace	SURF	Voting	Yes	Object Detection
Patil, 2013 [46]	DoG	SIFT	Mean shift clustering, voting for principal points	No	Object Recognition and Localization
Seib, 2013 [61]	Fast Hessian	SURF	Hough	Yes	Object Recognition
Li ,2014 [28]	DoG	SIFT	Bag of features max pooling linear SVM	Yes	Deformable Object Recognition for Manipulation
Martinez, 2014 [36]	DoG, FastHessian	SIFT, SURF	RANSAC L&R obj_rec_surf	Yes	Object Recognition
Nie, 2015 [45]	DoG	SIFT	Fuzzy control loop recognition	No	Object recognition for manipulation
Manfredi, 2015 [35]	DoG	ASIFT	Fast probability	No	Object recognition

$$E_{Moravec}(x, y, \Delta x, \Delta y) = \sum_{(u,v)\text{ around }(x,y)} w_{circle}(u, v)\big(I(u + \Delta x, v + \Delta y) - I(u, v)\big)^2 \tag{1}$$

with $w_{circle}(u,v)$ being a function that represents the window used for selecting the patch. The smallest energy between the central patch and its neighbors is chosen as the cornerness measure.

The cornerness function is computed by considering four possible directions, in the following way:

$$C_{Moravec}(x, y) = \min\big(E_{moravec}(x, y, 1, 0), E_{moravec}(x, y, 1, 1), E_{moravec}(x, y, 0, 1), E_{moravec}(x, y, -1, 1)\big) \tag{2}$$

Local maxima of the cornerness function indicate corner points that are used for representing the image by using a set of keypoints.

3.1.2. Harris detector and Shi and Tomasi detector

Interest points computed by the Moravec approach have problems with position repeatability because any corner inside the circular patch generates a similar response independent of its position inside the patch. Harris and Stephens [20] generated an improved corner detector by using a Gaussian window instead of a circular one for obtaining better position repeatability, and by applying a first order Taylor expansion to the SSD computed in Moravec's work. The image is first blurred using a derivation scale for enabling the computation of derivatives by using finite differences (see a complete derivation in [20]):

$$w_{gaussian}(u, v, \sigma) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3}$$

$$L(u, v, \sigma) = w_{gaussian}(u, v, \sigma_D)*I(u, v) \tag{4}$$

$$E_{Harris}(x, y, \Delta x, \Delta y) = \begin{pmatrix}\Delta x & \Delta y\end{pmatrix}\mu(x, y, \sigma_I, \sigma_D)\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} \tag{5}$$

$$\mu(x, y, \sigma_I, \sigma_D) = \sum_{(u,v)} w_{gaussian}(u, v, \sigma_I) \times \begin{pmatrix} \frac{\partial L(x, y, \sigma_D)}{\partial x}\frac{\partial L(x, y, \sigma_D)}{\partial x} & \frac{\partial L(x, y, \sigma_D)}{\partial x}\frac{\partial L(x, y, \sigma_D)}{\partial y} \\ \frac{\partial L(x, y, \sigma_D)}{\partial x}\frac{\partial L(x, y, \sigma_D)}{\partial y} & \frac{\partial L(x, y, \sigma_D)}{\partial y}\frac{\partial L(x, y, \sigma_D)}{\partial y} \end{pmatrix} \tag{6}$$

with σ_D representing the derivation scale, and σ_I representing the integration scale, that is related to the size of the Gaussian window. Usually $\sigma_I = 1.6\ \sigma_D$ is selected.

When the patch around (x, y) corresponds to a corner, the Harris energy function must be positive for all displacements $(\Delta x, \Delta y)$. Then, the two eigenvalue images $\lambda_1(x, y)$ and $\lambda_2(x, y)$ from the second moment matrix $\mu(x, y)$ must be positive around (x, y). Different corner detectors have been proposed by creating different cornerness functions.

- Harris cornerness function [20]:

$$C_{Harris}(x, y) = \det\big(\mu(x, y)\big) - k\ trace^2\big(\mu(x, y)\big) \tag{7}$$

- Shi–Tomasi cornerness [63]:

$$C_{Shi-Tomasi}(x, y) = \min\big(\lambda_1(x, y), \lambda_2(x, y)\big) \tag{8}$$

The keypoints computed by these methods correspond to the maxima of these cornerness functions.

3.1.3. Harris–Laplace detector

The Harris and Shi–Tomasi cornerness functions are rotation and translation invariant, but they depend on the size of the Gaussian window σ_I, i.e., they detect corners in a scale σ_I. When an image is rescaled, the scale of the corners changes, and then the cornerness functions will not be able to detect the corners in their new scale. The Harris corner detector can be modified for obtaining scale invariance, i.e., for detecting corners across all of the scales. When an image $I(x,y)$ has to be processed, a scale space $L(x, y,\sigma)$ of that image is constructed [29]:

$$L(x, y, \sigma) = w_{gaussian}(x, y, \sigma)*I(x, y) \tag{9}$$

The scale space can be constructed in basically two ways: the first is blurring the image several times before resampling it at half of the resolution, and the second way is by constructing a pyramid by blurring and resampling the image immediately at fraction of the original size, usually at 2/3 of the original resolution. Then, the Harris interest point detector is applied to all of the images in the scale space, generating corners that exist over several scales. An automatic scale selection procedure [29] is applied to every corner by selecting the scale with a maximal Laplacian of Gaussian response. The resulting (x,y,σ) keypoints are named Harris–Laplace points. The Laplacian of Gaussian response can be approximated by a Difference of Gaussian, which is computed by subtracting consecutive pairs of levels of the scale-space as shown in Fig. 2.

3.1.4. Harris–Affine detector

The Harris–Laplace detector can be adapted for reaching invariance against affine transformations. An ellipsoidal region is selected instead of the circular one, by imposing the condition that the two eigenvalues from the moment matrix μ must be the same when projecting the ellipsoidal region into a circular one. The equation for the ellipsoidal region is based on the affine second moment matrix, which is adapted for working with non-symmetric Gaussian kernels that represent ellipsoids.

$$w_{gaussian}(x, y, \Sigma) = \frac{1}{2\pi\sqrt{\det(\Sigma)}}\exp\left(-\frac{1}{2}\begin{pmatrix}x & y\end{pmatrix}\Sigma^{-1}\begin{pmatrix}x \\ y\end{pmatrix}\right) \tag{10}$$

$$\mu(x, y, \Sigma_I, \Sigma_D) = \sum_{(u,v)} w_{gaussian}(u, v, \Sigma_I) \times \begin{pmatrix} \frac{\partial L(x, y, \Sigma_D)}{\partial x}\frac{\partial L(x, y, \Sigma_D)}{\partial x} & \frac{\partial L(x, y, \Sigma_D)}{\partial x}\frac{\partial L(x, y, \Sigma_D)}{\partial y} \\ \frac{\partial L(x, y, \Sigma_D)}{\partial x}\frac{\partial L(x, y, \Sigma_D)}{\partial y} & \frac{\partial L(x, y, \Sigma_D)}{\partial y}\frac{\partial L(x, y, \Sigma_D)}{\partial y} \end{pmatrix} \tag{11}$$

A transformation that maps the ellipsoidal patch to a circular one can be obtained by using the affine second moment matrix.

$$M = \mu(x, y, \Sigma_I, \Sigma_D) \tag{12}$$

$$\begin{pmatrix}x_{circle} \\ y_{circle}\end{pmatrix} = M^{-1/2}\begin{pmatrix}x_{ellipse} \\ y_{ellipse}\end{pmatrix} \tag{13}$$

$$\Sigma_i = k\Sigma_D \tag{14}$$

Initial keypoint hypotheses are obtained by using the Harris–Laplace detector. As the ellipsoid corresponding to each point is not initially known, a first approximation is obtained by using Eqs.

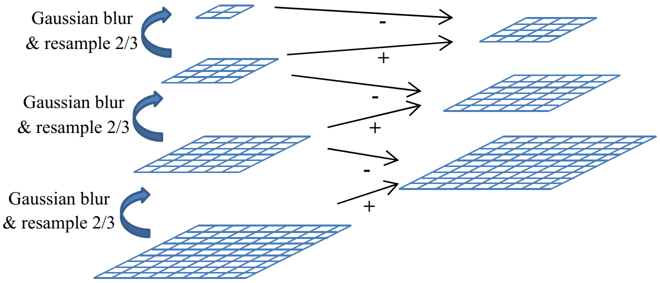

Fig. 2. Computation of difference of Gaussians Space using a Pyramidal Structure.

(12)–(14), which are applied using the initial circular window corresponding to the Harris–Laplace keypoint scale. The computation of Eqs. (12)–(14) is iterated until a final ellipsoidal region is obtained. This process, named affine adaptation [39], is very slow, and therefore this descriptor cannot be used for robotic applications. Furthermore, the problem generated by changes of viewpoints can be addressed by using several views from the object.

3.1.5. SUSAN and FAST detectors

SUSAN (Smallest Univalue Segment Assimilating Nucleus) [66] is based on searching for a small nucleus inside a sliding circular mask. If the nucleus is at (x_0,y_0), a score $N(M)$ indicating the number of similar pixels inside the circular mask M can be computed as:

$$N(M) = \sum_{(x,y)\in M} e^{-\left(\frac{I(x,y)-I(x_0,y_0)}{t}\right)^6} \tag{15}$$

with t being a soft luminance threshold used for segmenting the nucleus. The score $N(M)$ increases with the number of pixels in the neighborhood that are similar to the nucleus center. Points with a minimal $N(M)$ value can be selected as corners, as their mask contains many dissimilar values, that can be interpreted as a corner.

