1. Different types of objects: the approach should work with every type of object. Processes with the following challenging requirements:

- Video retrieval [1,2], or automatic robot navigation [3].
- Requires to detect and localize objects in the scene; they are crucial in scenes that are not well structured and constrained.
- Degrees of freedom. Object picking can be very complicated if the orientation of the next object to be picked up, according to the robot’s charge of driving the robot arms to the right 3D location and possibly coplanarly composed of robotic systems and sensors. The sensors are in charge of driving the robot arms to the right 3D location and possibly orientation of the next object to be picked up, according to the robot’s degrees of freedom. Object picking can be very complicated if the scene is not well structured and constrained.
- The automation of object picking by using cameras, however, requires to detect and localize objects in the scene; they are crucial tasks for several other computer vision applications, such as image/video retrieval [1,2], or automatic robot navigation [3].

This paper describes a new complete approach for pick-and-place processes with the following challenging requirements:

1. Different types of objects: the approach should work with every type of object of different dimension and complexity, with reflective surfaces or semi-transparent parts, such as in the case of pharmaceutical and cosmetic objects, often reflective or included in transparent flowpacks;
2. Random object disposal: most of the picking systems consider the case of well separated objects, well aligned on the belt and with a synchronized grasping of the objects. We would like to generalize the problem by relaxing these constraints. The ultimate goal is to work directly in bins (problem known as bin picking [4]), for saving time and/or for hygienic reasons, as shown in Fig. 1(b) and (c);
3. Multiple instances and distractors: in the case of pick-and-place applications the aim is not limited to count and classify the first (or best) instance, but to determine the locations, orientations and sizes of all (or most of) the duplicates/instances. Object duplicates can have different sizes, poses and orientations, and they can be seen from different viewpoints and under different illumination. Moreover, in real applications the system must also account for the presence of distractors, i.e. other types of objects, different from the target one (see, for instance, Fig. 1(d)), that should not be detected;
4. Heavily-occluded objects: as a consequence of requirements 1 and 2, objects can be severely occluded (see Fig. 1);
5. High working speed: the required working speed is very high; a fast detection technique should be adopted to work more than a hundred of objects per minute.

Machine vision often exploits a 3D CAD model of the object [5–7]. In particular, the active appearance models used for 3D face matching in [7] provide fast and accurate object matching. They may, however, result unsuitable for pick-and-place applications because of the illumination variations (e.g., the reflexes due to flowpacks), the severe occlusions and deformability of the objects.
The approach described in this paper is meant to tackle all these requirements by proposing a feature-based segmentation technique capable to detect multiple occluded objects. When objects are complex, reflective, low-textured and heavily occluded, very few distinctive feature points can be extracted from the image. Having few features to be matched with the original model, segmentation of multiple instances of the object is not straightforward. Having multiple overlapped instances, detection cannot be achieved with global features such as shape descriptors, texture or color histograms, while they are used in classification and retrieval after detection. Thus, we propose to use local features, based on single-point SIFT [8] feature detector, which has proven to be robust and sufficiently general.

Then, we define a general voting scheme to cluster the matched feature points among the different duplicates. This voting process assumes Euclidean transformation between the model and the current image. Although Euclidean transformation could be in general insufficient (w.r.t. to a projective one), it is a reasonable assumption in a pick-and-place setup (with downward-pointing cameras) and our extensive tests demonstrate its accuracy. To accomplish localization too, we define a small set of control points of the shape, i.e. points which characterize and delimit the object shape, such as, for instance, the four corners in the case of a rectangular-like shape. These points can be used as both delimiters of the object for the segmentation and grasping points, depending on the end effect or of the robot.

In order to improve the accuracy in the case of low-textured, reflective or semi-transparent objects (in which only few “strong” matches are found, also due to the occlusions), multiple models of the target object can be used. We use both multiple models of the same view of the object (to account for partial occlusions, reflexes, rotations of the object and noise in the acquisition process) and multiple models of different views of the target object (to account for the fact that, in case of random disposal, there is no guarantee of which face of the object will be visible).

It is worth emphasizing that, given its generality, this approach is applicable also to different contexts, with the only limiting assumption of a Euclidean transformation between the image model and the current image. Another important consideration is that the accuracy in the pixel-wise segmentation is not very important for our purposes. Indeed, the crucial aspects we would cope with are:

- to be able to accurately localize the grasping points;
- to detect and localize, one-by-one, all the duplicates of the target object: in fact, for a pick-and-place application it is not required to have a one-shot segmentation of all the duplicates since the robot can pick up one object at a time; conversely, it is required that at least one catchable object is found at each iteration and that all the duplicates are eventually found;
- to be very fast, in order to be competitive w.r.t. other approaches.

