Does Vision Work Well Enough for Industry?

Frederik Hagelskjær, Anders Glent Buch and Norbert Krüger
Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark

Keywords: Pose Estimation, Industrial Vision, Setup Time, Usability.

Abstract: A multitude of pose estimation algorithms has been developed in the last decades and many proprietary computer vision packages exist which can simplify the setup process. Despite this, pose estimation still lacks the ease of use that robots have attained in the industry. The statement ”vision does not work” is still not uncommon in the industry, even from integrators. This points to difficulties in setting up solutions in industrial applications. In this paper, we analyze and investigate the current usage of pose estimation algorithms. A questionnaire was sent out to both university and industry. From this survey, it is clear that the actual setup time of pose estimation solutions is on average between 1–2 weeks, which poses a severe hindrance for the application of pose estimation algorithms. Finally, steps required for facilitating the use of pose estimation systems are discussed that can reduce complexities and thus the setup times in industrial applications.

1 INTRODUCTION

Using vision for pose estimation could significantly increase the possible use cases for robotics, since expensive positioning systems such as bowl feeders could be avoided. However, at the moment, setting up a pose estimation system is very complicated and any re-configuration is a task almost equal in scale. That is one reason why ”blind” robot systems are quite common in industry, while vision based solutions are seldom seen. Even from integrators statements such as ”vision does not work” are still common in industry.

The flexibility of a robot arm provides a tool which can be reconfigured to different tasks using little programming. This flexibility has enabled a massive growth of robotics in industry, while there are still many complexities involved in the setup of a vision based robotic system (Krüger et al., 2014). Thus, while the number of robots in production is increasing, SMEs (Small- and Medium-sized Enterprises) have a much lower percentage of robots than large companies (Sørensen et al., 2015), as they often produce in much smaller quantities.

The current decade has seen the introduction of several robotic arms with simple setups; an example is the Universal Robot. A task can quickly be setup using 6D poses, given from either the controller or by kinesthetic guidance. With this manipulator, one can quickly generate repetitive movements, enabling automation of tasks for small productions. The limit of these ”blind” robot solutions is that the object poses need to be deterministic.

Several methods and software packages are currently available for doing pose estimation. However, for the integration of pose estimation into robot systems, many obstacles still exist.

High Requirements for Performance: The system should be able to quickly detect the object, allowing the production to continue at acceptable speed.
Demand for Robustness: Depending on the robot setup, the number of acceptable mis-detections can be as low as zero.
Inflexible Setups: A complete setup is often made to perform for a single task. Any change in the task configuration will often demand an entirely new solution.
Long Development Time: Overcoming the previously mentioned obstacles will often result in extensive testing before the system is working in an acceptable way.
Expensive System: Having an expert working on the setup and a long development time will make the system expensive.

Using a questionnaire, (see Fig 1), we have analyzed and examined the current situation of the application of pose estimation in academia and industry. We received 23 responses of which 11 were from people working in the industry (see Fig 2). In the following plots, we split the answers into industry and academia, respectively plotted as red and blue.

Our questionnaire sheds light on the actual situa-

Hagelskjær, F., Buch, A. and Krüger, N.
Does Vision Work Well Enough for Industry?.
DOI: 10.5220/0006645301980205
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 4: VISAPP*, pages 198-205
ISBN: 978-989-758-290-5

- What is your current employment?
- For how many years have you been using computer vision?
- How long time does it normally take you to setup and complete a pose estimation task?
- How many vision tasks using pose estimation have you solved in the last year?
- What software have you used for pose estimation the last year?
- How much time do you as a vision expert use on the physical aspect (light, camera lens, etc.) vs. software (algorithm, tuning, etc.), when doing pose estimation?
- How often is your vision system a part of a robotics setup?
- How often do you use a CAD model in your pose estimation?

Figure 1: List containing all the questions sent out in the questionnaire. The possible answers were provided as a range of multiple choice selections.

tion of the application of pose estimation algorithms in academia and industry. The following overall statements can be derived from the outcome of the questionnaire:

- More than 50% of pose estimation algorithms are applied in the context of a robotic system. Hence it is important to see pose estimation in this context.
- Although using currently available proprietary pose estimation software allows for applying a concrete algorithm within few hours, the average time used for making the system applicable for a task is still between 1–2 weeks in industry and 2-4 weeks in academia.
- Most time is spent on adapting a given software to a task and not for deciding about external factors such as sensors, light sources and the actual placement of this equipment.
- Developers often use more than one software package, indicating that it is far from being clear what algorithms to choose for a given task.

