Using spatial constraints for fast set-up of precise pose estimation in an industrial setting

Frederik Hagelskjær, Thiusius Rajeeth Savarimuthu, Norbert Krüger and Anders Glent Buch

Abstract—This paper presents a method for high precision visual pose estimation along with a simple setup procedure. Robotics for industrial solutions is a rapidly growing field and these robots require very precise position information to perform manipulations. This is usually accomplished using e.g. fixtures or feeders, both expensive hardware solutions. To enable fast changes in production, more flexible solutions are required, one possibility being visual pose estimation. Although many current pose estimation algorithms show increased performance in terms of recognition rates on public datasets, they do not focus on actual applications, neither in setup complexity nor high accuracy during object localization. In contrast, our method focuses on solving a number of specific pose estimation problems in a seamless manner with a simple setup procedure. Our method relies on a number of workcell constraints and employs a novel method for automatically finding stable object poses. In addition, we use an active rendering method for refining the estimated object poses, giving a very fine localization, suitable for robotic manipulation. Experiments with current state-of-the-art 2D algorithms and our method show an average improvement from 9 mm to 0.95 mm uncertainty. The method was also used by the winning team at the 2018 World Robot Summit Assembly Challenge.

I. INTRODUCTION

Pose estimation is the task of determining a 6-DOF (Degree of Freedom) pose of an object in a scene. In the most general form the scene is completely unknown and the pose of the object is fully unconstrained. Several strong computer vision algorithms have been developed to handle this problem, utilizing 3D or 2D data, either with classical methods ([1], [2], [3]), deep learning [4], or combining classical methods with data-driven approaches [5]. Although they do not perform perfectly, high recognition rates have been achieved [6].

In spite of this impressive performance, a problem persists in the accuracy of the found poses. For robotic manipulation, e.g. in industrial cases, high precision is necessary. In some of the examples mentioned above, the accuracy threshold for correct detections in the benchmarks is 5 to 2 centimeters and 15° to 5° angular error [7], [8]. Robotic operations (e.g., assembly, screwing, polishing, welding) have much lower tolerances. In robotics, several methods have been developed to compensate for lack of precision. Force feedback [9] and optimization of grippers [10] can compensate for deviations, but only of a much smaller magnitude compared with the error in present computer vision systems.

Another important aspect in pose estimation is the required setup time. In a study on the usage of pose estimation, the median time to setup a practical vision system was found to be 1–2 weeks [11]. A cost that heavily restricts the possible industrial use-cases. In this work, a method for precise pose estimation is presented. The system is built on the idea of rendering the object from an expected pose. The expected pose is constructed using workspace knowledge (the pose of the table) and using a constrained set of stable object poses. The table pose is a surface which the object is placed upon, see Fig. 1. The stable pose is a transform fixing the object to the surface. The object can move on the surface and rotate about the vertical axis, giving 3DOFs for the movement of the object. To easily acquire the required workcell-specific information for our system, a setup procedure is presented:

- The table plane is obtained, e.g. by placing a known chessboard pattern and thereby calculating the pose.
- A list of stable poses of the CAD model is presented to the user who selects the expected orientation.

These restrictions guarantee a low setup time and precise pose estimates for our method. In addition, the restriction that objects appear on a planar surface is common in many domains where robots can be introduced, e.g. in industry and in domestic environments.

In summary we have created a pose estimation system intended for very high precision systems. The first contri-
bution is a method for automatic selection of stable poses. Stable poses combined with a known table plane, is used to project a 3D object onto a virtual table for generation of templates used with an existing 2D object detector, Line2D [12]. The second contribution is pose refinement by creating and matching, online rendered local templates in the vicinity of the estimated pose, thereby accounting for perspective errors. The system used for experiments is shown in Fig. 2.

The remaining article is structured as follows: Section II outlines state of the art pose estimation methods and current datasets. Our method is presented in Section III. Section IV contains our experimental evaluation and results. Finally, we outline the conclusion of our work and the further perspectives of this approach in Section V.

II. STATE OF THE ART

There has been a significant development in 6-DOF pose estimation in recent years, and multiple approaches have been developed. Beginning with the rise of SIFT, a realm of different 2D feature-descriptors has emerged. Alongside a parallel development of 3D features has happened, significantly accelerated by the advent of the Kinect Sensor. Finally, the current rise in deep learning algorithms has resulted in new pose estimation algorithms. In between, there has also been a development of stronger template-based matching algorithms, which are also described. In the following section, an outline of these methods is presented along with their limitations concerning industrial pose estimation.