This paper is structured as follows. The next Section will present the related works, at the level of both applications and techniques. Section 3 will briefly describe the overall system, while Sections 4 and 5 will focus on the main contribution of this paper, i.e. the object detection algorithm, by presenting the proposed solution firstly using a single model for the target object (Section 4) and then extending the algorithm to account for multiple models (Section 5). Section 6 shows the accuracy and the efficiency of the proposed solution which fulfills all the requirements of the given pick-and-place application. Finally, Section 7 summarizes the contributions of the paper and draws the conclusions.

2. Related works

From the applications' point of view, the scientific literature in vision-based pick-and-place is very profuse, even though quite outdated. For instance, the pioneering work [9] exploits image processing techniques to determine the grasping points for picking up unknown
everyday objects. Basic techniques are exploited: thresholding to segment objects and moments to determine the location of the center of gravity and the orientation of the main inertial axis. In another work, Rahardja and Kosaka [4] propose the use of stereo vision to perform bin picking of industrial complex objects. Simple visual global features, such as region area, eccentricity, and gray scale mean value, are adopted for object recognition and pose estimation.

Unfortunately, these approaches adopt so simple image processing techniques that in complex scenes containing multiple objects, such as those reported in Fig. 1, cannot be applied. An alternative way instead is the detection-by-feature approach which searches for discriminative local features. This proposal is very old [10] and it is called local-feature-focus. This algorithm recognizes and locates partially visible 2D objects, by not performing the segmentation globally at pixel level, but on higher-level features, such as round holes and convex or concave 90° corners. The algorithm searches for a cluster of local features in a relative configuration which is characteristic of that specific object. One feature in the cluster is selected as the “focus” feature, i.e. the one with respect to which the other features are located. This approach accounts also for complex structure of features, by means of binary decision trees and feature indexed hypotheses. These methods exploited very specific features (such as round holes) that cannot be extended for whichever type of object.

The detection of complex objects from an image is undoubtedly a challenging task, as also demonstrated from the continuous presence of contest on this topic. For instance, in the “Solutions in Perception” challenge [11], the pose of rigid non-shiny, non-transparent textured and non-textured objects must be recovered, using, however, multiple sensors. This problem can be faced in different ways. Color-based segmentation has been widely used to detect and analyze objects on the basis of their appearance [12,13]. Unfortunately, the clutter of object appearance due to occlusions and distractors tends to make these approaches unreliable in real scenarios. An alternative approach is the use of edges to find the object boundaries. Often the edge information is strengthened with region-based information: examples are the work in [14] where the edge-based initial segmentation provides the starting point for a region growing algorithm, or the level set framework [15,7] or the various graph-based approaches [16]. This class of approaches is very fascinating and demonstrated excellent accuracy in several contexts. However, it still relies on region-based properties which are likely to be not tolerant to occlusions. Moreover, reflexes are very problematic for gradient-based techniques, typical of edge detectors. When the objects are complex but well modeled, 3D models can be exploited to disambiguate in the case of occlusions and to account for geometric transformations. For instance, Mian et al. [17] used range images of the target object to build a complete 3D model which is then efficiently compared with the current image by exploiting tensors and a hash table-based voting scheme. This approach proved to be very robust to occlusions but the average processing time per image is in the order of tens of seconds.

Given the limitations of the above approaches and our requirements presented in Section 1, a suitable approach is to use a point-based or feature-based solution. In this class of approaches single interesting points are matched between a model of the target object and the current image and these matches are exploited to detect and localize the objects. An interesting evaluation of different feature detectors and descriptors was proposed in [18] by Mikolajczyk and Schmid. In this evaluation, authors compare the performance of ten different descriptors.
computed for local interest region extracted by five different detectors and conclude that SIFT descriptor outperforms the others independently by the used detectors. To achieve this result, the descriptors are evaluated on real images with different geometric and photometric transformations such as rotation, scale change, viewpoint change, image blur, JPEG compression, and illumination.