These insights make it clear that—in addition to improving the precision and accuracy of algorithms—an important additional task is to facilitate the actual use of available pose estimation algorithms. After having reported on the results of the questionnaire in section 3, we give pointers to issues that should be addressed in pose estimation research in that context in section 4 and 5.

2 STATE OF ART

There has been a significant development in methods for pose estimation which can solve many complex tasks using both 2D and 3D, sensors (see, e.g. (Miksik and Mikolajczyk, 2012) and (Guo et al., 2014)).

This section gives an overview of the current state of the art, both in terms of algorithms as well as vision systems applied in industry.

2.1 2D Methods

With a standard RGB or mono camera, the input data is a 2D matrix of light intensity. Even though the data is without any 3D information, there exist methods which can return 6D poses. Two primary methods exist for handling such 2D data, i.e. template matching (Lewis, 1995) and feature matching (Gossow et al., 2012), respectively global and local approaches.

Template Matching (Global Approaches): In template matching, a model of the object that we wish to detect is used to match patches of the image. Methods to avoid instabilities connected to illumination changes, use edges instead of the intensity image (Lewis, 1995). By utilizing the orientation of the edges, (Hinterstoisser et al., 2012) newer additions have been able to enhance the robustness. However, template matching is not robust to occlusions or other aspects that hinder edge detection. Despite these problems, template matching is a widely used method which under the proper circumstances can be extremely accurate.

Feature Matching (Local Approaches): To avoid the drop in performance as a result of occlusion and background clutter small patches are matched withe the image as opposed to the full object. The procedure is to detect interest points, calculate descriptors at each interest point and then try to match the descriptors to find a transformation.

An early method which employed this with great success is SIFT (Lowe, 2004). Using the difference of Gaussian operator to detect interest points and a histogram of gradients as the descriptor, the method creates candidate matches. After evaluating the quality of the matches, usually using RANSAC (Fischler and Bolles, 1981), a transformation is found.

Using the same approach, several alterations have been made to this algorithm, eg. (Bay et al., 2008), (Alahi et al., 2012) and (Rublee et al., 2011). Using feature matching, it is possible to acquire real-time pose estimation of objects (Collet et al., 2011). One limitation of feature matching is that it requires some texture on the object to be detected.

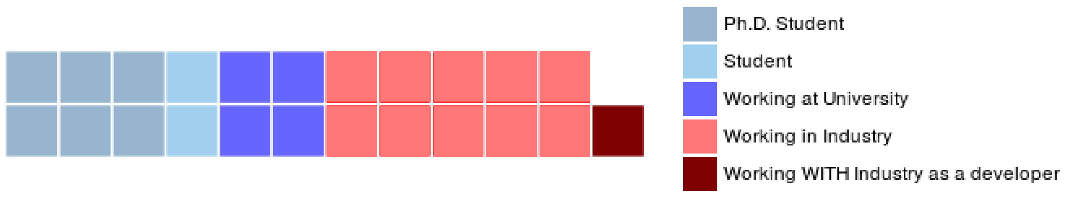

Figure 2: Employment of those who answered the questionnaire. The answer ”Working WITH Industry as a developer” was provided by a participant.

2.2 3D Methods

With the introduction of cheap 3D sensors like the Kinect (Zhang, 2012), along with increased processing power enabling fast stereo processing, a huge in increase in the availability of 3D algorithms has increased greatly in the last two decades. As in the 2D case there exist many different global and local methods.

Global Methods: As in the case of 2D, there are methods for matching the full model in 3D. The method in (Mian et al., 2006) uses tensors of different views of the object, compared with tensors of the scene. Another global approach employed in the proprietary software Halcon is called PPF (Drost et al., 2010). Here point pair features are matched globally to represent the model as opposed to local methods.

