Fast robust peg-in-hole insertion with continuous visual servoing

Rasmus Laurvig Haugaard
SDU Robotics
University of Southern Denmark Denmark
rlha@mmmi.sdu.dk

Jeppe Langaa
SDU Robotics
University of Southern Denmark Denmark
jela@mmmi.sdu.dk

Christoffer Sloth
SDU Robotics
University of Southern Denmark Denmark
chsl@mmmi.sdu.dk

Anders Glent Buch
SDU Robotics
University of Southern Denmark Denmark
anbu@mmmi.sdu.dk

Abstract: This paper demonstrates a visual servoing method which is robust towards uncertainties related to system calibration and grasping, while significantly reducing the peg-in-hole time compared to classical methods and recent attempts based on deep learning. The proposed visual servoing method is based on peg and hole point estimates from a deep neural network in a multi-cam setup, where the model is trained on purely synthetic data. Empirical results show that the learnt model generalizes to the real world, allowing for higher success rates and lower cycle times than existing approaches.

Keywords: Peg-In-Hole, Visual Servoing

1 Introduction

Uncertainties in robotic systems accumulate from various system-parts, including robot kinematics, tools, object grasping, camera calibrations, etc. In high-precision tasks, some uncertainties between target objects may have to be reduced. One approach to reduce the uncertainty is better global calibration, but that is likely to be in conflict with other desired system features, like passive grippers, passive compliance, or modular tool systems which enable more agile systems, but also introduce further uncertainties. Another approach to reduce uncertainties which remedies the above limitations is to handle the uncertainty live and locally, e.g. with force and/or visual feedback, depending on the task.

While many high-precision tasks exist, we focus on a common industrial task, inserting pegs into holes (peg-in-hole). Compliant insertion based on force feedback is effective, when the uncertainty is low enough to enable rich peg-hole interaction. If the uncertainty is slightly larger, classical methods like random search and spiral search can effectively reduce the uncertainty to enable compliant insertion. When the uncertainty becomes larger, the classical search methods become slow, as they blindly search the uncertainty region. More importantly, they assume that the surface is flat around the hole within the uncertainty region, which is only a good assumption if the region is small, and are likely to fail otherwise. In addition, these methods can be harmful for the surface quality of both the peg and the hole-object, and they might result in dropping the peg, if the grasp-force is inadequate, which is not unlikely with passive grippers.

Visual feedback has the potential to effectively reduce larger uncertainties. Only detecting one of the target objects though, only reduces some of the uncertainties. E.g. in the peg-in-hole case, mounting in-hand cameras on the peg-robot to detect the hole in the robot's tool-frame will ideally eliminate uncertainties from robot kinematics and the hole position relative to the peg-robot base, but it will not reduce the uncertainties from tooling and grasping but instead introduce new uncertainties from camera calibration. Classical vision methods tend to either be sensitive to lighting and background clutter or require visual markers.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