The FAST (Features from Accelerated Segment Test) corner detector [52–53] uses a circle of 16 pixels around the nucleus point. If the number of pixels in the circle that are darker than the center is over a given threshold (usually 9), the point is selected as a bright corner (for this reason the detector is sometimes called FAST 9-16). Also, a dark corner can be selected by ensuring that the center is darker than the pixels on the circle.

SUSAN is fast to compute, and FAST is even faster, as it requires only a small number of luminance comparisons. However, these detectors are not scale invariant.

3.1.6. BRISK and ORB detectors

The BRISK (Binary Robust Independent Scalable Keypoints) detector [27] is an adaptation of the FAST detector that searches for keypoints in the scale space of an image. The scale-space pyramid layers consist of n octaves c_i and n intra-octaves d_i, for $i=\{0,1,\ldots,n-1\}$, with typically $n=4$. The octaves are formed by half-sampling the original image (corresponding to c_0) progressively. Each intra-octave d_i is located in-between layers c_i and c_{i+1}. The first intra-octave d_0 is obtained by down sampling the original image c_0 by a factor of 1.5, while the rest of the intra-octave layers are derived by successive half sampling. The FAST detector is applied to all of the images in the octaves and intra-octaves, and then interest points are found by performing a non-maxima suppression both in the same layer (8 neighbors) and with respect to the layers above and below. The position of the interest point is refined by fitting a 1D parabola along the axis scale yielding the final score estimate and the scale estimate at its maximum. As a final step, the image coordinates between the patches in the layers next to the determined scale are re-interpolated. An orientation is assigned by computing a dominant gradient direction using several pairs of points.

ORB (Orientated FAST and Rotated BRIEF) [55] is a methodology for obtaining both keypoints and binary descriptors. The keypoint detector is an oriented FAST detector, in which the orientation of the keypoint is obtained by computing the direction between the position of the detected feature and the intensity centroid around the keypoint; i.e., it assumes that the intensity centroid is not at the same position as the keypoint.

3.2. Blob based detectors

Corner detectors give highly repeatable and stable interest points; however, for some applications, the number of interest points is more important than their quality. Blob based detectors generate a higher number of interest points by sacrificing repeatability and stability.

3.2.1. DOG detector

Blob based detectors are deduced from an analysis of the scale space of the image, by searching for structures that disappear over a certain level of blur. A scale space $L(x,y,\sigma)$ of the image is created, and a set of difference images $D(x,y,\sigma)$ is computed by pixel-wise subtraction of consecutive levels of the scale space.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{16}$$

Points that are 3D maxima in the space $D(x,y,\sigma)$ are named Difference of Gaussian (DoG) points [32], and correspond to structures that disappear from the scale space at that level of blurring.

3.2.2. Determinant of Hessian detector

Blob-like structures can also be detected by applying a Hessian operator on the image and then detecting local maxima of the determinant of the Hessian. The detected keypoints are named Determinant of Hessian (DoH) points [9], and they represent maxima of local curvature in the image.

$$H_{xx}(x, y) = \frac{d^2}{dx^2}I(x, y) \tag{17}$$

$$H_{yy}(x, y) = \frac{d^2}{dy^2}I(x, y) \tag{18}$$

$$H_{xy} = \frac{d^2}{dxdy}I(x, y) \tag{19}$$

$$\det(H) = H_{xx}H_{yy} - H_{xy}^2 \tag{20}$$

These keypoints are not scale invariant because they are affected by the resolution of the image.

3.2.3. Hessian–Laplace detector

The Determinant of Hessian detector can be improved by applying it on the scale space of the image. A scale space $L(x, y, \sigma)$ is constructed, and a Determinant of Hessian scale space is computed, by applying a determinant of Hessian operator for each image on the $L(x, y, \sigma)$ scale space. Then, local maxima on (x, y) are detected in each of the images. As a blob-like structure can generate 2D maxima over several scales, a Laplacian of Gaussian scale space $\nabla_\sigma L(x, y, \sigma)$ is constructed and local maxima in the σ axis are used for automatic scale selection.

3.2.4. Fast Hessian detector

A procedure named Fast Hessian [8] accelerates the computation of the scale space and the Hessian by using an approximation of the Hessian mask. The approximation is based on the use of masks composed of rectangular regions (see examples in Fig. 3), which can be convolved with the image efficiently by using integral images. An approximated determinant of Hessian layer is computed for each scale σ by using the filter responses:

$$\det(H_{approx}) = D_{xx}D_{yy} - \left(0.9D_{xy}\right)^2 \tag{21}$$

The maxima of the determinant of Hessian in (x,y,σ) are selected as interest points. This method uses the 3D Hessian scale space directly for automatic scale selection.

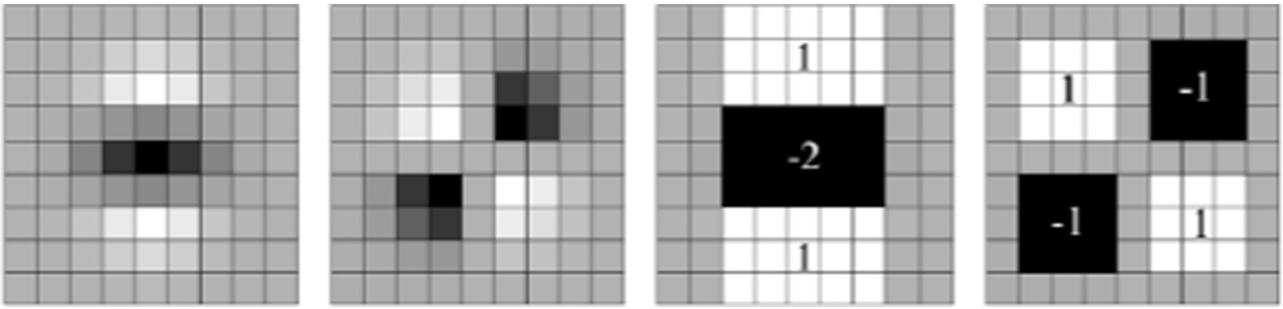

Fig. 3. Original Masks (L_{yy} and L_{xy}) are Approximated as a Sum of Square Functions (D_{xy} and D_{yy}).

Table 3
Overview of detector response at different kinds of image structures and invariance properties of different detectors (based on [68]).

Feature detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant
Harris	**yes**	no	no	**yes**	no	no
Shi–Tomasi	**yes**	no	no	**yes**	no	no
Det. of Hessian	no	**yes**	no	**yes**	no	no
SUSAN	**yes**	no	no	**yes**	no	no
Harris–Laplace	**yes**	partial	no	**yes**	**yes**	**no**
Hessian–Laplace	partial	**yes**	no	**yes**	**yes**	**no**
DoG	partial	**yes**	no	**yes**	**yes**	**no**
Fast Hessian	partial	**yes**	no	**yes**	**yes**	**no**
Harris–Affine	**yes**	**yes**	no	**yes**	**yes**	**yes**
Hessian–Affine	partial	**yes**	no	**yes**	**yes**	**yes**
MSER	no	no	**yes**	**yes**	**yes**	**yes**
FAST	**yes**	no	no	**yes**	no	no
ORB	**yes**	no	no	**yes**	**yes**	no
BRISK	**yes**	no	no	**yes**	**yes**	**no**

Table 4
Overview of performance of different detectors applied on 800 × 640 Images (based on [41,68]).

Feature detector	Repeatability	Localization accuracy	Robustness	Efficiency	Number of keypoints
Harris	+++	+++	+++	++	n/a
Shi–Tomasi	+++	+++	+++	++	n/a
Det. of Hessian	++	++	++	+	n/a
SUSAN	++	++	++	+++	n/a
Harris–Laplace	+++	+++	++	+	n/a
Hessian–Laplace	+++	+++	+++	+	n/a
DoG	++	++	++	++	1,552
Fast Hessian	++	++	++	+++	2,911
Harris–Affine	+++	+++	++	++	n/a
Hessian–Affine	+++	+++	+++	++	n/a
MSER	+++	+++	++	+++	483
FAST	+	+	+	++++	5,158
ORB	+++	++	+++	++++	594
BRISK	++	+	++	++++	1,874

3.3. *Region based detectors*

In contrast to the previously described methods, region based methods start from a segmentation perspective for finding affine covariant regions. The most well known method, MSER, was proposed by Matas et al. [37]. Other alternatives are described in [68].

3.3.1. *MSER detector*

Maximally Stable Extremal Regions (MSER) [37] correspond to image regions whose luminance values are significantly higher or lower than those in their surroundings, and are detected by using a watershed segmentation algorithm. These regions are very robust to transformations applied to the image.

3.4. *Summary statement on local interest point detectors*

Tables 3 and 4 summarize some of the main properties of the methods described. Analyses of rotation, scale, and affine invariance are presented in Table 3. In Table 4, *repeatability* indicates the proportion of matched features between two images with respect to the total number of features, i.e., the probability that a keypoint from one image will appear in a second image with the same graphical content, but taken from a different viewpoint. *Localization accuracy* indicates the position error of matching features between two images. *Robustness* is similar to repeatability, but it also includes other image alterations such as changes of illumination. *Efficiency* indicates the runtime speed achieved by the method. *Number of keypoints* indicates the number of keypoints that a method typically obtains in a given image.

It can be noted on Table 3 that all of the methods are rotation invariant. Scale invariance is of paramount importance for detecting objects in the real world, but not all of the methods have this kind of invariance. For robotic applications, the ability of the descriptors to be affine invariant is not relevant because images of the reference objects can be captured using different viewpoints for training the system.