Local Methods: With an overwhelming amount of different methods (Guo et al., 2014), feature matching is the most widely investigated approach for pose estimation in 3D data. In these methods, instead of using 2D features, the local 3D space at interest points is represented. One of the early methods is the Spin Image feature (Johnson and Hebert, 1998), which use a histogram of signed and radial distances for neighboring normal vectors as the descriptor. As in the case of 2D methods, there have been several different descriptors using various local 3D information, e.g. SHOT (Tombari et al., 2010) and FPFH (Rusu et al., 2009). Finally, there also exist many different matching schemes for these descriptors (Guo et al., 2014).

2D and 3D methods can also be combined using both color and depth information. C-SHOT (Tombari et al., 2011) is an example of a simple combination. Using the SHOT algorithm (Tombari et al., 2010), a histogram of the RGB intensities around the point are added to the descriptor increasing performance. The large variety of available pose estimation algorithms requires making application-specific choices, which is far from being trivial. This is one of the complexities in setting up pose estimation systems that leads to long setup times for vision solutions in practical applications, as it will become evident from our questionnaire.

2.3 Learning-based Methods

Methods based on training have had much success in the last years for object classification (Krizhevsky et al., 2012). Several machine learning approaches have been shown to work well in the context of pose estimation, e.g., Random Forest (Lai et al., 2011) and Neural Networks (Gupta et al., 2015). These methods use labeled training samples to learn correct parameters for classifying data. This gives the advantage that if the correct hyper-parameters are provided, they can be trained by non-professionals. By using robust parameters, more conventional vision methods can also utilize learning based approaches for a more simple setup (Leibe et al., 2008). The input to these models can be both 2D and 3D data and they can be combined with other methods to improve performance.

The output of thee applications ranges from a classification of images (Krizhevsky et al., 2012), to pixel-vise (Girshick et al., 2014) detection and finally an actual 6D pose (Rad and Lepetit, 2017). Thus a realm of possible solutions is beginning to emerge using machine learning.

2.4 Software Packages for Pose Estimation

There exist many proprietary software packages and open source solutions to ease the development of pose estimation. From figure 3 it is seen that a lot these packages are used in both industry and university. We will here list some of these packages and give a short description of the frameworks. The proprietary packages are listed with an ”(*)”.

Halcon (*): Halcon (MvTec, b) is a library which includes different kinds of 2D and 3D based pose estimation methods. Examples are template matching in 2D images, and many different 3D matching methods.

Matlab Computer Vision Toolbox (*): The computer vision toolbox (Matlab, 2016) in Matlab has a template matching method and a number of different feature matching algorithms as well as training based methods.

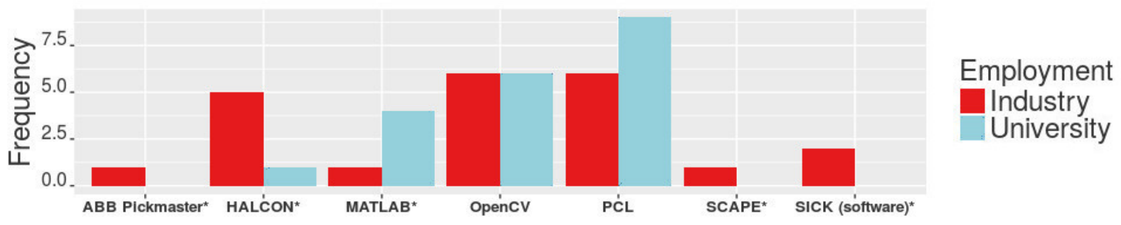

Figure 3: What software have you used in the last year? Proprietary software is marked with ” *”.

Vidi (*): Vidi (Cognex,) provides the ability to create matching systems based on deep learning methods.
Adaptive Vision (*): This is a suite to perform simple detection based on template matching (AdaptiveVision,).
OpenCV: OpenCV (Itseez, 2015) is a vision library for 2D methods. This library has an overwhelming amount of different methods, but to use the library, usually a lot of implementation is necessary.
Point Cloud Library: PCL (Rusu and Cousins, 2011) is an open source library for 3D algorithms implemented in C++. Many methods are available, but as in the case of OpenCV, much coding is required to implement these methods properly.

3 SETTING UP A POSE ESTIMATION SYSTEM

The creation of a pose estimation system usually requires many decisions and reiterations before the system has a sufficient performance. Fig. 5 illustrate the usual workflow, from an initial problem into a working solution. We will use this workflow to present the main results in our investigations.