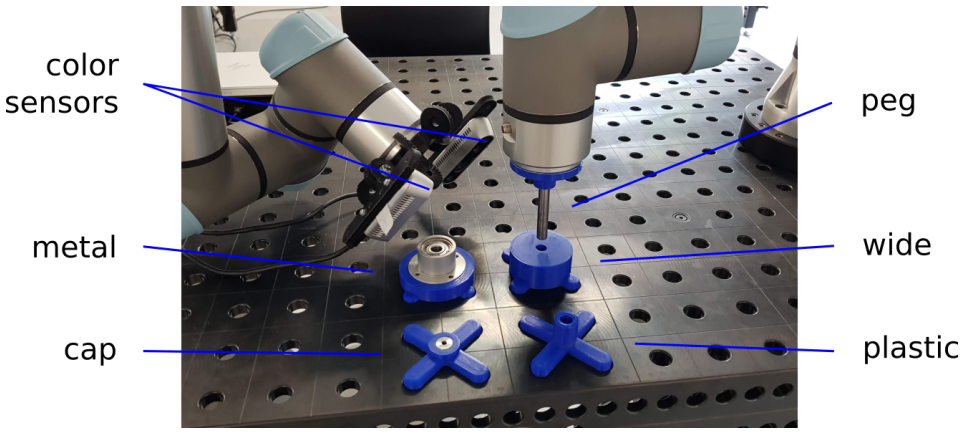

Figure 1: Setup consisting of two UR5 robots where two Intel Realsense D435 are mounted on one robot (camera-robot) and a peg is mounted on the other robot (peg-robot). Four hole-types, *metal*, *plastic*, *wide* and *cap* are mounted to the table. For the *cap* hole, an M4 bolt is used as the peg. Note that we only use the color sensors, and thus the size of the camera tool can be greatly reduced.

We propose learnt visual servoing for the initial peg-hole alignment which is robust to system calibration, grasping uncertainties, surface geometry and lighting conditions, while being significantly faster than spiral search, random search and recent methods based on deep learning. The servoing is based on peg and hole point estimates from a deep neural network in a multi-cam setup, where the model is trained on purely synthetic data.

Figure 1 shows our setup consisting of two cameras mounted on one robot (camera-robot), a peg mounted on another robot (peg-robot), and four holes. Our method also allows the cameras to be attached to the peg-robot or be fixed to the table, but having the cameras on a separate robot provides high flexibility, enabling visual servoing for multiple tools on multiple robots. To test generalization, the setup includes two peg types and four hole types, covering multiple materials, scales and tolerances, as well as different peg and hole surface geometries.

2 Related Work

A vast amount of robotic applications rely mainly on predefined and static movements, which do not account for dynamical changes of the environment or task. In contrast, visual servoing incorporates a visual feedback into the control loop, which enables the robot to deal with large deviations. A variety of different visual servoing methods exists including image-based visual servoing [1], pose-based visual servoing [2] and hybrid approaches [3] that use parts of both. Visual servoing opens up for a lot of possibilities by providing feedback without tactility, e.g. tracking and grasping objects with a robot [4, 5]. Classical visual servoing requires that the robot is accurately calibrated, while more recent methods have tried to address this problem with uncalibrated visual servoing methods [6, 7, 8], but these methods needs more time to converge, due to the omitted models of the kinematics and camera calibrations. Instead, we propose to make an initial calibration of the system, but to be robust towards uncertainties in the calibration, similar to [9]. This ensures that small changes to the setup do not affect the performance of the method, while still exploiting the advantages of a calibrated system.

In deep learning, domain randomization has enabled training models on synthetic data and successfully transferring them to the real world without further training in a variety of tasks, including object detection [10], using a robotic hand to manipulate a cube with reinforcement learning [11], car detection and pose estimation [12] and affordance learning [13]. [10, 12, 13] use synthetic 3D distractor objects to improve their model's generalization. Instead, we overlay natural images on our synthetic images in order to improve generalization.

More closely related to our proposed method is [14], which applies visual servoing for peg-in-hole alignment assuming a full pose- estimation from classical methods, where peg and hole are clearly marked. Although the full pose enables the method to handle the rotational alignment of the peg,

the requirement of peg and hole markers make such methods undesirable in most use cases. Our method does not require visual markers.

Recently, [15] proposed image-based visual servoing for peg and hole alignment prior to insertion, which is similar to our approach. They fix two in-hand cameras to the peg-robot in a top-view configuration and train a deep neural network to classify if the robot should move in one of four directions, or whether they should start the insertion. Their visual servoing is step-wise, not continuous. While they outperform spiral search in their results, forming the problem as classification and using step-wise visual servoing makes the method too slow for practical use with insertion durations in the range of 20-70 seconds. Also, since they only detect the hole, their method does not handle grasping and tool uncertainties. In contrast, we estimate the positional error of the peg-robot based on both peg and hole point estimates, which handles grasping and tool uncertainties, and reduce the error continuously, enabling significantly faster insertions.