In Table 4 it can seen that the fastest methods (highest efficiency) for detecting keypoints are FAST, ORB and BRISK, followed by SUSAN, FastHessian and MSER. Detectors that generate a large number of keypoints are more reliable since they make detection of objects that are far from the camera possible, and then cover small areas in the images. The number of detected features can be controlled in all of the methods by moving detection thresholds except in MSER, FAST, and ORB.

It can be then concluded that, given that in robotic applications the number of detected keypoints, invariance to scale, repeatability, and efficiency are the main requirements, the most suitable keypoint detectors are ORB, BRISK, Fast Hessian, and DoG.

4. Local descriptor computation and matching

4.1. Local descriptor computation

4.1.1. Local descriptors and local reference frame determination

A local descriptor is a feature vector that describes the patch around an interest point. Each patch is described by using its local reference frame, and then local descriptors are invariant respect to geometrical transformations applied to the image. The descriptors must also be robust to changes in illumination and viewpoint since, in robotic applications, the objects to be detected have different positions and orientations. As local descriptors are distinctive, the problem of matching two images can be reduced to the problem of matching two sets of distinctive local descriptors. This last procedure can be successful if the images have non-repetitive visual texture. In addition, the non-existence of visual texture produces non-existence of local descriptors, and repetitive patterns cause the local descriptors to be non-distinctive, degrading the performance of the matching process.

Interest points have some invariance properties. If an image is warped by using a geometrical transformation, its interest points will be warped in the same way. An orientation θ can be assigned to each interest point by computing the dominant orientation of the gradient around the point [32]. Then, each point corresponds with a local reference frame (x,y,σ,θ), in which (x,y) corresponds to the origin of the reference frame, σ corresponds to its scale and θ corresponds to its orientation. Local reference frames depend on structures existing in the image. If warping is applied to the image, local reference frames are also warped, and then the patches around each local reference frame are invariant with respect to geometrical transformations applied to the images.

4.1.2. SIFT descriptor

The Scale invariant feature transform (SIFT) [32] is an algorithm for generating a set of local descriptors from an image, involving both computation of the interest points (DoG points) and a SIFT descriptor. The SIFT descriptor is based on generating a histogram of the gradients inside the oriented patch. The patch, that is of size 16×16, is divided into 4×4 sub-regions (see Fig. 4). For each sub-region, a histogram including a total of 8 possible directions (separated by 45°) is computed. By concatenating the histograms from all sub-regions, a final 128-dimensional histogram is computed. The histogram is then normalized, and the sum of the square of its component is equal to 1. For more details on the computation of the SIFT histogram, see [32].

4.1.3. PCA–SIFT descriptor

The PCA–SIFT descriptor [24] is built by generating a normalized gradient patch and applying Principal Component Analysis (PCA) for reducing the dimensionality of the gradient patch, while preserving its distinctiveness. The dimensionality of the descriptor is 20, which is much smaller than the dimensionality of SIFT (128). In the original work [24], PCA–SIFT is shown to have both better precision and runtime than SIFT. However, in other studies, such as [40], PCA–SIFT did not outperform the original SIFT descriptor in distinctiveness.

4.1.4. Color SIFT descriptors

Standard SIFT descriptors are computed in grayscale images, but they can be extended to deal with color information. In that case, instead of processing the grayscale luminance values, the color channel values are processed. The different color SIFT algorithms choose different color space representations and use different mechanisms to obtain illumination and/or color invariance. Color SIFT variants include: *HSV–SIFT* [69], *HueSIFT* [69], *OpponentSIFT* [69], *C-SIFT* [1], *rgSIFT* [58], *Transformed color SIFT and RGB–SIFT* [58].

In [69], several color descriptors are compared. When using the PASCAL 2007 database, C-SIFT is the descriptor with the best performance (precision 0.44), and beats the standard SIFT descriptor (precision 0.40). In a second database (a video sequence) OpponentSIFT yields the best results (precision 0.40), surpassing the standard SIFT descriptor (precision 0.38). The mean precision improvement gained by using the best color descriptor against the standard SIFT descriptor is less than 0.05 in both databases. Color descriptors require longer computation times than SIFT descriptors because of the need for computing descriptors over several channels and using larger feature vectors. Therefore, in the case of using color SIFT descriptors, there is a trade-off between the precision of the method and its processing time. Consequently, this low precision gain (less than 0.05) is not worth in robotics applications, and color SIFT descriptors are normally not useful.

4.1.5. ASIFT descriptor

The Affine SIFT descriptor [43] is computed by generating a set of patches through warping the original patch for simulating changes of viewpoint. In each of the warped patches, a SIFT descriptor is computed. By using ASIFT descriptors generated by using different image transformations, it is possible to recognize objects with a change of viewpoints of 80%. However, the computation of several descriptors for each keypoint makes the system slow and infeasible for robotic applications. Also, changes in viewpoint can be handled by using different views from each

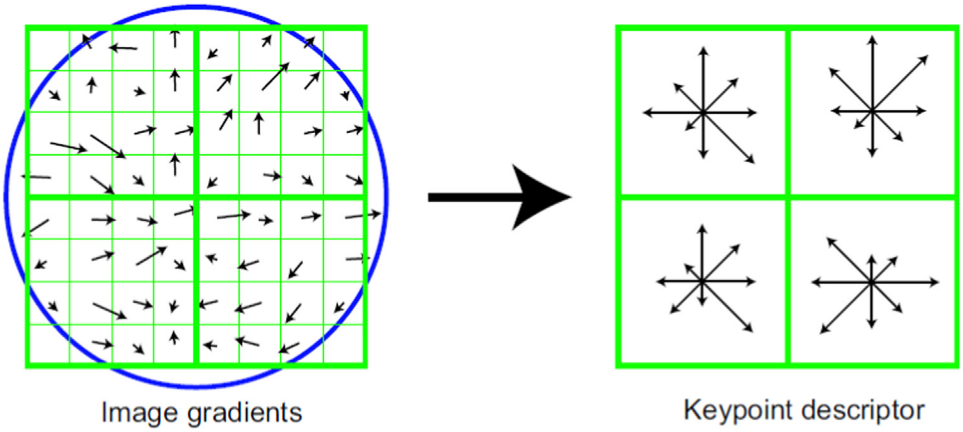

Fig. 4. SIFT Descriptors are formed by computing image gradients for each pixel in the patch. The window is divided into subregions. For each subregion, an histogram of orientations is computed. Original figure from [32].

object, making the ASIFT approach not useful for visual object recognition in robotics.

4.1.6. SURF descriptor

The SURF descriptor [9] is also based on an analysis of the gradients inside an oriented patch, which is divided into 4 × 4 sub-regions. If the components of the gradients inside each sub-region are named dx and dy, a descriptor for the sub-region is computed as (Σdx, Σdy, Σ|dx|, Σ|dy|). Given that there are 16 sub-regions, the full SURF descriptor has 64 dimensions.

4.1.7. BRIEF, ORB, BRISK, and FREAK descriptors

The BRIEF (Binary Robust Independent Elementary Features) descriptor [10] is a binary string, with each of the bits corresponding to an intensity difference test done on a smoothed image patch. The computation of this descriptor is very fast and requires only a small amount of memory. A test τ on a patch p of size $S \times S$ is defined as:

$$\tau(p; x, y) = \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where $p(x)$ is a pixel intensity in a smoothed version of p at $x=(u, v)$. The BRIEF descriptor is a n_d dimensional bit string defined as:

$$f_{n_d}(p) = \sum_{i=1}^{n_d} 2^{i-1}\tau\left(p; x_i, y_i\right) \tag{23}$$

The algorithm requires a strategy for selecting sampling pairs (x,y) for the intensity difference tests. A Gaussian sampling strategy is preferred.

As has already been explained, ORB [55] is a methodology for obtaining both keypoints and binary descriptors. The descriptor is basically an oriented BRIEF descriptor, computed at the position, scale, and orientation provided by each keypoint. The sampling strategy for the BRIEF descriptor is learned from the training images by selecting tests that are uncorrelated and have high variance.

The BRISK (Binary Robust Invariant Scalable Keypoints) descriptor [27] is also based on the generation of binary strings as descriptors. Once a keypoint is found, a precomputed regular sampling pattern is used for computing the dominant local orientation, and also for computing the descriptor. Consider that p corresponds to a sampling position. Pairs of sampling points (p_i, p_j) are used for describing the patch. Two sets of pairings, short-distance pairings and long-distance pairings, are computed as:

$$S = \left\{ (p_i, p_j) | \|p_j - p_i\| < \delta_{MAX} \right\} \tag{24}$$

$$L = \left\{ (p_i, p_j) | \|p_j - p_i\| \geq \delta_{MAX} \right\} \tag{25}$$

Long-distance pairings L are used for computing the dominant orientation of the descriptor, and short-distance pairings S are used for computing the binary strings by performing intensity comparisons as in BRIEF. The BRISK binary string contains 512 bits.

The FREAK (Fast Retina Keypoint) descriptor [2] is also based on the use of binary strings. It uses a retinal-based sampling procedure that emulates receptive fields, enabling the sampling areas to have overlap. The sequence of the pairs of samples to use is generated by using a machine learning approach in which the most discriminative pairs of samples are selected. A pair of samples is considered discriminative if it has high variance on the training data, and low correlation with the other selected pairs of samples. The 512 principal bits are computed by intensity comparisons and are used as the binary string descriptor.

4.1.8. LIOP, MROGH and MRRID descriptors

All of the previous rotation invariant descriptors require the computation of a dominant gradient orientation around the keypoint. A local reference frame is then generated, and the local descriptor is constructed on it. As shown in [14], small errors in the estimation of the orientation of the keypoints generate a noticeable loss of precision in the matching process.