The very first step is to analyze the situation and determine the success criteria, Fig. 5(A). From this, constraints can be defined and the actual setup can begin, Fig. 5(B). In Fig. 5(C), a sensor system needs to be selected; the sensor establishes both the limitation of workspace and possible algorithms. Then, an algorithm is required to do the actual pose estimation (Fig. 5D) and the parameters can be set (Fig. 5E).

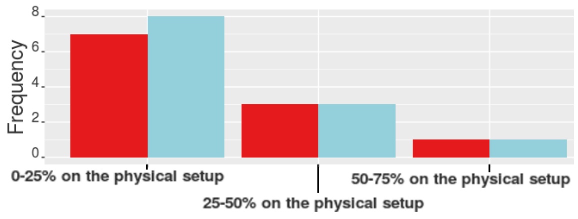

Figure 4: How much time do you spend on physical setup vs software?

The whole setup is then tested (Fig. 5F), and adjusted for several iterations until it performs successfully (Fig. 5G). Depending on the scale of the error, one often goes back multiple stages while trying to adjust the system. We will now elaborate these phases and relate them to the answers in the survey.

Initial Analysis; Fig. 5(B): Pose estimation performed on a dataset often focuses on getting the best performance in terms of number of correctly recognized poses, whereas a robotics setup often has particular requirements (such as processing speed, precision as well as setup time). In Fig. 6 we see that most pose estimation tasks in industry have an actual robotics setup, where such requirements need to be specified.

This will also help to restrict the task, for example, if only one object needs to be detected or some errors can be accepted. Restrictions and requirements for pose estimation are often quite different between robotic setups. Thus an analysis of each situation will make the creation of a pose estimation task much simpler.

Constraints; Fig. 5(B): A user of the pose estimation algorithm may exploit known constraints to the system, which will make the detection more feasible. For example, an object rarely appears in completely random 6D poses. Usually, there are constraints in the workspace of where objects can appear, which could increase the speed and reliability of the pose estimation considerably. The object could also be bound to a particular plane, (e.g. a table, the floor) or some distinct orientations (e.g. stable poses due to gravity), which could increase the precision significantly. This knowledge is of course taken into account when deciding upon the sensor positioning, but usually the available software packages that we are aware of ignore such constraints in the actual pose estimation step. Some 2D algorithms have distance as part of their parameters, whereas many 3D algorithms seldom take additional constraints into account.

Hardware Selection; Fig. 5(C): There exist many different sensors, which are able to provide intensity and color images and various kinds of depth information. Many combinations exist, each with different coverage, precision, resolution, etc.

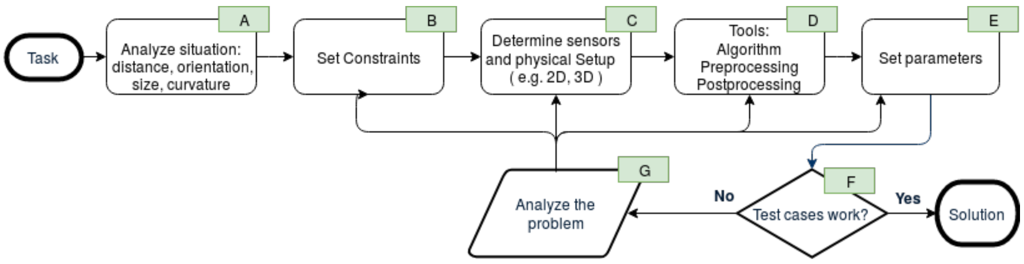
Task
A
Analyze situation: distance, orientation, size, curvature
B
Set Constraints
C
Determine sensors and physical Setup (e.g. 2D, 3D)
D
Tools: Algorithm Preprocessing Postprocessing
E
Set parameters
G
Analyze the problem
No
F
Test cases work?
Yes
Solution

Figure 5: Visualization of the integrators pipeline to create a working pose estimation setup. A number of reiterations are performed.

These are external parameters which set the basis for the following setup. Any constraint established by the hardware influences the rest of the system. Thus it is possible that these parameters need adjustment during development. However from our questionnaire (see Fig. 4) we can see that this is a task which takes much less time than choosing and fine tuning the algorithms of the software. The usual way seems to be to make a qualitative choice and then work hard on finding a software solution afterwards. However, if tuning the software solution does not lead to a satisfying solution, the decision that the whole physical setup might be changed might be required at some point.