3 Methods

This section presents an overview of the alignment methods examined in this work, including the proposed visual servoing. Our visual servoing alignment is divided into two tasks: First, the hole and peg center points are estimated in images using a deep neural network. Secondly, based on the point estimates, the peg is aligned to the hole in the plane that is perpendicular to the insertion direction. After alignment, the peg is inserted with compliance using force-feedback.

3.1 Overview of peg-in-hole alignment

We examine three methods for peg-hole positional alignment: random search, spiral search and our proposed visual servoing alignment. The methods are illustrated in Figure 2. Note that all three methods only align the position, not the orientation. In random search, a point within the uncertainty boundary is sampled and an insertion is attempted at that point. This is repeated until the peg reaches below the hole-surface or a time limit is exceeded. In spiral search, the peg moves outwards in a spiral while pressing against the hole-surface. The peg is partially inserted in the hole when it moves over the hole within a success region based on the peg- and hole tolerances. Spiral search will continue until a force limit is reached, indicating that the peg is in contact with the hole, or until the peg exceeds the uncertainty boundary. In our proposed visual servoing, both peg and hole center points are continuously estimated in two or more cameras. An error vector is estimated based on the estimated points, and the error is attempted minimized in a servoing loop. Visual servoing will continue either until convergence, until a time limit is reached or until the peg exceeds the uncertainty boundary. At convergence, the peg is moved downward along the insertion direction until force-feedback indicates contact. To further increase robustness during alignment, visual servoing can be followed by spiral search or random search with a smaller uncertainty region where appropriate.

After successful positional alignment, there may still be small position uncertainties as well as orientation uncertainties. To remedy this, the peg is inserted with compliance using force-feedback.

3.2 Point estimation

We make point estimates based on heatmaps from deep neural networks, similar to [16, 17] which estimate human pose keypoints, but here adapted to pegs and holes. As shown by [18], outputting heatmaps allows for better spatial generalization than directly regressing point coordinates from a global, latent representation. Let $p = (x, y)^T$ be the pixel coordinates of a point in an image, I. Then a desired heatmap, Φ, from a desired point, p^*, can be defined using a Gaussian kernel:

$$\Phi_{p^*} = \exp\left(-\frac{||p - p^*||^2}{2\sigma^2}\right), \tag{1}$$

where σ is a hyper parameter controlling the size of the active area of the heatmap. We set $\sigma = 3$px and have separate heatmaps for the peg and hole points. Figure 4.f shows an example of target heatmaps.

A deep neural network, f, with trainable parameters, ϕ, estimates the heatmap, $\hat{\Phi} = f_\phi(I)$, with the same resolution as the input image. We choose a U-Net [19] architecture with an ImageNet pre-

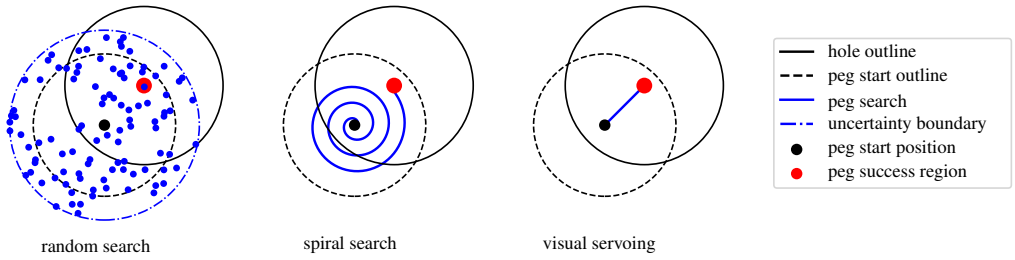

Figure 2: Illustration of the three examined methods for peg-in-hole alignment. The paths between the points in random search are left out for clarity. Note that the peg is in contact with the hole-surface at the search points in random search and throughout spiral search while the peg moves above the surface in visual servoing.

trained ResNet18 [20] backbone for simplicity and to leverage transfer learning while keeping the inference time low. As will be apparent from the results in Section 4, this architecture is sufficient to model the problem at hand.