Another interesting approach based on feature is the Dominant Orientation Templates (DOT) proposed in [19]. Similarly to HoG (Histograms of Gradients) approach [20], DOT computes local gradients, but differently from it, it relies on locally dominant orientations and made explicitly invariant to small deformations and translations. Despite the interesting results reported in [19], this approach has never been tested in case of occlusions, which may result in performance degradation.

Feature-based object duplicate detection can be described with three main phases. The first aims at defining and computing a proper similarity measure between the target object (or part of it) and the object duplicates in the image. Two state-of-the-art techniques have been proposed. The first is the Bag of Words (BoW) model, which is based on the histogram of local features [21]. However, since the BoW model is based on histograms, its main weakness relies in the lack of spatial information, which makes it unreliable in cluttered...
for localization purposes and not for detecting overlapped objects. In [3] the SIFT clustering allows the localization of the object, while our approach allows the precise detection of overlapped objects too. In this context, our approach could be viewed as an extension of the work in [3]. However, our approach is capable to extract more precisely the orientation of the object and which face of the object is visible, thanks to our multi-model, multi-face approach. Moreover, the approach in [3] does not consider the case in which the center of gravity is not the best (and the only) grasping point. All these issues are addressed by our approach and are crucial for a pick-and-place application.

Leordeanu and Hebert in [27] present a spectral method for finding consistent correspondences between two sets of features. The nice mathematical framework presented here permits to include also spatial relationships under the form of affinity between two correspondences. This is also our case but the graph-based approach proposed for finding the optimal solution is likely to be too onerous for our time constraints. Moreover, it has not been designed specifically to handle images with duplicates.

3. Picking and placing

The approach we are proposing is specifically oriented to a typical pick-and-place system. An example is the system sketched in Fig. 2. The objects are dropped in bulk on bins positioned on a belt. The belt is moved to bring the bins to the picking station where two cameras acquire a pair of synchronized stereo images. The second camera is used to estimate the distance from the cameras of the object to be picked up: this estimation, through a proper calibration of the stereo cameras, will allow the correct 3D localization of the object. Once the objects are detected (Fig. 3(a)), one of them is selected as the next one to be picked up (Fig. 3(b)). This Object Selection phase can be based on some models to optimize the robot trajectories.

The third phase (3D Object Localization) exploits camera calibration to obtain 3D world coordinates of the grasping points of the selected object. In particular, standard camera calibration is employed to transform the 2D image coordinates obtained in the first phase into 2D world coordinates, while the stereo cameras are used to estimate...
the third dimension and to transform it in world z coordinate. Finally, these coordinates are sent to the robot which picks up the selected object (Object Picking phase).

This process is iterated by acquiring a new image from a single camera (see Fig. 3(c)). In principle, the image segmented at the previous iteration could be used to obtain the next selected object (without performing object detection again — see dotted line in Fig. 3), but the picking of the object often shuffles the remaining objects.

4. Single model approach for object detection

Fig. 1 shows examples of cluttered images where the object detection and localization must be provided. The proposed solution should be invariant to the object in-the-plane rotations. If the object is mainly bi-dimensional and symmetric in both views, a single model of the object is sufficient, by considering only a main view. The object model (or template) can be captured with whichever orientation and under any illumination condition. This proposal will be extended to multiple models in Section 5.

Our proposal goes through the following two steps:

1. Feature extraction and matching: significant features are extracted from both the object model and the current image; given a proper similarity measure, features are matched between the model and the current image, and the best correspondences are retained;
2. Object localization: given the set of correspondences, it is possible to compute a registration transform between the model and the best location of the detected object in the current image.

The first step can be further divided in three sub-steps: the off-line extraction of features on the model, the on-line feature extraction on the current image and the matching between the model and the image. The features on the model can be extracted off-line and stored for iterated searches of the same model, while the features on the current image must be computed on-line.

Among the possible local features to be used for model-image matching, we selected the SIFT features and the 2NN (two nearest neighbors) heuristic proposed in [8]. SIFT has proved to be very robust to noise and invariant to scaling, rotation, translation and (at some extent) illumination changes. Moreover, SIFT computation can be onerous for real time applications, but a GP-GPU (General Purpose Graphical Processing Unit) version is available (called SiftGPU [28]) which is extremely fast.