The Local Intensity Order Pattern (LIOP) [70], the Multisupport Region Order-Based Gradient Histogram (MROGH) [14] and the Multisupport Region Rotation and Intensity Monotonic Invariant Descriptor (MRRID) [14] are based on sorting the pixels of their support region by their intensity. The ordered pixels are partitioned into B bins, each representing an intensity range. The bins will then have different shapes on the support region.

Each bin is composed of a set of pixels. For each of these pixels, a local reference frame is constructed centered at the keypoint and pointing to the pixel; i.e., it is a pixel-wise reference frame. In that reference frame determined by the pixel p_1, three additional pixels, p_2, p_3 and p_4 are selected at predefined positions. The four pixels are sorted by their illumination value, and the indexing order (named Local Intensity Order Pattern) is stored in a table. After that is done for all of the pixels of the selected bin, the table contains a histogram of the different possible permutations corresponding to the possible orderings of the four pixels. The LIOPs are computed and collected for each of the bins, and then concatenated into a feature vector, named the LIOP descriptor.

For computing the MROGH descriptor, each of the pixels of each bin is associated with a pixel-wise reference frame. Four pixels selected in that reference frame at predefined positions are used for computing the local gradient at the point. For each bin, a histogram of orientation of the gradients around points in the bin is computed. The histograms for the different bins are concatenated, generating a MROGH feature vector.

The MRRID descriptor is also computed by using pixel-wise reference frames. In each frame, a set of pixels at predefined positions is used for performing binary comparisons, generating a binary string. Binary strings for each of the pixels in each bin are summed into accumulated binary strings, and are concatenated over all of the bins for generating an MRRID feature vector.

4.2. Descriptors matching

Images sharing graphical content can be matched by pairing similar descriptors among them. A brute force approach consists of comparing all the possible pairs of descriptors for searching the nearest ones; however this approach becomes infeasible when the number of training images increases. Then, an approximated nearest neighbor search can be used for generating matches between descriptors from the test and training images. There are three main families of algorithms used for retrieving the nearest descriptor from the database in an efficient way: tree based algorithms, locality sensitive hashing algorithms, and fast brute force algorithms for binary descriptors.

4.2.1. Tree based Algorithms

A kd-tree can be used for storing the training descriptors; i.e. the descriptors computed in the training images. Then, approximate nearest neighbor search can be done with a method named *Best Bin First* (BBF) [32] by traversing the tree upon reaching a leaf node. That node becomes the current best. The search then continues by backtracking along the unexplored branches, searching for better current bests. Another option is to use a set of smaller kd-trees (a kd-tree forest) [16,44] for searching the approximate nearest neighbor in parallel.

Each approximate nearest neighbor search using a kd-tree requires $O(k^{*}\log(n))$, with k the number of tree traversals, and n the number of descriptors stored in the tree.

In addition, it is important to note the order in which to filter wrong matches; in [32] the following algorithm is proposed, and normally used in most implementations. For each descriptor f_t in the test image, the two nearest descriptors, f_1 and f_2 from the training database are selected, and the distances $d_1 = \text{dist}(f_t, f_1)$ and $d_2 = \text{dist}(f_t,\ f_2)$ are computed. If the distance d_1 to the nearest neighbor is short compared to d_2, the match has high confidence since the probability of confusion is low. Then, if the ratio d_1/d_2 is below a threshold, the match is accepted [32].

4.2.2. *Locality sensitive Hashing*

This approach consists of selecting hashing functions with the property that "the hashes" of elements that are close to each other are also likely to be close. There are several variants of this algorithm. The computation time needed for each approximate nearest neighbor search is $O(k)$, with k the number of request trials.

A comparative study presented in [44] indicates that approximate neighbor search in kd-trees outperforms hashing-based methods.

4.2.3. *Fast Brute Force for Binary descriptors*

Distance between binary descriptors can be computed by using a Hamming distance, i.e., by applying an Xor operation to the binary strings and then counting the number of bits in the result. This procedure is faster than kd-trees when using databases with a limited number of descriptors, and it enables getting the exact nearest neighbor. The computation time required for performing a nearest neighbor search using brute force is $O(n)$, with n the number of stored descriptors in the database. However, if the database is large enough, tree based algorithms perform better as their required computation time is logarithmic and not linear.

4.3. *Summary on local descriptors*

Tables 5 and 6 summarize some of the main properties of the descriptors described. *Compactness* is a measure of the size of the descriptor. *Efficiency* refers to the processing time required by the method; i.e., a faster method has better efficiency. *Precision*, *Recall*, and *MAP* are performance measures. When a training database is created, new descriptors are matched to the database for searching nearest neighbors. Each of the new descriptors can be matched to the corresponding descriptor, or it can be matched to an incorrect descriptor, or it cannot be matched at all. *Precision* refers to the proportion of correctly retrieved descriptors with regard to the total number of matches that were generated, i.e., $precision = \frac{\#correct\_matches}{\#total\_matches}$. *Recall* refers to the proportion of correct retrieved matches with respect to the total number of new descriptors that were considered, i.e., $recall = \frac{\#correct\_matches}{\#new\_descriptors}$.

Average Precision (AP) is a measure of performance for a ROC curve. It is defined as the mean of the precision $p(r)$ along all of the recall values r. Mean Average Precision (MAP) is the mean of several average precisions obtained from several different ROC curves.

Table 5
Properties (invariance, compactness, efficiency) for Local Descriptors.

Descriptor	Rotation invariant	Scale invariant	Compactness	Efficiency (Processing time)
SIFT	Yes	Yes	+	++
PCA–SIFT	Yes	Yes	++	++
Color SIFT	Yes	Yes	+	+
SURF	Yes	Yes	++	++
BRIEF	No	No	+++	++++
ORB	Yes	Yes	+++	++++
BRISK	Yes	Yes	+++	++++
FREAK	Yes	Yes	+++	++++
ASIFT	Yes	Yes	+	+
LIOP	Yes	Yes	+	+
MROGH	Yes	Yes	+	+
MRRID	Yes	Yes	+	+

Table 6
Precision and Recall for several Interest Point Detectors and Descriptors. Data from [41].

Descriptor	Detector	Precision	Recall	MAP
SURF	Fast Hessian	0.485	0.513	0.334
SIFT	Fast Hessian	0.525	0.533	0.491
BRIEF	Fast Hessian	0.517	0.546	0.514
ORB	Fast Hessian	0.448	0.470	0.437
BRISK	Fast Hessian	0.536	0.553	0.530
BRISK	BRISK	0.504	0.527	0.492
ORB	ORB	0.493	0.495	0.463
FREAK	FREAK	n/a	n/a	n/a
SIFT	FAST	0.366	0.376	0.336

In Table 5, it can be noted that all of the descriptors, with the exception of BRIEF, are rotation and scale invariant, and that binary descriptors (BRIEF, ORB, BRISK, FREAK) outperform the other available options both in compactness and runtime. In Table 6, it can seen that the best performance with respect to precision, recall, and MAP is achieved by BRISK descriptors applied on FastHessian keypoints. Second-best results are obtained by BRIEF with FastHessian keypoints, followed by BRISK with BRISK keypoints, and SIFT with FastHessian keypoints. When considering only binary keypoint detectors and binary descriptors, the best precision, recall and MAP is achieved again by the BRISK algorithm.

Using Tables 5 and 6 we can conclude that in robotics applications the most suitable descriptors are ORB, BRISK, SIFT and SURF. BRIEF is not included because it is not rotation and scale invariant.

5. Geometric verification of matched features

5.1. *Transformation computation*

Each match between two local descriptors generates a similarity transformation hypothesis. Sets of compatible geometric transformations can indicate the presence of the searched object inside the image, and therefore, they need to be computed. The two most popular methods for computing the geometric transformations are Hough transform [8] and the RANSAC algorithm [15].

5.1.1. *Hough transform*

A similarity transformation is defined by four parameters: a translation $(t_X,\ t_Y)$, a rotation θ and a scale change s. By using the Hough Transform procedure, the parameter space can be quantized into a set of bins, each having an accumulator that starts empty. Each match between two descriptors generates a similarity transformation hypothesis, and a vote in one of the bins in the parameter space [7,32]. When the object to be detected is in the test image, matches related to it (inliers) are accumulated in one of the bins, while spurious matches (outliers) are spread over the parameter space. Bins with a high number of votes are selected as candidate detections. The Hough transform procedure can work even when the number of outliers is much larger than the number of inliers.

5.1.2. *RANSAC*

When the number of inliers is high enough, the RANSAC algorithm [15] can be applied for searching the correct transformation. One of the oriented descriptors is selected for generating a similarity transformation hypothesis, while the others are compared with the hypothesis. If the number of compatible matches, i.e., the consensus, is over a threshold, the hypothesis is accepted; if not, another descriptor is selected for generating a new hypothesis and the procedure is repeated. RANSAC is faster than Hough when both the number of objects to recognize and the number of outliers are low. If these conditions are not fulfilled, the mean number of iterations needed for RANSAC to achieve the required consensus grows.

5.2. *Hypotheses verification stages*

In complex, real-world object recognition problems, the initial hypothesis set (i.e. geometric transformation) may contain a large number of false detections. For handling incorrect hypotheses, several verification stages need to be used.

The most widely used hypothesis verification methodology is the one proposed by Lowe in his seminal work [32]. On the other hand, the L&R method, based on Lowe's method, adds several interesting verification stages. Both methods are presented.