At inference, the point with the largest corresponding value in the estimated heatmap is chosen as the point estimate, $\hat{p}$:

$$\hat{p} = \underset{p}{\operatorname{argmax}}\, \hat{\Phi}_p \tag{2}$$

3.2.1 Data generation

Our proposed model used in our real-world experiments is trained purely on synthetic data with no manual annotations, inspired by [10, 11, 12, 13]. The virtual scene consists of a camera, a cylinder and a surface with a hole in it. For each image, peg- and camera transforms are sampled, a high dynamic range image (HDRI) environment map is sampled, and a procedural material is sampled. We do not apply randomization to the geometry.

When synthesizing an image, the distance from the camera to the hole is sampled uniformly between 12 cm and 15 cm. The elevation (the angle between the hole plane and the optical axis) is sampled between 35° and 45°. The roll angle (the rotation around the optical axis) is sampled between -5°and 5°. The peg position is sampled from a disc centered around the hole with a radius of 15 mm and a height relative to the hole between 5 mm and 15 mm. The peg orientation error is sampled as an axis angle vector, where the axis is sampled from the surface of a unit sphere, and the angle is sampled between 0° and 5°.

We use an in-house developed framework [21] for synthetic image generation in Blender[1] with procedural materials and HDRI environment maps from HDRI Haven[2]. Environment maps provide a large variety of natural lighting and reflections. We hypothesize that realistic, diverse lighting and reflections enable the model to learn features that generalize well to the real world. The sampled, procedural material is applied to both the hole-surface and the peg to discourage the model from learning to distinguish between different samples of the procedural material. We render 1000 images in about 10 minutes on an NVIDIA GeForce RTX 2080. Examples of the synthetic renders are shown in Figure 3.

If the images shown to the model during training only contain the peg and the hole-surface, the model will likely be sensitive to background clutter. One approach to remedy this is to use natural images as backgrounds as in [22], but this encourages the model to discriminate between the natural- and synthetic domain. [22] alleviates the problem by using somewhat realistic models and freezing a pre-trained backbone. [10, 12, 13] overcome the problem by using 3D distractor objects, such that both falses and positives are from the synthetic domain. Since it is not obvious how to obtain a diverse set of 3D distractor objects, we instead introduce distractors from natural images. We

[1] https://www.blender.org/
[2] https://hdrihaven.com/

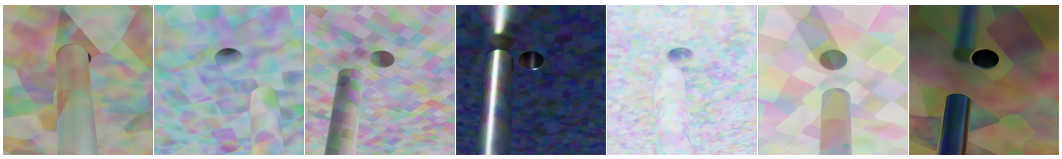

Figure 3: Examples of synthetic renders with procedural materials and HDRI environment maps.

Figure 4: Example of a synthetic training image. a) Synthetic render. b) Overlay. c) Mask. d) Composite image using a-c. e) Composite image with random crop, random horizontal flip, and random blur. f) Target heatmaps (hole: blue channel, peg: red channel).

sample a natural image as an overlay image and another natural image, converted to grayscale, as the alpha channel of the overlay image. We sample the natural images from the MS COCO dataset [23]. The augmentation is illustrated in Figure 4(a-d). Since the overlay from the natural domain is imposed on the whole image, the model will not be able to simply distinguish between the synthetic- and natural domain. Also, natural overlays on peg and hole will effectively serve as extra randomization on peg and hole appearance, which can be hard to simulate. Since compositing is computationally efficient, the augmentation is done live during training with different overlays at each epoch, effectively reducing the amount of required renders.

To evaluate the model trained on synthetic images, we obtain natural datasets for the *metal* and *plastic* holes. To obtain the datasets, a rigid camera mount is attached to the camera-robot, the whole setup is calibrated, and robot movements between consecutive sampled peg and camera transforms are ensured to be safe. The peg and camera transforms are sampled like for the synthetic images. We do not attempt to add any variations to the natural environment, e.g. turning on and off lights or capturing images from multiple rooms, since it is impractical.