Thus, we can use SIFT to obtain a set of keypoints $K = \{k_i = (x^M_i, y^M_i, \theta^M_i), i = 1, \ldots, n\}$ for the model $M$, where $x$ and $y$ are the 2D image coordinates, $D$ the 128-value SIFT descriptor and $\theta$ the main orientation computed by SIFT. Once the keypoints $k_i$ of the model are computed, during the on-line functioning, the system applies the same algorithm to the current image $I$, resulting in a set of keypoints $K = \{k_j = (x^I_j, y^I_j, \theta^I_j), j = 1, \ldots, R\}$. Given the two sets $K$ and $K$, the standard 2NN matching algorithm computes the Euclidean distance between $D^M$ and $D^I$ to find the image-to-model matches (or correspondences) $M = \{m_i = (m_{x_i}, m_{y_i}, m_{\theta_i}), i = 1, \ldots, m_n\}$, where each match $m_i$ contains the $(x,y)$ coordinates on the two reference systems and the main orientation on the current image: $m_{xy} = ((x^I_j, y^I_j, \theta^I_j), (x^M_i, y^M_i))$. To increase robustness, 2NN rejects the matches for those keypoints for which the ratio of the nearest neighbor distance to the second nearest neighbor distance is greater than 0.8. This discards many of the false matches arising from background clutter. Finally, to avoid the expensive computation of Euclidean distance, an approximate algorithm called the best-bin-first algorithm is used [29].

Given the set $M$, the simplest approach for computing the registration transform between $M$ and $I$ (point 2 above) is to estimate the planar homography using a least squares approach. For instance, the second column of Table 1 shows the resulting homographies obtained using SVD to solve the linear equations. The first column of Table 1 shows the obtained matches, where the model is in the top-left corner.

However, this approach is very sensitive to outliers in the set of correspondences. For instance, the second row of column “SVD” in Table 1 shows an example of incorrect homography obtained by least square method with SVD using all the matches: it is evident that the fragmentation of the matches on multiple instances generates several outliers in the estimation of the homography’s parameters. A well known method to deal with the outliers is provided by RANSAC [30] which finds a set of inlier correspondences which can be used for computing the transform as described above. The result of estimating the planar homography with RANSAC and least square estimation is shown in the third column of Table 1. Although the result is appreciable, this approach still presents some drawbacks. The first is that, in the case of multiple instances of the same object, the SIFT does not guarantee to find all the correspondences on the same instance. Even though RANSAC can, at a certain level, handle this situation by iteratively estimate the most consistent set of matches (as in the case of the second row of Table 1), it is not able to cope with a large number of outliers (in percentage) due to the presence of multiple instances, as shown in the last two rows.

A possible solution, described in the next section, is to cluster the matched features $M$ on $S$ subsets $M_t$, possibly containing only the
features belonging to a single object/instance \(O_i\). By running the RANSAC and least square on each subset \(M_i\), the resulting homographies are generally more accurate (see column “RANSAC clustered” in Table 1). However, this approach gives fewer points on which the homographies’ parameters are estimated, resulting, typically, in less accurate results, especially if these matches are fairly collinear (see first column, last row of Table 1) where the few matches on the middle object are collinear and the resulting homography is imprecise). Moreover, RANSAC’s result is unpredictable due to its random sampling procedure, which might be a problem in industrial applications.

4.1 Segmentation of multiple instances

The proposed approach for object detection is reported in Fig. 4. Given the drawbacks of the approaches mentioned above, we have developed a 2-steps method based on a voting scheme, which is employed to estimate the locations and the orientations of duplicate objects. The first of the 2 steps (clustering of the matched features — see Fig. 4) allows the estimation of the object center’s position and performs the matches’ clustering. In the second step, the positions of a fixed number of control points are computed for each cluster in order to define the delimiting shape and/or the grasping points of the object.

The clustering of matches to obtain the subsets \(M_i\) can be performed by considering, similarly to [3], the relative position of each match with respect to the center of the object. In other words, given the object model and having selected manually its center \(P_0\), the vector distance between each keypoint \(\kappa_i\) of the model and \(P_0\) is computed and stored. Then, given the matched features \(m_j\) in \(M\) and assuming a pure roto-translational (i.e., Euclidean) transform, these vectors are reported in the current image by exploiting the main orientation \(\theta_i\) of each keypoint provided by SIFT.