5.2.1. *Lowe's method*

In [32], a Hough transform over the similarity transformation space is used for generating candidate detections. For each bin in the Hough space, the following steps are performed:

a) Bin filtering: bins with less than 3 votes are rejected
b) Affine transformation determination: An affine transformation is determined by using the matches inside the current bin. A least-squares method is used for obtaining the best transformation parameters (m_{11},m_{12},m_{21},m_{22},t_x,t_y) as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \Rightarrow \begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}\begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} u \\ v \\ \dots \end{pmatrix} \tag{26}$$

$$\begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} = X, \quad \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{pmatrix} = \vec{m}, \quad \begin{pmatrix} u \\ v \\ \dots \end{pmatrix} = \vec{u} \tag{27}$$

$$X\vec{m} = \vec{u} \quad \Rightarrow \quad \vec{m}^* = \left(X^T X\right)^{-1} X^T \vec{u} \tag{28}$$

c) Lowe's probability computation: A probability of the transformation being correct is computed. This probability depends on the number of matches compatible with the current transformation, and the total number of matches between both images. If the probability is lower than a threshold (95%), it is rejected.
d) Top-down matching: Matches over the Hough space that are inside another bin but are compatible with the current transformation are collected into the current bin.

5.2.2. *L&R method*

After the Hough transform, an extra set of tests for rejecting incorrect transformations, named L&R verification stages [57], can be performed. The fastest tests are performed before the slower ones for obtaining a fast procedure for transformation verification. The tests are applied in the following order:

a) Bin filtering: bins with less than 3 votes are rejected
b) Non-maxima suppression: Bins that are no local maxima in the Hough space are rejected.
c) Linear correlation test: Descriptors that voted for the current bin determine a set of interest points in both the training and the current image. If the interest points in one of the two images lie on a line, the affine transformation will be undefined in the direction perpendicular to the line. Therefore, correlation coefficients are calculated in both images, and used for rejecting unstable transformations (see [57]).
d) Fast probability computation: A fast algorithm for computing the probability that the current bin corresponds to a correct detection is applied (see details [30]).
e) Affine transformation determination: An affine transformation is determined by using the matches inside the current bin.
f) Geometrical distortion test: The amount of affine distortion is used for rejecting wrong transformations because the local descriptors can work only with out-of-the-plane rotations of about 45°, and that bound the amount of acceptable affine distortion.
g) Lowe's Top-down matching.
h) Lowe's probability computation: A probability of the transformation being correct is computed, as in Lowe's method.
i) RANSAC test: If the current set of selected matches is composed mostly of inliers, then a fast RANSAC test is applied for rejecting outliers that could not be rejected by the previous steps.
j) Transformation fusion: Because of the quantization done in the Hough transform procedure, a correct object match spread votes over several neighbor bins, producing multiple detections. Then, all possible pairs of transformations are compared by using a fast procedure for detecting duplicate detections. Duplicated transformations are fused.
k) Pixel correlation test: A pixel-level correlation is computed by comparing the pixel intensity values from the training image against the projected values in the current test image. If the pixel correlation value is lower than a threshold, the transformation is rejected.

5.3. *Summary on geometric verification*

An object recognition method requires high precision and high recall in order to obtain successful performance. The effect of using several consecutive procedures for rejecting false positives is relevant in real-world problems where clutter background and multiple objects are present. In [57], how the use of such a method – L&R in that case – is able to reduce the false positive rate greatly (from 81.9% to a 3.2%!) is shown. A comparative study of several object recognition methods that address this issue and that confirms the convenience of using several geometric verification tests is presented in the next section.

6. Case study: a comparative study of object recognition systems in real domestic settings

As a case study, we evaluated several object recognition systems using local invariant features in a service-robot's domestic environment. The final task to be performed by the service robot was the manipulation of objects in a domestic setup; and the robot's ability to recognize objects placed on a planar surface (e.g. a table or a shelf) was evaluated. The different object recognition systems were built using different combinations of local interest detectors, local descriptors and geometric verification stages.

6.1. *Experimental setup and methodology*

A service robot, Bender [56], was used as the platform for testing the different object recognition approaches. The robot has a Kinect camera and an RGB camera mounted over its head. Both are placed at a height of approximately 1.6 [m], pointing downwards with an angle of 56° with respect to the horizon. The Kinect has a resolution of 640×480 and an angular field of view of 57° horizontally and 43° vertically. The RGB camera has a resolution of 1280×720 pixels, and has an angular field of view of 60° horizontally and 45° vertically. In the experiments reported, two kinds of object placements were used: objects were placed on a table or on a shelf (see Fig. 5). When placed on the table/shelf, the mean distance between the robot's camera and the objects was 104.1 [cm]/40 [cm].

A set of 40 objects was selected for performing the tests; the objects are those typical found in a home-like environment (see Fig. 6). From the set of objects, 20 have visual textures, and the other 20 objects have uniform surfaces (no textures). For each object, 12 different views were captured by rotating the objects 30° between two consecutive frames. For each view, both an RGB and a depth image were captured. Thus, a total of 480 RGB images were used as the gallery (database) in the object recognition experiments. Depth images were used only for segmenting the objects from the table/shelf.

Different setups, corresponding to the actual conditions of a domestic environment, were used for evaluating the performance of the different object recognition methodologies under comparison. The possible setups differ in the following conditions: (a) distance to the object: 40 cm/104.1 cm; (b) surface background: white/colored/cluttered; (c) Illumination: normal/low; (d) occlusion: no occlusion/50% occlusion. Then, for each camera-object distance, the testing setups were the following:

- S1: One object, white background, normal illumination, no occlusion.
- S2: One object, white background, low illumination, no occlusion.
- S3: One object, colored background, normal illumination, no occlusion.
- S4: One object, cluttered background, normal illumination, no occlusion.
- S5: One object, white background, normal illumination, 50% occlusion.

For each of these 5 setups and for each camera–object distance, 160 experiments were carried out by selecting each object 4 times; each time the object's view was chosen randomly. The random view is selected by putting the object inside the field of view of the cameras, and then selecting a random number between 0° and 360° for setting the object's orientation. The recognition is considered successful if the correct object is identified, independently of the recovered viewpoint; i.e., the current object must be matched correctly with one among the 480 images (40 objects) in the database.

In order to evaluate the performance of the different methods, the precision, recall and F1 score from the detection statistics are computed. Precision and recall are computed as TPR/(TPR+FPR) and TPR/(TPR+FNR), respectively, with TPR the true positive rate, FPR the false positive rate, and FNR the false negative rate. The $F1$ score is computed as the harmonic mean of the precision and recall measures: $F1=2^*(\text{precision}^*\text{recall})/(\text{precision}+\text{recall})$. In addition, the execution time of the different methods is also analyzed.

6.2. Methods under comparison

In recognizing objects using local invariant features, the performance of the final system will depend on the local interest points and descriptors being used, as well as on the geometric verification stages being implemented. Therefore, various object recognition systems are built using different combinations of local interest detectors, local descriptors and geometric verification stages.

Based on the conclusions of Sections 3.4 and 4.3, ORB, BRISK, Fast Hessian, and DoG are selected as keypoint detectors, and ORB, BRISK, SIFT, and SURF as descriptors. Regarding geometric verification, two approaches are selected: the use of RANSAC for searching a similarity transformation, which is a popular choice when implementing robotics applications, and the use of the L&R method, which includes several geometric transformations and hypothesis verification stages (see Section 5.2). Then the following object recognition systems are built and compared:

- ORB–ORB: ORB keypoints and ORB descriptors.
- ORB–BRISK: ORB keypoints and BRISK descriptors.
- BRISK–BRISK: BRISK keypoints and BRISK descriptors.
- DoG–SIFT: DoG keypoints and SIFT descriptors.
- FHessian–SURF: Fast Hessian keypoints and SURF descriptors.
- DoG–SIFT–GPU: DoG keypoints and SIFT descriptors.

In the case of ORB–ORB, BRISK–ORB and BRISK–BRISK, the OpenCV [73] implementation of the algorithms is used. In the case of DoG–SIFT and FHessian–SURF, the authors' own implementation is used. For DoG–SIFT–GPU, the code available in [74] is used. It was decided to test two independent implementations of the popular "SIFT object recognition method" (DoG–SIFT), in order to analyze how the final performance of the methods depends on the method's implementation.

Each method was tested when used together with RANSAC and with the L&R verification method. All experiments were run on an

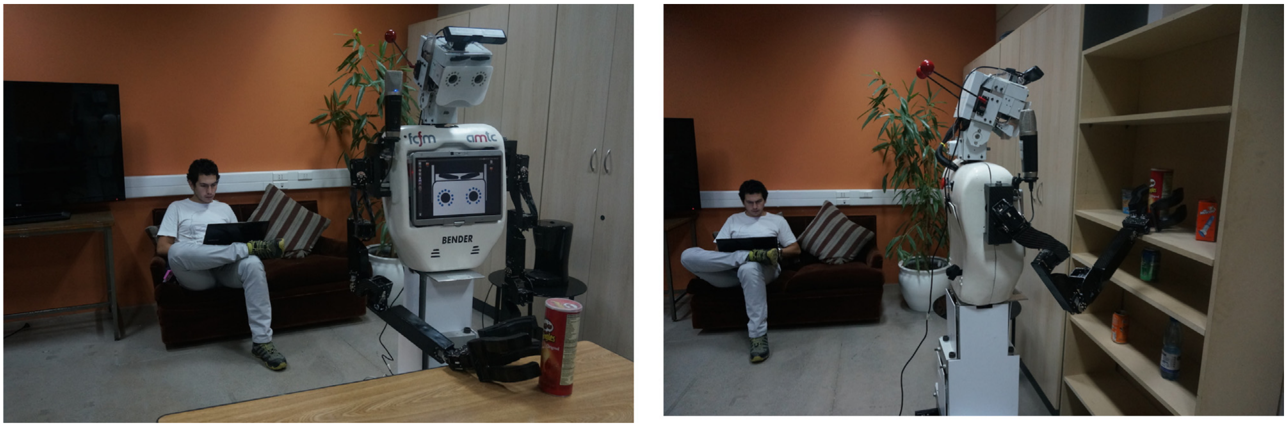

Fig. 5. Bender robot observing an object on a table (left) and on a Shelf (right).