A. 2D Features

2D feature matching is based on the description of local patches in the image. These features are expected to remain invariant under different types of noise and transformations. Having found good matches, a 6-DOF pose of the object can then be found [13]. Many different feature descriptors have been developed to describe patches. SIFT [1] uses a histogram of local gradients to decompose the patch, whereas Gabor Jets [14] and Binary Tests [15] have also been employed. When strong features are present these methods perform well, but a significant hindrance exists in industry; many objects usually do not possess texture. Many industrial parts are uniformly colored or consist of shiny metallic surfaces with very low contrast. These limitations severely hinder the use of 2D feature descriptors in industry.

B. 3D Features

To overcome a lack of visible texture for 2D features, it is possible to describe features in 3D space instead. Using, e.g. Lidar or stereo vision a depth image is captured, this depth image can then be represented as a set of 3D points. Local areas are then extracted and a feature computed. As in 2D, many different feature descriptors exist. The Spin image [2] is a classical descriptor using the normal vectors and distances of close by points to describe each patch. The normal vector is used to rotate the patch and make the feature invariant of rotation, a method often used by 3D descriptors. Many other descriptors have been employed with even greater success [16]. An example is [17], here a histogram is created by the relationship of a point and all nearby points, described by their orientation towards the normal vector.

Both 2D and 3D feature matching approaches utilize a high number of parameters. To obtain good results, a correct parameter configuration is essential [5]. This configuration can be obtained through optimization, but an adjustment is necessary for all new objects. When features have been computed, they are matched between object and scene to create a set of good matches. These matches are then used to compute a 6-DOF pose of the object. Fine tuning of the position is then performed using e.g. Iterative Closest Point (ICP) [18]. ICP is done by finding the distance between the nearest points from the object and points in the scene. A transform is then performed to decrease the distance. This continues until convergence or when a set number of iterations is reached.

Although such method works well for surfaces where a point cloud can be obtained, there are several difficulties. If the object surface is specular, instead of diffuse, good depth data cannot be obtained. Additionally, if the object is very small or does not exhibit 3D features, the 3D methods will not work.

C. Deep Learning

In recent years, huge developments have occurred in deep learning. Here we will focus on papers that perform 6-DOF pose estimation. Deep learning approaches use synthetic training images to train convolution neural networks to detect the objects. As deep learning methods are 2D based with an RGB-image as input, the output will be pixel positions. The
Fig. 3. Example of estimation of stable pose. From left to right, original CAD model, convex hull, selected plane with center of mass projected, Monte Carlo sampling of center of mass.

The final result can then be obtained either by projecting corners with PnP [19] to or by using ICP [4] to find the position. These methods have shown very promising performance, but are dependent on an appearance based model with textures and colors. This is generally not possible for industrial cases and no performance have been shown for uniformly colored objects.

D. Template Matching

The restrictions described for the 2D features, 3D features and deep learning methods are maybe best illustrated by the nature of the benchmarking datasets. All objects in both [8], [7], [12] have diffuse surfaces and color is represented in the models. This could be a plausible representation of a service robotics setup, but it is far from a realistic industrial setup.

To deal with objects that do not exhibit good features, either because of low contrast, specular surfaces or object structure, edge-based matching is possible. Edge matching is based on a full template of the object being matched in the image. Thus edges in the image are calculated and matched with the silhouette of the object.

The 3D refinement method ICP [18] also exist for 2D images here called 2D-ICP [20]. Edges are detected from gradients in the 2D image. From these edges, a chamfer image created with values lowering as the distance to edges increases. The chamfer image is used to decrease the effect of noise. Edges of the 3D object model are then projected into the chamfer image and an error is calculated. By least-square minimization of the error, the position of the object is refined to the image. This method requires a good initial estimate and cannot perform a full pose estimation.

A well-known algorithm for full edge-based matching is LINEMOD [12]. Although also utilizing depth data, a part of the algorithm is a gradient-based 2D template matching called Line2D. Line2D is the matching algorithm that we employ in this paper for coarse pose estimation. LINEMOD then employs several extra checks, e.g. color and depth verification, to sort the matches. We have chosen to exclude these extra steps as such information is usually not available in industrial cases.