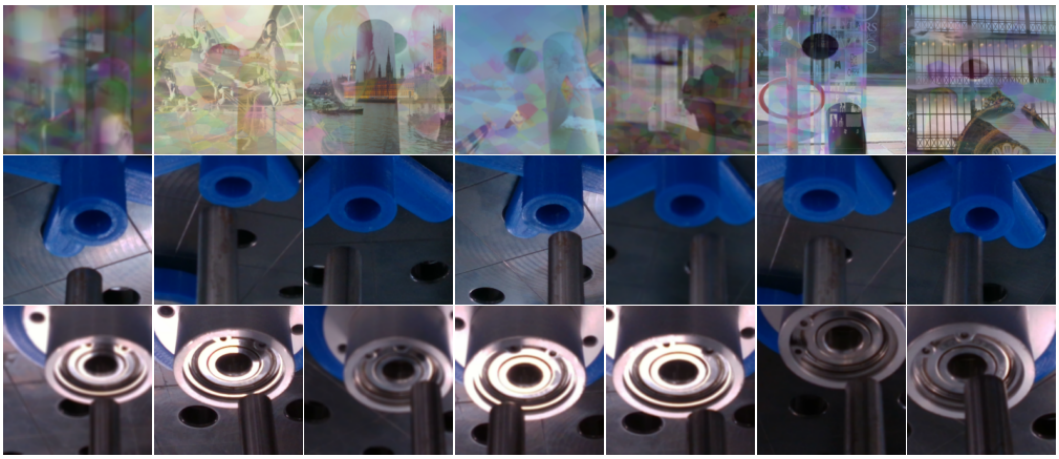

Figure 5: Examples from the three training datasets, from top- to bottom row: synth, plastic, metal.

3.2.2 Training details

All images are resized to 224x224 before they are handed to the model. The loss is defined as the mean squared error between the desired and output heatmaps. Each of our datasets consists of 1000 images, where 100 of them are reserved for validation. We also trained on 10x more images on the synthetic domain while keeping the total amount of parameter updates, but with little to no effect. We conclude that the data augmentation provides enough variation within the domain. We train our models with the one-cycle [24] policy for 30 epochs with a maximum learning rate of 0.001 and weight decay set to 0.0001. During training, both synthetic and natural images are augmented with random crop, random horizontal flip and a 50% chance of box blur with a kernel size of either 3 or 5 px. The augmentation is illustrated for a synthetic image in Figure 4. One training takes about 8 minutes on an NVIDIA GeForce RTX 2080, and we obtain single-image inference at 150 Hz.

3.3 Visual servoing

Our method requires regions of interest (ROIs) in the images, providing crops as seen in Figure 5, as well as approximated depths (distance from cameras to peg and hole). ROIs and depths can either be calculated based on an approximate hole frame or by manually marking the ROIs in the full images once for a given setup. Our visual servoing alignment method is outlined in Algorithm 1.

Algorithm 1: Visual servo positional alignment

Input: Insertion direction, l. Estimated depths, z_i, in n cameras. Target convergence threshold, ϕ_t. Filter parameters α_τ, α_γ and α_ϕ.

$\tau \leftarrow$ initial peg-robot position // filtered target position

$\phi \leftarrow 10\phi_t$ // filtered error magnitude

while $\phi > \phi_t$ **do**

  $q \leftarrow$get position of peg-robot

  for $i = 1, ..., n$ **do**

    $I_i \leftarrow$ get image from camera i

    $p_i^*, h_i^* \leftarrow$ estimate peg and hole points in I_i

    $p_i, h_i \leftarrow$ estimate 3D points based on p_i^*, h_i^*, z_i and the system state- and calibration

    $c_i \leftarrow$ get position of camera i

    $v_i \leftarrow (p_i + h_i)/2 - c_i$ // view direction

    $u_i \leftarrow \dfrac{v_i \times l}{||v_i \times l||}$ // direction of error from this view