This corresponds to a voting scheme (see Fig. 4), where features belonging to the same object will vote for similar center’s positions. However, given the approximations introduced by the image noise, the features localization and the Euclidean transform assumption, the center estimation is not accurate. To deal with this, the center’s estimates are clustered by using the Mean Shift [31] and only the centers with a minimum number of contributing matches are considered as correct. This minimum number depends on the object nature: increasing the object’s complexity, the minimum number is decreased. With the mean shift clustering, the set \(M\) of correspondences is partitioned in subsets \(M_i\) for each of the duplicate object \(O_i\) found in the current image.

In comparison to the RANSAC clustered approach described at the end of the previous section, our proposal is to further relax the problem’s conditions by assuming a complete Euclidean transform for all the pixels of the model (not only the center). This means that we only consider in-the-plane rotations and translations, not permitting the model to scale (reasonable condition if we assume that the objects are more or less at the same distance from the camera) or to rotate (much) out of the image plane. This is enough because a precise segmentation is not required since our goals are to find the grasping points and to evaluate occlusions (in order to avoid the picking of covered objects): thus, in the case of tilted objects with out-of-plane rotations, the system will still detect them, but with an imprecise segmentation, often sufficient, however, for object picking.

Once the matches have been partitioned, the control point localization (see Fig. 4) procedure is repeated for \(L\) relevant points of the object (which may be delimiting and/or grasping points):

1. during the definition of the object’s model (leftmost upper part of the scheme in Fig. 4), the user can select \(L\) control points \(P = \{P_0, ..., P_{L-1}\}\), where \(P_0\) is the center of the object and the other points represent both other grasping points and points delimiting the objects, such as extrema points of the oriented bounding box;

![Exemplar images of the objects used in our tests.](image-url)
2. for each \( m_\theta \in M_i \), the estimate for each of the \( L \) control points is computed; let us define as \( P_j(i, y) \) the estimate obtained from match \( m_\theta \) of instance \( O_i \) for the control point \( P_j \) with \( j = 1, \ldots, L - 1 \);
3. \( L - 1 \) mean shift algorithms are issued to find the best estimate \( P_j(i, *) \) for \( P_j \) in \( O_i \); in this case, the mean shift is not employed for clustering (thus named “Mean Point Estimation” in Fig. 4); however, computational complexity of mean shift with some tens of points, as in most of our cases, is negligible;
4. the \( L - 1 \) estimates \( P_j(i, *) \) are used to obtain the segmentation of \( O_i \).

The fifth column of Table 1 shows how the problem inherent to homography can be solved with our approach. It is worth emphasizing that, supposing the clustering to be the same, “RANSAC clustered” and our approach differ only in the type of transformation being estimated. Since Euclidean transformation is enough for our application, the latter is more appropriate since the transformation can be correctly estimated also with few correct matches.

Additionally, Fig. 5 shows two examples of the result achieved with this procedure. The large green circles represent the estimates \( P_0(i, *) \) of

---

**Table 3**

Experimental results for single model approach.

<table>
<thead>
<tr>
<th></th>
<th>Object-level</th>
<th>Pixel-level</th>
<th>Center dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
</tr>
<tr>
<td><strong>Juice</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All RS</td>
<td>100.00%</td>
<td>25.00%</td>
<td>22.95%</td>
</tr>
<tr>
<td>Clus RS</td>
<td>91.67%</td>
<td>82.50%</td>
<td>77.43%</td>
</tr>
<tr>
<td>Ours</td>
<td>97.37%</td>
<td>92.50%</td>
<td>88.55%</td>
</tr>
<tr>
<td><strong>Nutella</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All RS</td>
<td>100.00%</td>
<td>15.38%</td>
<td>13.23%</td>
</tr>
<tr>
<td>Clus RS</td>
<td>66.67%</td>
<td>33.85%</td>
<td>38.96%</td>
</tr>
<tr>
<td>Ours</td>
<td>97.84%</td>
<td>86.78%</td>
<td>82.87%</td>
</tr>
<tr>
<td><strong>Flyer</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All RS</td>
<td>90.00%</td>
<td>16.36%</td>
<td>17.46%</td>
</tr>
<tr>
<td>Clus RS</td>
<td>74.00%</td>
<td>64.91%</td>
<td>71.31%</td>
</tr>
<tr>
<td>Ours</td>
<td>96.15%</td>
<td>90.91%</td>
<td>86.35%</td>
</tr>
</tbody>
</table>

---

Fig. 10. Experimental results on three types of objects (first row: “Mascara 2”, second: “Coffee”, third: “Nutella”) with increasing number of models.
1. the SIFT algorithm relies on the gradient and on the textured areas of the object with high contrast; the object's center and the values close to them are the number of matches assigned to that object through the clustering. The small circles in blue, magenta, yellow and green represent the estimates \(P_j(i,y)\) of the other four control points (the extrema of the bounding box, in these examples). The estimates are mixed up and fairly distributed, but the mean shift is nonetheless capable to act correctly. The white lines connect the estimates \(P_j(i,y)\) of the control points and are the boundaries of the final segmentation. It is evident that this approach is able to segment also very occluded objects, as shown in Fig. 5(b).