The method presented in [21], found in the Halcon framework1, is another popular approach for detecting objects in 2D data. Initially, the search space of the algorithm is restricted by selecting a range of camera positions. A search of the object templates is performed in a hierarchical fashion. Similar to [20] a least-squares refinement is performed to adjust the object position. We refer to this method as Halcon2D.

Although these methods show good results, a limitation observed for 2D matching approaches is illustrated by the following quote from [3, page 5]:

"...we noticed that although many results of the SD2[Halcon2D] seemed to be correct when projecting them into the images ...because of the large focal lengths, a small error in the estimated object scale in the image or a small error in the size of the CAD model result in large errors in the z coordinate."

This was a problem that we also noticed during our participation in the World Robot Summit Assembly Challenge. Our approach to overcome this limitation is to use knowledge of a known table plane and a canonical stable pose. Using stable poses for pose optimization has previously shown advantageous for ICP [22], although in that paper an optimal placement of the camera in the workcell was the main objective.

III. METHOD

In the following section the method presented in this article will be elaborated. First we will describe the method to compute stable poses. Secondly, the formal aspects of our method will be outlined. Finally, the template generation and the method for refinement using online template rendering is described.

A. Computing stable poses

The use of a canonical pose is fundamental to our approach. To simplify the creation of new detections a method to propose and sort stable poses is presented. Our method resembles the methods presented in [23] and [24], but analytically has a more simplistic approach. In the following, we assume that a 3D mesh model is given, e.g. from a CAD drawing.

 Firstly, a convex hull is calculated for the model and the center of mass $m_e$ is found according to [25]. As the object is to be placed on a planar surface, every triangle in the convex hull is a possible stable pose. All neighbouring triangles sharing the same normal vector are combined, creating a list of triangles $T_i = \{t_1 \ldots t_n\}$ for all possible stable poses. To test whether the center of mass is inside a triangle we project into the coordinates of the first vertex by subtracting it from the remaining two vertices and $m_e$ giving the new points; $v_0, v_1$ and $v_2$. We can then calculate the following by transforming into Barycentric coordinates.

\[
\begin{align*}
 u &= (v_1 \cdot v_1)(v_0 \cdot v_2) - (v_0 \cdot v_1)(v_1 \cdot v_2) \\
 v &= (v_0 \cdot v_0)(v_1 \cdot v_1) - (v_0 \cdot v_1)(v_0 \cdot v_1) \\
 w &= (v_0 \cdot v_0)(v_1 \cdot v_1) - (v_0 \cdot v_1)(v_0 \cdot v_1)
\end{align*}
\]  

(1)

(2)

Using Eq. (1) and Eq. (2) a test of whether $m_e$ is inside the triangle is performed.

\[
in(t,m_e) = \begin{cases} 1, & (u > 0) \text{ and } (v > 0) \text{ and } (u+v < 1) \\ 0, & \text{otherwise} \end{cases}
\]

(3)

This test is then performed for all triangles $t_1 \ldots t_n$ corresponding to the transform $T$, if $m_e$ is inside any of the triangles a score

\[
inAny(T,m_e) = \sum_{i=1}^{n} in(t_i,m_e)
\]

(4)

For each transform $T_i$ a test is performed for all triangles $t_1 \ldots t_n$. The center of mass is projected to a Barycentric representation and is assumed statically stable if it lies within any of the triangles [24].

Even though a pose is found to be stable under static conditions the behaviour dynamic behaviour could be completely different, i.e. how much force is necessary to topple the object. In [24] the dynamic stability is tested by projecting a $20^\circ$ cone onto the stable surface and measuring the area inside. We instead choose a stochastic approach to determine the stability, see Fig. 3. First the distance from the center of mass to the surface is found and used as standard deviation for the Gaussian noise, $\sigma$. Using this distance makes the method independent of object size and use inherent stability in the distance to the mass center. By adding Gaussian noise to the center of mass, 1000 poses are sampled. Each mass center is then projected and the number of inliers is counted, resulting in a normalized stability score. The scores for all transforms in $T_i = t_1 \ldots t_n$ are sorted and a list of stable poses is presented to the user.

\[
stability(T,m_e) = \frac{\sum_{i=1}^{1000} inAny(T,N(m_e,\sigma))}{1000}
\]

(5)

To obtain the table plane multiple methods can be applied. Examples of such are robot to table calibration with existing camera calibration, and plane detection if a 3D camera is available. For our method a known chessboard pattern is placed on the table, the corners are detected and a transform matrix is calculated using [13].