    $b_i \leftarrow u_i \cdot (h_i - p_i)$ // magnitude of error from this view

  end

  $(A, b) \leftarrow \left(\begin{bmatrix} u_{1x} & u_{1y} & u_{1z} \\ u_{2x} & u_{2y} & u_{2z} \\ & \vdots & \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ \vdots \end{bmatrix} \right)$

  $\hat{e} \leftarrow$ solve (A, b) by least squares // estimated error vector

  $t \leftarrow q + \hat{e}$ // unfiltered target position

  $\tau \leftarrow \alpha_\tau \tau + (1 - \alpha_\tau)t$ // update filtered target position

  set peg-robot reference to τ // move robot towards τ

  $\gamma \leftarrow \alpha_\gamma \gamma + (1 - \alpha_\gamma)\hat{e}$ if γ is defined, else $\hat{e}$ // update filtered error vector

  $\phi \leftarrow \alpha_\phi \phi + (1 - \alpha_\phi)||\gamma||$ // update filtered error magnitude

end

While performing the alignment with visual servoing, the robot moves in a plane perpendicular to a provided insertion direction, l. Note that the insertion direction is the known direction in which the peg-robot will move when the positional alignment along the plane is done. It does thus not need to be exactly parallel to the axis of the peg or the hole. Because of depth-ambiguity, points from the images of the i'th camera provide information only along the direction, u_i, in the movement plane that is perpendicular to the view direction, v_i. The point estimates from the image are projected to 3D based on the system state, system calibration and constant depth estimates, z_i, used for both the peg- and hole point, p_i, h_i. Peg and hole estimates from multiple cameras then allow us to estimate the peg-robot error vector $\hat{e}$.

A target position, t, for the peg-robot is found by adding the estimated error vector, $\hat{e}$, to the current peg-robot position, similar to visual servoing algorithms that use the estimated image Jacobian for controlling the robot. The target position and the error magnitude are filtered as shown in Algorithm 1 with coefficients $\alpha_\tau = \alpha_\gamma = \alpha_\phi = 0.9$. Finally, we use the built-in servoing api from the UR5 robot to move towards the target position, and terminate the visual servoing when the filtered error magnitude is smaller than the threshold ϕ_t, which is set to be 1/20 of the peg-hole diameter.

4 Experiments

In this section, we empirically validate different aspects of our visual servoing system. We start with an accuracy test of the visual part, the point estimation, and then we go on to test the full system, including both alignment and insertion.

4.1 Point estimation accuracy

Three models are trained on images of *plastic*, images of *metal* (see Figure 1), and the synthetic images, respectively (900 images per dataset). Each model is evaluated on all validation datasets (100 images per dataset). The cross-performance is illustrated in Figure 6, showing detection success rates versus increasing distance tolerances. The models trained on the natural domains fail on the synthetic domain, and more importantly, the model trained on the *metal* domain fails to detect the *plastic* hole consistently. In contrast, the models trained on the synthetic domain perform consistently well in all three domains, indicating that the synthetic images force the models to learn more general features.

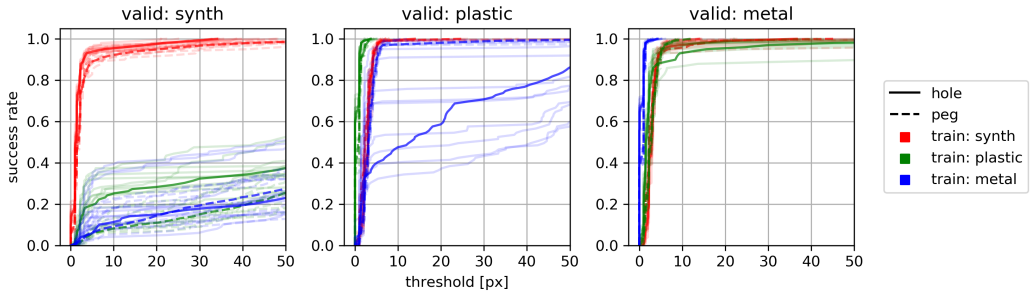

Figure 6: Point detection success rates at different pixel thresholds. 10 models are trained for each of the three training datasets, and each model is evaluated on all of the validation datasets. Both mean performance per training dataset and the individual model performances are plotted. The individual performances are plotted in a lighter color for clarity. Note that the peg- and hole diameter is approximately 50 px in the images.