2. our tests have shown that SIFT is not robust enough to local brightness changes, especially in the case of reflective or semi-transparent objects, resulting in too few and unreliable matches (see Fig. 6(b));

3. our approach becomes less reliable when the model has few matches, due to the few votes in determining the control points of the object.

To increase the number of the available matches for the object detection, the current image can be matched with different models of the same view of the object. All the models are taken under free environmental illumination and using different object orientations in order to be robust to the reflexes created by a possible transparent container. In addition, objects may have different faces/views and, given our requirement of random object disposal, they may be visible in any of them. Consequently, different models for each view must be acquired. The number of images to be collected depends on the type of object: low-textured objects will require more images (on the order of ten), while well-textured objects will need only one or two images. In general, about a hundred of features must be collected from all these model images.

In conclusion, the complete set of models \(M\) for a given object is composed of \(N\) images, where \(N = \sum_{i=1}^{FN} N_i\), with \(FN\) the total number of faces and \(N_i\) the number of models used for the face \(i\). We will indicate the model \(M_i = M\) as the \(i\)-th model for the \(i\)-th face, with \(i = 1, \ldots, FN\) and \(j = 1, \ldots, N_i\). The snapshots in Fig. 7 show four models (two for each face) for the "Coffee" object.

Each considered model generates a set of keypoints \(K_j^i\) (Fig. 4). The keypoint sets of a given face \(F\) are grouped together \(K^F = \{K_1^F, \ldots, K_{N_i}^F\}\) and compared with the set \(K\) of the keypoints extracted from the current image to form the set of matches for the face \(F\) \(M^F\), whose cardinality is typically larger than that of \(M\) defined before. For instance, Fig. 6(c) shows the matches obtained with 5 models on the same image of Fig. 6(b). The set \(M^F\) is then partitioned using the clustering procedure described in Section 4.1 to obtain the sets \(M_j^F\) for each object \(O_i\) (Fig. 4). These sets bring to the segmentation \(\mathcal{E}\) of the current image. This procedure is then repeated for each face of the object and the resulting segmentations are merged together. An example of the segmentation obtained with the multi-model extension in a complex scene is shown in Fig. 8, with reference to the models reported in Fig. 7.

6. Experimental results

6.1. Experimental setup

In order to validate our approach, experiments over very different types of objects are performed. A summary of the objects used in these tests is reported in Table 2. For each object, a snapshot and a list of peculiarities are reported. Some peculiarities may be of help (i.e., they are “positive” in table) for the detection (such as the texture or the size), some other may hinder (denoted by “negative”) it (such as the reflectivity or the transparency). The grade (from “Very Low” to “Very High”) reported for each peculiarity represents how much it applies to that type of objects. The column “Difficulty” is a subjective evaluation of how difficult is to detect the object.

The accuracy of our approach can be measured by means of three different measures: the precision/recall at object-level, the precision/recall at pixel-level and the accuracy of the center location. The first measure accounts for how many correct objects are detected (where correct segmentations are evaluated by the operator), while the second considers the pixel-by-pixel segmentation. In this case, the precision and the recall are computed only for the detected objects (either correctly or not), and thus they tend to be very high. Finally, the last measure is more...
application-oriented, thinking to a pick-and-place application where the
accuracy in determining the grasping point (e.g., the center) is crucial. All
these measures are computed with respect to a manually determined
ground truth.

As mentioned earlier, the most important aspects for our applica-
tion are: the accuracy in determining the grasping point, the fact that
at least one catchable object is detected at each iteration (in order to
“feed” the robot) and the picking of all the objects after a certain
number of iterations. These last two aspects will be evaluated in a
specific test. Last but not least important for our purposes, we also
aim at evaluating the efficiency of our proposal by measuring the
computational time also in dependence of the number of models
used.