B. Formal aspects

In the following a number of formal aspects of our method are explained.

1) Position on plane: Firstly we give method to determine the 3DOF position on a plane, from a pixel position given by $x$ and $y$. The camera parameters, focal lengths $f_x$ and $f_y$ and image centers $(c_x,c_y)$, are used to create a vector pointing out from the camera, $l$.

\[
l = \begin{pmatrix} (x-c_x) \cdot f_x \\ (y-c_y) \cdot f_y \\ 1 \end{pmatrix}
\]

(1)

The vector $l$ is combined with the table plane normal vector, $n$, and a point on the plane, $t$. This gives $P$ the final 3D point which corresponds to the pixel coordinate.

\[
P = l \cdot t \cdot n
\]

(2)

2) Calculating pose on table: Using the method described in Sec. III-A a canonical pose has been obtained, described as $plane^T_{obj}$. We have also obtained the transform from camera to table plane, $cam^T_{plane}$ via calibration. Using this a position on the table can be found:

\[
cam^T_{obj} = cam^T_{plane} \cdot plane^T_{obj}
\]

(3)

This will only render a single position of the object. But the object can move about on the surface, i.e. X and Y on the plane. The object is also able to rotate about normal vector of the surface while maintaining the stable pose, giving 3DOF, shown in Fig. 1. To represent the rotation of the object an extra term is added to Eq. (3), namely the rotation about the normal vector, $R_z(\theta)$. To move in position one adds the translation to the position term of $cam^T_{plane}$, resulting in the full transform as seen in Eq. (4).

\[
cam^T_{obj} = cam^T_{plane} \begin{bmatrix} R_z(\theta) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} plane^T_{obj}
\]

(4)

C. Template rendering and matching

To generate the initial templates, the middle position of the image is reconstructed to the 3D table plane using Eq. (2). This 3D pose is then inserted as the translation into $cam^T_{plane}$. By rotating $360^\circ$ about the normal vector templates representing all rotations are generated, see Fig. 4. By subtracting the background from the image the silhouette of the object is found and used as template.

Fig. 4. Three instances of templates of object 05 from [8] generated on the table with varying rotation.
The templates are then used as input for the Line2D algorithm [12] described in Section II. The detection found with the highest score is then used. When generating the templates the pixel offset from template corner to the table position was stored. Thus when a 2D coordinate is found this offset is added, giving coordinates \((match_x, match_y)\). These are input to Eq. (1), the output of which is input into Eq. (2) along with the rotation corresponding to the template giving a 6-DOF pose. We denote this approach "CP+Line2D", i.e. Template generation with Canonical Pose (CP) and Line2D detection.

This 6-DOF pose is affected by perspective differences between the position the template was generated in and the actual object pose. To eliminate this effect a second template is generated at found position, see Fig. 5. A rotation 20° range is added to the rotation to compensate for rotation errors from perspective. This is used for a second detection Line2D giving an increased position estimate. This method is denoted as "CP+Line2D+Cor", as a correction is added.

To further reduce the pose uncertainty a second refinement is performed. The templates are generated as above, but Line2D is not used for matching the local template. Instead, the Canny detector is employed to find edges. A simple template matching is then performed using the silhouette of the generated object template. As the pose is believed to be at the object, a less robust, but more precise matching can be performed. The pixel position is projected to 3D and a final 6D pose is found. This final method is denoted as "Full". All three methods are compared in the experiments in the following section.

IV. EVALUATION

To evaluate the performance of our method we have compared the methods in real world scenes. Objects with a known CAD model is placed in the scene and a detection have been performed, both for our own method and current methods. We have tested on two datasets, one we created to specifically test the high precision performance and a benchmarking which validates our approach. The setups are described in the following section.

A. Performance with very high precision requirements

To compare the high precision repeatability of our pose estimation approach with other well-known methods we created a dataset. The dataset consists of four scenes composed of two different objects and two different camera distances. The objects are placed in a fixture which can be fixed to the table by equidistantly placed threaded holes. By moving the fixture between threads multiple images are obtained while the distance between each pose is well known, the methodology is shown in Fig. 6.