Software Selection; Fig. 5(D): As seen in section 2, there exists an overwhelming amount of different algorithms for pose estimation. Many of these can also be combined. There exists both open source algorithms and many kinds of proprietary software. Using knowledge and experience, the user can get a starting point for choosing specific algorithms, but there are no actual guidelines when deciding which algorithm best fits any particular situation. The setup time is often much shorter using proprietary software, but the drawback is a lack of control and flexibility, as the proprietary systems set the structure for development. However, depending on the knowledge level of the user, this lack of flexibility is not necessarily a drawback.

The survey shows that a mix of open source and proprietary methods are used in both industry and academia, as seen in Fig. 3. The patented HALCON is used mostly in industry, whereas Matlab is often used in academia. It can also be observed that the two open source solutions are used in both industry and academia. Additionally, our data reveal that 13 out of

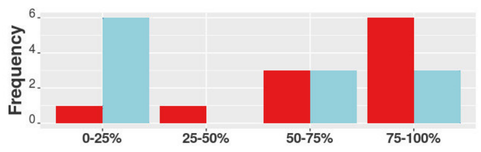

Figure 6: How often is your vision system a part of a robotics setup.

23 parties use more than one type of software.

Selection of Appropriate Parameters; Fig. 5(E): The methods discussed in Section 2 are seldom used directly out of the box. There exist many parameters to select, and the correct configuration can be crucial for the algorithm to work. These parameters can both be constraints of the algorithms search space, but can also be settings unique to the algorithm.

For example with the SIFT (Lowe, 2004) algorithm, crucial parameters are the number of octaves and the sigma value. It requires quite some knowledge about these parameters to judge what settings are required. Another example is the method ”find_shape_model_3d” (MvTec, a, Page 155) from HALCON. Even though HALCON simplifies the installation procedure by reducing the number of possible parameters, one still needs to determine the possible poses and adjust the remaining parameters. These parameters are ”MinScore” and ”Greediness” which can be used to adjust between accuracy and processing speed. Although these parameters seem more straightforward and intuitive than in the case of SIFT, they still require adjustment to ensure good results.

Optimization; Fig. 5(F and G): When one has gone through the steps in Fig. 5(A, B, C, D, and E), the system can be evaluated. For that, the success criteria of the setup needs to be defined.

A simple success criterion is that the system should be able to correctly recognize all instances of an object at high precision in real time. However, in cases with complicated workspaces, this is rarely possible. Objects could be blocking each other, unfavorable lighting conditions, random clutter and noise could make detection almost impossible. Fortunately though, there are cases where a perfect detection of every object is not required. For some applications only 95% precision may be sufficient, or one only needs to detect one instance at a time from a set of instances. These are all factors that determine if the system is usable.

When all the steps have been completed, and the system has been tested, it rarely performs as desired. Testing on actual data often reveals a need to adjust

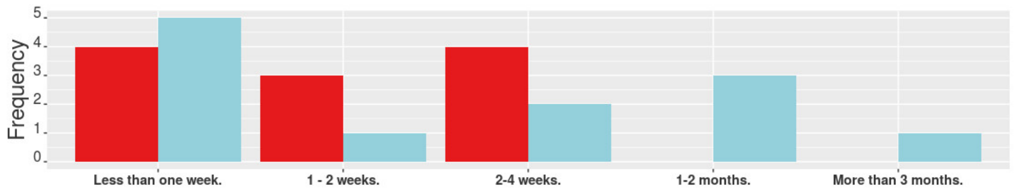

Figure 7: Time spent to complete a pose estimation task.

parameters. This can either be physical constraints that need adjustment or that the algorithm does not perform well enough and needs correction.

Depending on the severity of the error, the developer might need to go back further and further in the process. Small errors can be corrected by adjusting parameters Fig. 5(E), whereas larger errors could result in the need for a new algorithm, Fig. 5(D). One could even end up realizing that the sensor system is not able to solve the problem, Fig. 5(C), or maybe the problem needs to be redefined, Fig. 5(B).