The experiments have been divided in five tests. The first one aims at
demonstrating the accuracy of the choice for single model approach and
the voting scheme for object duplicate detection (Section 6.2). The
second test (Section 6.3) has the objective to show how the perfor-
mance increases by using multiple models (Fig. 9). The third test
(Section 6.4) shows the capability of the system to always select a
catchable object and to pick up all the objects in complex scenes, con-
taining also several distractors. The fourth test will evaluate the accura-
cy with each type of object in several tests using a fixed and optimal

![Fig. 12. Examples of segmentation for increasing number of models.](image)

![Fig. 13. Sequence of iterative picking for object “Mascara 1.”](image)
number of models (Section 6.5). Finally, the efficiency of the adopted solution will be also measured (Section 6.6).

6.2. Test for single model approach

Results for the first test are summarized in Table 3. We compared our proposal with the use of homography-based segmentation by either RANSAC on all the matches (all RS in Table 3) or RANSAC on clustered matches (clus RS), as described in Section 4. Each of the three objects reported in Table 3 has been tested with 10 images with an increasing number of objects (from 1 to 5 for the juice, from 1 to 9 for the nutella and the flyers, which are smaller). The degree of overlapping is increasing as a consequence. Finally, as stated in the introduction, light conditions are not controlled, using normal illumination and an open machinery.

Since the RANSAC applied on all the matches finds a single instance of the object, the precision at object-level is close to 100%, but the recall at object-level is very low. Instead, the RANSAC applied to clustered matches shows a poor accuracy in identifying the center. This is due to the fact that this algorithm finds more objects than the version on all the matches (precision/recall both at object- and pixel-level are higher), but the resulting homographies are less accurate since they are estimated by fewer matches.

Our approach outperforms the other two in all the cases and for every measure, with average precision and recall at object-level of 97.84% and 86.78%, at pixel-level of 82.87% and 83.13%, and an average center’s distance of 3.86 pixels. These tests have been performed on fairly simple objects with highly textured faces, such as juices, nutella and flyers. However, the single model approach is likely to be much more imprecise when applied to low-textured objects.

6.3. Test for increasing number of models

In order to prove the enhancement of object detection using multiple models for low-textured objects, we measured the performance for the "Mascara 2" object (which is the most problematic in terms of texture) with an increasing number of models. These tests have been performed with no distractors (to ease the manual ground-truthing). The threshold on the minimum number of matches for the segmentation has been fixed empirically to 3 for all the experiments. Each test has been run on five different scenes with a different number of objects and complexity (in terms of overlap and number of objects). The results at object level are reported in the graphs of the first row of Fig. 10. Using a single model no mascara is found in scenes 1, 4 and 5 (this situation is indicated by an arrow and the text “No correct object found”). By increasing the number of models (up to 7) the system...
finds more objects, i.e. the recall at object-level increases (Fig. 10(b)), and the objects found are always correct, i.e. the precision at object-level is very often equal to 1 (Fig. 10(a)). The precision and recall at pixel-level are not reported since they are almost independent on the number of models.

In addition, we want to show the advantage in terms of accuracy obtained by using models of different faces. Second and third rows of Fig. 10 report the graphs for “Coffee” and “Nutella” objects, respectively. In addition to the considerations made for the “Mascara 2” object, it is interesting to note that there is a strong improvement in the recall at object-level when introducing the models also of the second face (Fig. 10(d) and (f)): in particular, in the case of “Nutella” the recall with models only of the first face is never higher than 60%, while it quickly reaches 100% when the models of the second face are added, at least in the simplest scenes.

We also evaluate the influence of the number of models in the computation of the center (Fig. 11). In the case of “Mascara 2” the center distance is rather limited (no more than 6 pixels) which is important since it defines the accuracy in determining the grasping point. The same applies for the other two objects and it even tends to decrease when the number of models increases in the case of “Coffee” (Fig. 11(b)). The optimum number of models cannot be found analytically but must be decided empirically, depending on the complexity of the object.

In order to also provide a visual proof of the accuracy of our method, Fig. 12 shows some examples of segmentation (for scene 5): the column
still represents the object type, while the rows indicate the number of models (top row: minimum number; middle row: average number; bottom row: maximum number).