Intel Core i7 CPU 920 @ 2.67 GHz x8 with a GeForce GTX 285/PCIe/SSE2 card.

6.3. Results

Tables 7 and 8 show the performance of the different systems under comparison, in terms of the F1 score, for the five different experimental setups, considering two different distances from the objects.

The first observation is that, as expected (e.g. [57]), the performance of the systems depends heavily on the use of appropriate hypothesis verification stages. For each system, the performance increases appreciably when simple and complex geometric verification stages, which consider geometric transformations computation as well as hypothesis verification procedures, are used (L&R method case), compared to the situation where a single hypothesis verification method that looks for a geometric transformation among the matches is used (RANSAC case). In Tables 7 and 8 the *Increase* row corresponds to the % of increase when using the L&R method compared to the RANSAC case. The performance increase is greater when recognition conditions are harder (e.g. low resolution of the objects in the images). It can therefore be concluded that it is important to use geometric transformation computation as well as hypothesis verification procedures when implementing object recognition systems based on local invariant features.

A second observation is that, as reported in [36], the performance changes appreciably depending on the resolution of the objects in the images. Thus, in the two typical situations that are reported (a real service robot observing objects placed on a table, and on a shelf), the performance of the methods decreases with the distance from the objects, and this decrease is much larger for some methods. When comparing the corresponding *Mean* row values between Tables 7 and 8, it can be observed that the mean *F*1 score of ORB–BRISK, BRISK–BRISK, DoG–SIFT–GPU, and FHessian–SURF decrease by more than 40% when the distance from the objects is 104.1 cm compared to the case when this distance is 40 cm. On the other hand, the score decrease of the ORB–ORB and DoG–SIFT methods is 27% and 13%, respectively, when the L&R verification method is used. Thus, these two keypoint–descriptor combinations are more robust to conditions under which the resolution of the objects is low.

From the experiments reported, it can be observed that for each system, the performance decreases when the illumination is low (setup S2), which is typical in some actual domestic environments, when the background is complex and cluttered (setups S3 and S4), and when parts of the objects are occluded (setup S5). Thus, as can be seen readily, when the experimental conditions are not very challenging (e.g. in setup S1 at a short distance from the objects), the performance of the different systems is somewhat similar, and no large variations are observed. However, this situation changes dramatically when the conditions are more difficult. For example, when the illumination is low (setup S2), the differences in performance between the systems are greater than 500%. Clearly, the BRISK–BRISK method is not robust to large changes in illumination. A similar situation occurs when cluttered backgrounds are included in the experiments (setup S3); i.e. differences in performance greater than 500% are observed. When, parts of the objects are occluded, the variations in performance between the different systems are important but smaller (maximum variations of 50%).

In summary, when the distance from the objects is moderate (40 cm) and the proper verification stages are used, the three best performing keypoint–descriptor combinations, with small variations in the *F*1 score, are ORB–ORB (highest *F*1 score), DoG–SIFT (second highest *F*1 score), and ORB–BRISK (third highest *F*1 score). But when the distance from the objects is high (104.1 cm), the best

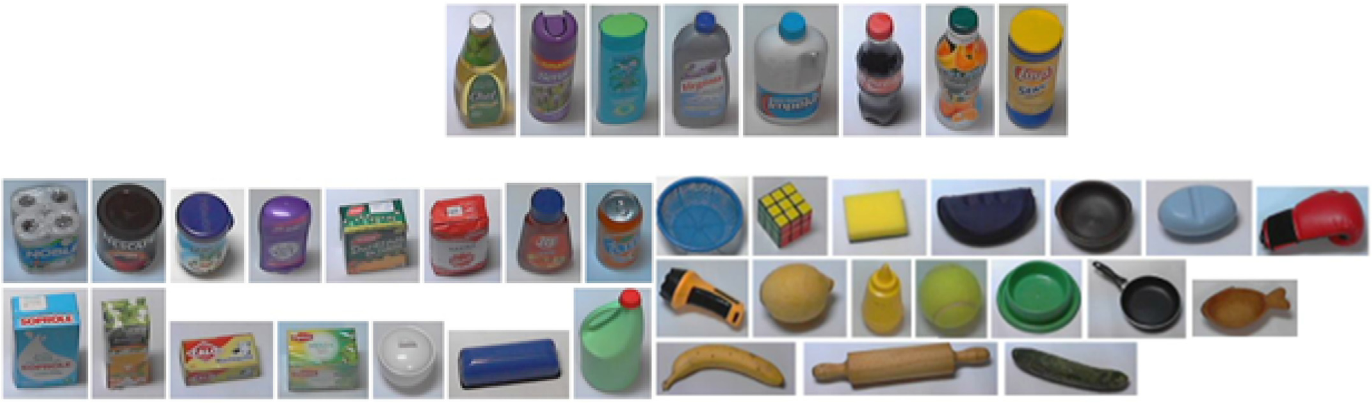

Fig. 6. Set of 40 Objects used for evaluating the Object Recognition Algorithms.

Table 7
*F*1 Scores for the different methods under different setups (S1–S5). Distance to the object: 40 cm. Mean: Mean score between the different setups. Increase: % increase when comparing the same method using RANSAC and L&R.

Verifier	**RANSAC**						**L&R**					
Keypoint detector **Descriptor**	**ORB** **ORB**	**ORB** **BRISK**	**BRISK** **BRISK**	**DoG** **SIFT–GPU**	**DoG** **SIFT**	**FHessian** **SURF**	**ORB** **ORB**	**ORB** **BRISK**	**BRISK** **BRISK**	**DoG** **SIFT–GPU**	**DoG** **SIFT**	**FHessian** **SURF**
S1	0.694	0.740	0.171	0.752	0.663	0.663	0.852	0.848	0.463	0.758	0.879	0.810
S2	0.316	0.192	0.049	0.376	0.425	0.413	0.585	0.500	0.081	0.500	0.566	0.512
S3	0.539	0.375	0.037	0.394	0.406	0.475	0.755	0.675	0.209	0.697	0.642	0.591
S4	0.498	0.289	0.037	0.340	0.356	0.419	0.664	0.611	0.141	0.602	0.626	0.588
S5	0.367	0.242	0.084	0.376	0.350	0.275	0.608	0.643	0.266	0.711	0.654	0.483
Mean	0.483	0.368	0.076	0.442	0.440	0.449	0.693	0.655	0.232	0.654	0.673	0.597
Increase	–	–	–	–	–	–	43%	78%	205%	48%	53%	33%

performing keypoint–descriptor combination is DoG–SIFT and the second best is ORB–ORB. The performance of the other methods is much less in this case.

It is interesting to note that important differences between the two "SIFT" implementations are observed. DoG–SIFT (our own implementation) is much more robust than DoG–SIFT–GPU. Differences in performance are higher than 90% in some cases. On the other hand, DoG–SIFT–GPU is much faster; ten times faster in some cases.

Table 9 shows the processing time of the different methods. It can be seen that for each method the processing time decreases when the objects are at a greater distance (fewer keypoints and descriptors to be computed), and, in general terms, it increases when more verification stages are used. The fastest keypoint–descriptor combination, ORB–ORB, is about 6 times faster than DoG–SIFT when the distance to the objects is 104.1 cm.

6.4. Discussion

The analyzed methods have very different computational complexities, which depends on the computational requirements of the used local interest point detectors and local descriptors, as well as on the implemented verification stages. The first successful methodology for building object recognition systems using local features was DoG–SIFT. In Tables 8 and 9 it is shown that this method achieves high F1 scores, the highest in the case of the experiments reported in Table 9. However, the method is not able to run in real time, as it requires around 7 seconds for processing a 1280 × 720 image (see Table 9). As real time processing is needed in many applications, different strategies were developed in order to reduce the processing time. Both the use of GPUs and the development of faster alternatives (SURF, BRISK, ORB) were successful at improving the processing speed. Computing each SIFT descriptor requires to process a 16 × 16 patch by computing gradients and generating a 128-dimensional histogram. The histogram considers 4x4 spatial sub-regions, and 8 possible orientations. As each pixel must vote by its 8 nearest neighbors along three dimensions, 16*16*3 = 768 comparisons must be computed. Also, as each pixel vote by its 8 nearest bins in the histogram, the pooling operations require 16 × 16 × 8 = 2,048 additions for computing each descriptor. SURF requires computing four gradient statistics for 4x4 spatial sub regions in the image in a 20x20 patch. Each pixel votes only for its nearest sub-region. Then, computing a SURF descriptor requires 20*20*2 = 800 comparisons and 20*20*2 = 800 additions. Binary descriptors like ORB require only 256 binary comparisons to be computed. In addition, the speed of the descriptor's matching process depends of the descriptors length and on the nature of the descriptors (the evaluation of binary descriptors is faster). When analyzing why SURF is faster than SIFT, it should be taken into account that SURF tends to generate less keypoints, and therefore less descriptors need to be computed and compared.