The camera is a Basler-acA1920-48gc camera with an Edmund Optics 8.5mm C Series, the lens is calibrated with a 39° field of view.

For each scene, sixteen images have been created with the fixture moved between known positions. For each image, a 6-DOF pose is calculated and the distance between each pose in the scene is calculated. As the fixture is moved between known distances the calculated and expected distances can be compared and an error is computed. Compared with current benchmarking datasets [12], [8], [7] we do not compare with ground truth, but instead measure the mean error of the expected and found distance. This is done to avoid introducing additional errors from uncertainty in the ground truth information.

The results for the dataset is shown in Tab. I. From these results it can be seen that our method is almost a factor ten less erroneous compared with Halcon2D [21] and more than twice as good as 2D-ICP [26] which was manually set and adjusted. Therefore, when a very low uncertainty
is necessary, i.e. 1mm, it is only possible with our method. A comparison between Halcon2D and our method is shown in Fig. 7.

![Fig. 7. Histogram of distance errors compared with the expected from the table. Comparing our method with Halcon2D at 1.5m distance. Our result is the narrow black columns and Halcon2D is shown by the white columns with black top.]

B. Testing on the Tejani dataset

To further ascertain the validity of our method we also tested it on an existing dataset [8]. The dataset consists of six scenes each with a different object placed on a table. The scene contains both background clutter and occlusion. As the dataset does not contain any plane information, we obtain the plane by the depth data. The plane is then removed from the data to remove clutter. The method is then run as described in Section III, with the difference that the final edge-based refinement is omitted. This was done as object models does not correspond precisely with the actual data, up to 15% difference and as such the very precise refinement was not possible. A results of such fitting with "CP+Line2D+Cor" is shown in Fig. 8.

![Fig. 8. Example of a pose estimation with outline of fitted template projected onto object. The gray area is the detected table which has been removed.]

The results are shown in Tab. II. It is seen that at 5cm distance a very high recognition rate is achieved without the online part. But when the threshold is reduced to 2cm this rate drop significantly. By using the correction, the results improve even though the data is noisy and occlusion is present. But the correction method is not able to compensate for all errors. The number of detections within 5cm is a lot higher and thus the possibility for even further correction.

V. CONCLUSION

We have presented a method for obtaining very precise 6-DOF pose estimations. The method is intended to bridge the gap between current pose estimation systems and industrial requirements. The algorithm has been tested against current state of the art 2D pose estimation methods, in a setup resembling a possible industrial use case. Our method consistently shows higher accuracy. The system was also tested on a noisy benchmarking dataset. While the feasibility of online rendering is shown on the benchmarking dataset, the limitations are also visible, as all possible poses are not corrected. To further increase the performance a more robust matching in the fine-tuning is required. In spite of this, we have shown a system which is usable for pose estimation in industrial applications where the objects are required for manipulation. This system has shown its value at the World Robot Summit Assembly Challenge were detected objects were used for assembly. Additionally, surprise objects were introduced shortly before the competition, were our simple setup procedure allowed us to detect all of these.

REFERENCES

### TABLE I

<table>
<thead>
<tr>
<th>Method</th>
<th>Dist</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP+Line2D</td>
<td>5cm</td>
<td>84</td>
<td>99</td>
<td>93</td>
<td>95</td>
<td>94</td>
<td>99</td>
<td>93</td>
</tr>
<tr>
<td>CP+Line2D</td>
<td>2cm</td>
<td>69</td>
<td>95</td>
<td>61</td>
<td>57</td>
<td>70</td>
<td>84</td>
<td>60</td>
</tr>
<tr>
<td>CP+Line2D+Cor</td>
<td>2cm</td>
<td>78</td>
<td>94</td>
<td>89</td>
<td>80</td>
<td>72</td>
<td>89</td>
<td>85</td>
</tr>
</tbody>
</table>

**RESULTS**

The resulting mean error between detections compared with known distances. For the **Halcon2D** algorithm the results were omitted if the distance is larger than 5 cm. Our method did not have any outliers. The 2D-ICP was put in a very close position with small error and fine-tuned until it no longer converged.

---

### TABLE II

**RESULTS** for our method on the tabletop dataset [8]. The results are determined correct if the distance to the nearest match is less than the distance set in column "Dist".

---