6.4. Evaluation of the iterative picking procedure

As stated above, from the application’s point of view, one of the most important characteristics of the system must be to provide to the robot a catchable object at every step, until the picking of all the target objects present in the scene. We have simulated this behavior by iteratively picking the selected object by hand and segmenting the new image. The sequence of segmentations achieved are reported in Figs. 13 and 14 for “Mascara 1” and “Coffee” objects, respectively. Moreover, in the case of “Coffee” several distractors have been included. Please note that the colors used for the segmentations are randomly chosen. In addition, we also performed a quantitative analysis of the iterative picking procedure by computing the percentage of times in which at least one object results to be catchable (Fig. 15).

6.5. Overall evaluation

Section 6.3 reported the experimental results using an increasing number of models. This section, instead, keeps the number of models fixed (to the maximum one) while increasing the number of objects in the scene and its complexity (including distractors). The results for all the nine types of objects reported in Table 2 are shown in Figs. 16–20 with the maximum number of models used listed below the graphs.

The precision at object-level is always close to 100%, except in the case of “Dropper” (Fig. 19(b)) because of its complexity and the
presence of flowpacks which create reflexes. The recall at object-level typically decreases with the increase of the number of objects (and of complexity), except in the case of “Flyer” (Fig. 16(b)) because they are very well textured. Moreover, this measure is lower for difficult objects (see Fig. 20). This is true also for the precision and recall at pixel-level, which are in general quite high. Fig. 21 shows some examples of segmentation in complex cases. Fig. 21(a) and (b) refer to the case in which the centers of two overlapped objects are almost coincident: this situation can result in the matches on the two objects to vote for the same location. However, thanks to the keypoint orientation our approach is capable to segment correctly the topmost instance of the object. This is possible since, once the center has been determined, the control points are voted by each match and more votes are casted by the object closest to the camera.

6.6. Efficiency evaluation

Besides the accuracy, a very important requirement for our application is its efficiency. The whole system has been developed in standard C/C++ making use of OpenCV library for certain functions. Initially, the system has been implemented on a standard PC with a Intel Core2 T7200 CPU and 2 GB of RAM. The performance has been tested on all the nine types of objects by measuring the time required for the main steps of the algorithm. As demonstrated by the results reported in Fig. 22(a) the step requiring most of the computational time is the computation of SIFT descriptors on the current image together with the matching with the keypoints of the models. This step is influenced by the number of keypoints found: less textured objects such as “Syringe” or “Mascara” require less time, while well textured objects such as “Flyer” require much time. The graph in Fig. 22(a) also reports a line indicating the time to be reached for obtaining 120 pieces per minute, which has been our target time. It is evident that this PC-based solution is not enough to reach that achievement.

Thus, we have implemented a new version on a PC with Intel Core2 T6600 CPU equipped with a GPU Nvidia 9800 GTX+ on which a specialized version of the SIFT algorithm [28] has been used. By exploiting the parallel nature of the GPU (which is composed of 128 processors) the computation and matching of SIFT keypoints is greatly accelerated, as shown in Fig. 22(b), where the target time of 0.5 s for piece has been reached for all the objects. It is interesting to note that the “Flyer” objects require a significantly higher time for both SIFT (as it was also in the PC version) and segmentation: this latter is due to the fact that many matches vote for each center and mean shift is run on thousands of points instead of tens (as in the other cases). In fact, mean shift complexity is $O(n^2)$, with $n$ number of points/samples. Computational time is not influenced by the number of models used, unless more models produce a significantly higher number of models-image matches.

7. Conclusions

This paper describes the entire procedure conceived to detect and localize whatever object, disposed at random or in bins. This procedure
has been implemented as part of a real-working complete system for pick-and-place applications. The system has been fully tested, giving excellent results in terms of flexibility (it has been tested with several types of objects, much more than those presented here), accuracy (especially in terms of capability to always find a catchable object and in precision in determining the position of the grasping point(s)), and efficiency (thanks to a GPU-based version which does not significantly increase the costs). As a further result, this system has been patented in Italy and now extended to European and USA patenting.

As future directions, we are evaluating different detection strategies for texture-less objects (e.g., based on 2D or 3D model matching), as well as studying algorithms to detect whether an object is occluded (especially in its picking point) or not and implementing strategies for ordering the detected (and pickable) objects to minimize the traveling time of the robot.

Acknowledgments

The authors wish to thank Marchesini Group SpA, Pianoro (BO), Italy, for the financial support for this project. The work presented here has been patented by Marchesini Group SpA in Italy and Europe. US patent pending.

References