A second aspect to analyze is that the geometric verification of the matches was not considered when faster methods were designed. But, as it can be observed in Table 9, this is a very relevant factor when using fast descriptors. When using ORB–ORB descriptors for recognizing objects at 40 cm from the robot, the use of RANSAC requires 1.743 s for processing a frame, while the use of L&R reduces the time down to 0.258 s. When observing an object at 104.1 cm from the robot, using RANSAC requires 0.9 seconds for processing a frame, while L&R requires only 0.155 s This behavior is explained because RANSAC depends on the inlier ratio from the matches that decrease when detecting multiple objects, but L&R has fixed computational requirements. Then, for obtaining both fast and reliable results in multi-object detection, the ORB–ORB with L&R verification procedure is recommended.

A third important aspect to analyze is the high dependency of some methods on the resolution of the objects in the images. It can be said, as a general rule, that methods that make any kind of approximations when computing the interest points and the descriptors are less robust to variation on the image resolution. For instance, the scale-space computation used in FastHessian is an approximation based on integral images. Also, BRISK use a coarse scale space representation. ORB uses the FAST detector for generating keypoints that are then tested against a localized Harris scoring in the scale space, but BRISK does not perform this test. Finally, BRISK is based on a handcrafted sampling pattern, while ORB's sampling pattern has more randomness and is designed for obtaining uncorrelated samples. Because of the explained factors, FastHessian and BRISK are not robust when dealing with low

Table 8
*F*1 Scores for the different methods under different setups (S1–S5). Distance to the object: 104.1 cm. Mean: Mean score between the different setups. Increase: % increase when comparing the same method using RANSAC and L&R.

Verifier	**RANSAC**						**L&R**					
Detector	**ORB**	**ORB**	**BRISK**	**DoG**	**DoG**	**FHessian**	**ORB**	**ORB**	**BRISK**	**DoG**	**DoG**	**FHessian**
Descriptor	**ORB**	**BRISK**	**BRISK**	**SIFT–GPU**	**SIFT**	**SURF**	**ORB**	**BRISK**	**BRISK**	**SIFT–GPU**	**SIFT**	**SURF**
S1	0.222	0.212	0	0.092	0.413	0.238	0.626	0.380	0.107	0.562	0.846	0.565
S2	0.012	0.061	0	0	0.165	0.057	0.437	0.399	0	0.117	0.520	0.162
S3	0.012	0.025	0	0	0.193	0.091	0.543	0.407	0.012	0.269	0.472	0.220
S4	0.012	0.012	0	0	0.156	0.113	0.414	0.315	0	0.161	0.362	0.060
S5	0.061	0.049	0	0	0.165	0.132	0.517	0.383	0.060	0.414	0.729	0.469
Mean	0.064	0.072	0	0.0184	0.218	0.126	0.507	0.377	0.036	0.305	0.586	0.295
Increase	–	–	–	–	–	–	692%	424%	–	1558%	169%	134%

Table 9
Mean processing time (in Seconds) of the different methods when objects are observed at 40 and 104.1 cm.

Verifier	**RANSAC**						**L&R**					
Detector	**ORB**	**ORB**	**BRISK**	**DoG**	**DoG**	**FHessian**	**ORB**	**ORB**	**BRISK**	**DoG**	**DoG**	**FHessian**
Descriptor	**ORB**	**BRISK**	**BRISK**	**SIFT–GPU**	**SIFT**	**SURF**	**ORB**	**BRISK**	**BRISK**	**SIFT–GPU**	**SIFT**	**SURF**
40 cm	1.743	2.133	0.970	0.949	7.172	0.387	0.258	0.769	3.044	0.830	7.494	0.328
104.1 cm	0.900	0.643	0.138	0.653	1.265	0.244	0.155	0.809	1.230	0.496	1.181	0.279

resolution images at test time. In contrast, object recognition systems built using DoG–SIFT and ORB–ORB combinations are more robust.

In conclusion, the experiments and analysis reported here show that when real conditions are considered, the use of appropriate geometric verification stages is a must, and the two best performing keypoint–descriptor combinations are ORB–ORB and DoG–SIFT; ORB–ORB is faster and DoG–SIFT more robust. So, the selection of the method to be used will depend on the tradeoff between robustness and speed.

7. Conclusions

In this survey, a complete analysis of object recognition methods based on local invariant features from a robotics perspective was presented. The survey includes a brief description of the main algorithms described in the literature, with specific analyses of local interest point computation methods, local descriptor computation and matching methods, as well as geometric verification methods.

Different algorithms were analyzed by considering the main requirement of robotics applications: real-time operation with limited on-board computational resources and constrained observational conditions derived from the robot geometry (e.g. limited camera resolution). Evaluations in terms of accuracy, robustness, and efficiency were presented. In addition, various object recognition systems built using different keypoint–descriptor combinations were evaluated in a service-robot domestic application, where the final task to be performed by a service robot was the manipulation of objects.

From the results reported it can be concluded that for robotics applications (i) the most suitable keypoint detectors are ORB, BRISK, Fast Hessian, and DoG, (ii) the most suitable descriptors are ORB, BRISK, SIFT, and SURF, (iii) the final performance of object recognition systems using local invariant features under real-world conditions depends strongly on the geometric verification methods being used, and (iv) the best performing object recognition systems are built using ORB–ORB and DoG–SIFT keypoint–descriptor combinations. ORB–ORB based systems are faster, while DoG–SIFT are more robust to real-world conditions.

Acknowledgments

This work was partially funded by FONDECYT Grant 1130153.

References

[1] A.E. Abdel-Hakim, and A,A,Farag, CSIFT: A SIFT descriptor with color invariant characteristics, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, USA, 2006, pp. 1978–1983.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst, Freak: Fast retina keypoint, in: CVPR '12 Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 510–517.

[3] F. Alhwarin., C. Wang, D. Ristić-Durrant, and A. Gräser,. Improved SIFT-features matching for object recognition. In Proceedings of the 2008 international conference on Visions of Computer Science: BCS International Academic Conference (VoCS'08), 2008, pp. 179-190.

[4] U. Asif, M. Bennamoun, & F.A. Sohel, Efficient RGB-D object categorization using cascaded ensembles of randomized decision trees. Robotics and Automation (ICRA), 2015 IEEE International Conference on, 2015, pp. 1295-1302.

[5] U. Asif, M. Bennamoun, F. Sohel, Unsupervised segmentation of unknown objects in complex environments, Autonom. Robots (2015) 1–25.

[6] P. Azad, T. Asfour, and R. Dillmann, 2009. Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition, in: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09), IEEE Press, Piscataway, NJ, USA, pp. 4275-4280.

[7] D.H. Ballard, Generalizing the Hough transform to detect Arbitrary Shapes, Pattern Recognit. 13 (2) (1981) 111–122.

[8] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, SURF: Speeded Up Robust Features, Comput. Vis. Image Underst. (CVIU) 110 (3) (2008) 346–359.

[9] P.R. Beaudet, Rotationally invariant image operators, in: Proceedings of the 4th International Joint Conference on Pattern Recognition, November 1978, pp. 579–583.

[10] M. Calonder, V. Lepetit, C. Strecha and P. Fua, BRIEF: Binary robust independent elementary features, in: ECCV'10 Proceedings of the 11th European conference on Computer vision: Part IV. Springer Berlin, Heidelberg, 2010, pp. 778–792.

[11] A. Collet Romea, D. Berenson, S. Srinivasa and D. Ferguson, Object recognition and full pose registration from a single iImage for robotic manipulation, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09), May, 2009, pp. 48–55.

[12] M. Danieletto, M. Mina, A. Zanella, P. Zanuttigh, E. Menegatti, Recognition of smart objects by a mobile robot using SIFT-based image recognition and wireless cCommunication, ECMR (2009) 73–79.

[13] S. Effendi, R. J. Jarvis and D. Suter: Robot manipulation grasping of recognized objects for assistive technology Ssupport using stereo vision, Australasian Conference on Robotics and Automation, ACRA, 2008 01/2008.

[14] B. Fan, F. Wu, Z. Hu, Rotationally invariant descriptors using intensity order pooling, IEEE Trans. Pattern Anal. Mach. Intell. 34 (10) (2012) 2031–2045.

[15] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Comm. of the ACM 24 (6) (1981) 381–395, http://dx.doi.org/10.1145/358669.358692.

[16] FLANN - ⟨http://www.cs.ubc.ca/research/flann/⟩.

[17] D. Gossow, P. Decker, D. Paulus, An evaluation of open source SURF implementations. RoboCup 2010: Robot Soccer World Cup XIV, Springer, Berlin Heidelberg 2011, pp. 169–179.

[18] K. Grauman, B. Leibe, Visual Object Recognition, Morgan & Claypool Publishers, 2011.

[19] Y. Han, Recognize objects with three kinds of information in landmarks, Pattern Recognit. 46 (11) (2013) 2860–2873.

[20] C. Harris and M. Stephens, A combined corner and edge detector, in: Proceedings of the 4th Alvey Vision Conference. pp. 147–151.a, 1988.

[21] H. Jegou, M. Douze, C. Schmid, P. Perez, Aggregating local descriptors into a compact image representation. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 13-18 June 2010, pp. 3304–3311.

[22] Z. Jia, A. Saxena, and T. Chen. Robotic object detection: learning to improve the classifiers using sparse graphs for path planning, in: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Three (IJCAI'11), 2011, pp. 2072–2078.

[23] T.K. Kang, I.H. In-Hwan Choi, M.T. Lim, MDGHM-SURF: a robust local image descriptor based on modified discrete Gaussian–Hermite moment, Pattern Recognit. 48 (3) (2015) 670–684.