4.2 Peg-in-hole performance

Using models trained on all three domains, we compare our visual servoing to random search and spiral search. The results are shown in Table 1 See supplementary material for a video of the methods.

The initial peg positions are sampled from a disc centered around the hole with a diameter of three times the peg diameter and a height relative to the hole between 5 mm and 15 mm (3 mm and 5 mm for *cap*). The peg-hole orientation error is sampled up to 2°. To show robustness towards calibration, camera extrinsic errors are sampled with up to 2° in orientation and 10 mm in position. Note that these are large calibration errors.

The peg used for *bearing*, *plastic* and *wide* is the 10 mm shaft with an h7 tolerance shown in Figure 1. The *bearing* hole is 10 mm with an H7 tolerance while the *plastic* and *wide* holes are 3D-printed with a measured diameter of 10.6 mm and 10.4 mm, respectively. The *cap* hole is 4.4 mm and the peg is an M4 bolt with a measured diameter of 3.9 mm.

	classical search methods		visual servoing, trained on:			
	random	spiral	plastic	metal	**synth (proposed)**	optimal
metal	-	10% (7.1 s)	**99%** (4.1 s)	**100%** (4.0 s)	**100%** (3.6 s)	3.2 s
plastic	0% (-)	27% (10 s)	**100%** (2.4s)	**100%** (2.7 s)	**100%** (2.2 s)	1.0 s
*plastic**	-	-	93% (2.6 s)	11% (3.7 s)	**100%** (2.3 s)	-
wide	4% (15 s)	**100%** (36 s)	**99%** (2.4 s)	**100%** (2.2 s)	**100%** (2.3 s)	1.0 s
cap	34% (16 s)	-	1% (1.7 s)	0% (-)	**100%** (1.6 s)	0.4 s

Table 1: Success rates and mean time for successful attempts. Timings indicate the total time from the initial position to reaching an insertion depth of 10 mm (5 mm for *cap*). Each entry is based on 100 runs. Highest success rates are marked in bold. If an entry is not significantly lower than the highest success rate with a p-value of 5%, it is also marked in bold. Optimal: peg is aligned to the hole based on calibration (no search).

Figure 7: Example images from the cameras during visual servoing of the peg-in-hole cases. From left to right: *metal*, *plastic*, *plastic**, *wide* and *cap*.

Random search relies on detecting the peg reaching below the hole-surface. For this reason we do not apply randomization in peg height. It also means that random search requires a rather large tolerance compared to the uncertainty, which is why random search is not considered for *metal* with H7/h7 tolerance. Visual servoing is followed by spiral search as mentioned in Section 3.1 for *metal*, *plastic* and *wide*, but is avoided for *cap* to avoid damaging the hole surface with the M4 bolt. For the same reason, we do not attempt force-based insertion for *cap*.

In the *plastic** experiment, a light source was placed pointing into the cameras, causing flares and hard reflections on the hole-surface (see Figure 7). All attempts were successful. The peg for the *cap* hole is threaded, and is thus far from any of the three examined domains. The models trained on the natural domains completely fail to generalize, while the model trained on the synthetic domain succeeds in all the attempts.

The attempts with our proposed visual servoing are consistently both faster and more robust than random search and spiral search. Furthermore, the model trained on the synthetic domain generalize to all the examined cases, in contrast to the models trained on natural domains. [15] also performs peg-in-hole visual servoing based on deep learning on synthetic images. They use a 10 mm peg and a 10.4 mm hole, similar to *wide*, and a similar but only positional error in the start positions. They achieve insertion times between 20 s and 70 s, while our average insertion time is 2.3 s.