[24] Y. Ke, and R. Sukthankar, PCA-SIFT: A More Distinctive Representation for Local Image Descriptors , in: CVPR'04 Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. 506–513.

[25] R. Kouskouridas E. Badekas A. Gasteratos. Simultaneous Visual Object Recognition and Position Estimation Using SIFT Proceedings of ICIRA, 2009, pp. 866–875.

[26] D. Kragic, M. Bjorkman, H.I. Christensen, J.O. Eklundh, Vision for robotic object manipulation in domestic settings, Robot. Autonom. Syst. 52 (1) (2005) 85–100.

[27] S. Leutenegger, M. Chli, R.Y. Siegwart, R.Y., BRISK: Binary robust invariant scalable keypoints. Computer Vision (ICCV), IEEE International Conference on. IEEE, 2011, pp. 2548–2555.

[28] Y. Li, C.F. Chen, and Allen, P.K.: Recognition of Deformable Object Category and Pose. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, June 2014, pp. 5558–5564.

[29] T. Lindeberg, Feature detection with automatic scale selection, Int. J. Comput. Visio.n 30 (2) (1998) 77–116.

[30] P. Loncomilla J. Ruiz-del-Solar. A fast probabilistic model for hypothesis rejection in SIFT-based Object Recognition Lecture Notes in Computer Science, vol. 4225 (CIARP 2006), Springer, pp.696–705.

[31] M. Lopez-de-la-Calleja, T. Nagai, M. Attamimi, M. Nakano-Miyatake, H. Perez-Meana, Object detection using SURF and superpixels, J.Softw. Eng. Appl. 6 (9) (2013) 511–518.

[32] D.G. Lowe, Distinctive image features from Scale-Invariant Keypoints, Int. J Comput. Vis. 60 (2) (2004) 91–110.

[33] H. Lu, Z. Zheng, Two novel real-time local visual features for omnidirectional vision, Pattern Recognition 43 (12) (2010) 3938–3949.

[34] M. Madry, D. Song, C. Ek, D. Kragic, Robot bring me something to drink from: object representation for transferring task specific grasps, in: The ICRA'12 Workshop on Semantic Perception, Mapping and Exploration, SPME, 2012.

[35] G. Manfredi, M. Devy, D. Sidobre, Textured Object Recognition: Balancing Model Robustness and Complexity. Computer Analysis of Images and Patterns, Springer International Publishing 2015, pp. 52–63.

[36] L. Martinez P. Loncomilla J. Ruiz-del-Solar. Object recognition for manipulation tasks in real domestic settings: A comparative study RoboCup Symposium, Joao Pessoa, Brazil, 2014, pp. 207–219.

[37] J. Matas, O.Chum, M. Urban, and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, in: Proc. of British Machine Vision Conference, 2002, pp. 384-396.

[38] Z. Miao, X. Jiang, Interest point detection using rank order LoG filter, Pattern Recognit 46 (11) (2013) 2890–2901.

[39] K. Mikolajczyk, C. Schmid, Scale and affine invariant interest point detectors, Int. J. Comput.r Vis. 60 (1) (2004) 63–86, http://dx.doi.org/10.1023/B:VISI.0000027790.02288.f2.

[40] K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1615–1630.

[41] O. Miksik, K. Mikolajczyk, Evaluation of Local Detectors and Descriptors for Fast Feature Matching. Pattern Recognition (ICPR, 21st International Conference on, pp. 2681–2684.

[42] H. Moravec, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute, 1980.

[43] J.M. Morel, G. Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison April 2009 SIAM Journal on Imaging Sciences archive, Vol. 2, Society for Industrial and Applied Mathematics Philadelphia PA, USA, 2009, pp. 438–469.

[44] M. Mujam, D.G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, Pattern Anal. Mach. Intell. 36 (11) (2014) 2227–2240.

[45] H. Nie, K. Long, J. Ma, D. Yue, J. Liu, Using an improved SIFT Algorithm and Fuzzy closed-loop control strategy for object recognition in cluttered scenes, PLoS One 10 (2) (2015) e0116323, http://dx.doi.org/10.1371/journal.pone.0116323.

[46] G.G. Patil, Vision guided pick and place robotic arm system based on SIFT, Int. J. Sci. Eng. Res. 4 (9) (2013) 242–248, IF 1.4.

[47] P. Piccinini, A. Prati, R. Cucchiara, Real-time object detection and localization with SIFT-based clustering, Image Vis. Comput. 30 (8) (2012) 573–587.

[48] S. Pillai, J. Leonard, Monocular SLAM Supported Object Recognition, arXiv Preprint arXiv 1506 (2015) 01732.

[49] V. Ramanathan, A. Pinz, Active Object Categorization on a Humanoid Robot, VISAPP' 11 (2011) 235–241.

[50] A. Ramisa, S. Vasudevan, D. Aldavert, R. Toledo and R. Lopez de Mantaras, Evaluation of the SIFT Object Recognition Method in Mobile Robots, in: Proc. of the 2009 Conf. on Artificial Intelligence Research and Development, pp. 9-18, 2009.

[51] F. Rigual, A. Ramisa, G. Alenyà and C. Torras, Object detection methods for robot grasping: experimental assessment and tuning. Artificial intelligence research and development: proceedings of the 15th International Conference of the Catalan Association for Artificial Intelligence. Alacant: IOS Press, 2012, pp. 123-132.

[52] E. Rosten and T. Drummond, Fusing points and lines for high performance tracking. ICCV '05 Proceedings of the Tenth IEEE International Conference on Computer Vision, Vol. 2, pp. 1508–1515.

[53] E. Rosten and T. Drummond, Machine learning for high-speed corner detection. European Conference on Computer Vision. Volume 3951 of the series Lecture Notes in Computer Science, 2006, pp 430–443.

[54] E. Rosten, G. Reitmayr, and T. Drummond, Real-time video annotations for augmented reality., in: SVC'05 Proceedings of the First international conference on Advances in Visual Computing, pp. 294–302.

[55] E. Rublee, V. Rabaud, K. Konolige and G. Bradsk, ORB: an efficient alternative to SIFT or SURF, in: Proceedings of the Computer Vision (ICCV), IEEE International Conference on, 2011, pp. 2564 –2571.

[56] J. Ruiz-del-Solar, M. Correa, R. Verschae, F. Bernuy, P. Loncomilla, M. Mascaró, R. Riquelme, F. Smith, Bender – a general-purpose social robot with human-robot interaction abilities, J. Hum. – Robot Interact. 1 (2) (2012) 54–75.

[57] J. Ruiz-del-Solar, P. Loncomilla, Robot head pose detection and Gaze direction determination using local invariant features, Adv. Robot. 23 (2009) (2009) 305–328.

[58] K. Saenko, S. Karayev., Y. Jia, A. Shyr, A. Janoch J. Long, M. Fritz and T. Darrell, Practical 3-D Object Detection Using Category and Instance-level Appearance Models, in: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), 2011 pp. 793–800.

[59] Van de Sande, K.E.A., Snoek, C.G.M.., Smeulders, A.W.M.: Fisher and VLAD with FLAIR. Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, 23-28 June 2014, pp. 2377–2384.

[60] C. Schmid, R. Mohr, Local Greyvalue invariants for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 19 (5) (1997) 530–535.

[61] V. Seib, S. Christ-Friedmann, S. Thierfelder and D. Paulus, Object class and instance recognition on rgb-d data, in: Proc. SPIE 9067, Sixth International Conference on Machine Vision (ICMV 2013), 90670J, December 24, 2013.

[62] Viktor, Seib Susanne, Thierfelder Dietrich Paulus. Object recognition tasks for Service Robots 8th Open German-Russian Workshop: Pattern Recognition and Image Understanding, 2011, pp. 258–261.

[63] J. Shi, and C. Tomasi, Good features to track, in: Proceedings of the 9th IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593 – 600.

[64] D. Shin, T. Tjahjadi, Clique descriptor of affine invariant regions for robust wide baseline image matching, Pattern Recognit. 43 (10) (2010) 3261–3272.

[65] J. Sivic, A. Zisserman, Video google: a text retrieval approach to object matching in Videos. ICCV '03 Proceedings of the Ninth IEEE International Conference on Computer Vision, Page 1470-1477, vol 2, 2003.

[66] S.M. Smith, J.M. Brady, SUSAN – a new approach to low level image processing, Int. J. Comput. Vis. 23 (1) (1997) 45–78, http://dx.doi.org/10.1023/A:1007963824710.

[67] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, M. VandeWeghe, HERB: A Home Exploring Robotic Butler, Autonomous. Robots 28 (1) (2010) 5–20.

[68] T. Tuytelaars, K. Mikolajczyk, Local invariant feature detectors: a Survey, Found. Trends Comput. Graph. Vis. 3 (3) (2008) 177–280.

[69] K.E.A. Van de Sande, T. Gevers, C.G.M. Snoek, Evaluating color descriptors for object and scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. 32 (9) (2010) 1582–1596.

[70] Z. Wang, B. Fan, F. Wu, Local intensity order pattern for feature description, in: IEEE ICCV, 2011.

[71] Q. Yan, Y. Xu, X. Yang, T. Nguyen, HEASK: robust homography estimation based on appearance similarity and keypoint correspondences, Pattern Recognit. 47 (1) (2014) 368–387.

[72] S. Zickler and V. Veloso, Detection and localization of multiple objects, Humanoid Robots, 6th IEEE-RAS International Conference on, 2006, pp. 20–25.

[73] OpenCV Official website: ⟨http://opencv.org/⟩.

[74] Siftgpu website: ⟨http://cs.unc.edu/∼ccwu/siftgpu/⟩.