5 Conclusion

We studied the use of deep learning based point estimation and continuous visual servoing for peg-in-hole positional alignment. We demonstrated that our method can significantly increase speed and robustness compared to classical methods like random search and spiral search. We also showed that our method is significantly faster than a previous attempt based on deep learning. Since visual servoing does not require peg-hole contact, our method is robust to hole surface geometry and gentle to the peg and hole surfaces. We trained point estimation models on both natural and synthetic domains. We demonstrated that synthetic domain randomization and using distractors from natural images introduces enough variation to enable the model to generalize to all the examined peg-in-hole cases. While compliant force insertion is likely not enough to account for larger angle errors in tight tolerances, integrating axis alignment in visual servoing could remedy this problem and is an area for future work. Finally, our method is focused on peg-in-hole tasks but we hypothesize that deep learning based visual servoing can be applied successfully in many assembly sub-tasks.

Acknowledgments

The authors gratefully acknowledge the economic support from Innovation Fund Denmark through the project MADE FAST.

References

[1] L. Weiss, A. Sanderson, and C. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal on Robotics and Automation*, 3(5):404–417, 1987.

[2] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice. Position based visual servoing: keeping the object in the field of vision. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1624–1629 vol.2, 2002.

[3] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, 1999.

[4] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, 1993.

[5] H. Wu, T. T. Andersen, N. A. Andersen, and O. Ravn. Application of visual servoing for grasping and placing operation in slaughterhouse. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 457–462, 2017.

[6] J. A. Piepmeier, G. V. McMurray, and H. Lipkin. Uncalibrated dynamic visual servoing. *IEEE Transactions on Robotics and Automation*, 20(1):143–147, 2004.

[7] Z. Qiu, S. Hu, and X. Liang. Model predictive control for constrained image-based visual servoing in uncalibrated environments. *Asian Journal of Control*, 21(2):783–799, 2019. doi:10.1002/asjc.1756. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asjc.1756.

[8] Y. Chang, L. Li, Y. Wang, and K. You. Toward fast convergence and calibration-free visual servoing control: A new image based uncalibrated finite time control scheme. *IEEE Access*, 8: 88333–88347, 2020.

[9] C. J. Taylor and J. P. Ostrowski. Robust vision-based pose control. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2734–2740 vol.3, 2000.

[10] S. Hinterstoisser, O. Pauly, H. Heibel, M. Marek, and M. Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object instance detection. *CoRR*, abs/1902.09967, 2019. URL http://arxiv.org/abs/1902.09967.

[11] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation, 2018.

[12] R. Khirodkar, D. Yoo, and K. M. Kitani. Domain randomization for scene-specific car detection and pose estimation, 2018.

[13] A. Hämäläinen, K. Arndt, A. Ghadirzadeh, and V. Kyrki. Affordance learning for end-to-end visuomotor robot control. *CoRR*, abs/1903.04053, 2019. URL http://arxiv.org/abs/1903.04053.

[14] S. Huang, K. Murakami, Y. Yamakawa, T. Senoo, and M. Ishikawa. Fast peg-and-hole alignment using visual compliance. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 286–292. IEEE, 2013.

[15] J. C. Triyonoputro, W. Wan, and K. Harada. Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data. *CoRR*, abs/1902.09157, 2019. URL http://arxiv.org/abs/1902.09157.

[16] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. *CoRR*, abs/1804.06208, 2018. URL http://arxiv.org/abs/1804.06208.

[17] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[18] A. Nibali, Z. He, S. Morgan, and L. Prendergast. Numerical coordinate regression with convolutional neural networks. *CoRR*, abs/1801.07372, 2018. URL http://arxiv.org/abs/1801.07372.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[21] Rasmus Laurvig Haugaard. synth-ml, 2020. URL https://gitlab.com/sdurobotics/vision/synth-ml.

[22] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. *CoRR*, abs/1710.10710, 2017. URL http://arxiv.org/abs/1710.10710.

[23] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[24] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. URL http://arxiv.org/abs/1803.09820.