4 REQUIREMENTS FOR A DIRECT USE IN SMES

As seen in the previous section, the user needs to take a lot of decisions when setting up a pose estimation system. For all of these choices, there are additional parameters and adjustments, making the design process cumbersome. Implementing these algorithms takes time, testing takes time and making changes takes time.

Fig. 7 is a visualization of the time spent on a pose estimation task. The median time spent on a task is 1–2 weeks for people employed at industry and 2–4 weeks in academia. This indicates a very long break-even time, which is a significant obstacle for SMEs to apply vision solutions. There is thus a clear need to bring down this development time.

To decrease the setup time of pose estimation algorithms a higher level of usability is required. Several aspects define the complexity of a solution: integration, algorithms, and setup. By these complexities, the usability of different software packages can be segmented into different levels. A representation of usability levels is shown in Fig. 8. An example could be HALCON, although widely applied, it is at best usable by integrators but not by any person without a solid engineering education. This is because functions need to be implemented and their parameters need adjustment, which presupposes a basic understanding of the underlying algorithms. To further facilitate and increase the application of pose estimation in industry (and in particular SMEs), it would be necessary to decrease the complexity and the average time used for setting up pose estimation systems.

We will now look into the obstacles and required means for a system, where an industry user with some basic education can set up a pose estimation.

Integration: When creating the system, the software should allow for guidance in the many possible sensor choices and placements, and should, of course, integrate with the hardware easily. This could be achieved by simulating the sensor itself and the sensor setup and optimizing. First steps in this direction are done in, e.g., (Jørgensen et al., 2017).

Likewise, there should be a simple integration of the found poses to the robot control system. This will not only save initial setup time but also allow for much easier testing of the system.

Algorithms: There are several ways the application of the algorithm can be facilitated: First, the system would need algorithms to automatically adapt its internal parameters to solve the task at hand robustly. Second, the selection of parameters should be as simple as possible. A condensation of the parameters could simplify this to fine tuning between different aspects, e.g., speed vs. recall. Another approach is to combine descriptors to cover a larger range of problems and avoid the need to test every situation (Buch et al., 2016). Third, there exist good examples of using simulation using either traditional vision methods (Jørgensen et al., 2017) or neural networks (Rad and Lepetit, 2017).

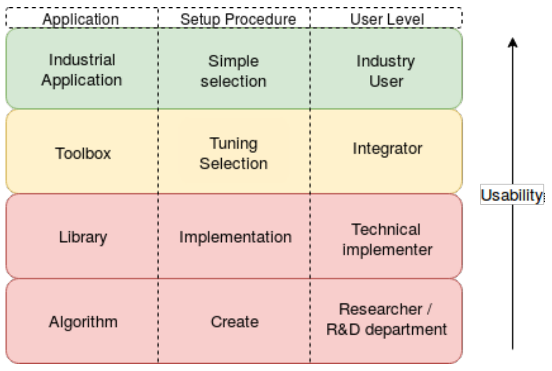

Figure 8: Four scales of usability for implementations of pose estimation system. The top one being the simplest to implement for the user.

Installation Procedure: With knowledge of the different parts of the system, a program would be able to guide the user through an installation. The user should select various aspects of the pose estimation in an intuitive way, potentially selecting on the screen where the object is to be detected, whether a detection is correct and so forth. The system should also be flexible enough to allow for easily adapting to small changes in objects variants instead of starting the process from scratch each time. Ideally, a database would store already found solutions intelligently and would match an existing task to earlier found solutions. This gives a good starting point for a given problem.

5 CONCLUSION

We have shown an analysis of the current situation of the applications of pose estimation in industry and academia. Our results show that the average setup time for pose estimation systems in the industry is 1–2 weeks, and 2–3 weeks in academia. Despite many improvements, there still exist many obstacles before pose estimation can be installed with a reasonable setup time. This currently limits the application of vision systems to large production sizes.

The results indicate that in addition to the algorithmic problems of pose estimation, the actual implementation of the system is still a challenge. In section 4, we proposed a scaling of the usability of vision algorithms. These levels indicate how much time and expertise it takes to set up a pose estimation system. At the top level, the system can be quickly set up by a non-expert, which is currently not possible.

To achieve this level of usability, the parameters of vision algorithms needs to be condensed, both to reduce the number of parameters, but also to make them more intuitive. There is also a need for a more efficient framework for setting up pose estimation systems, guiding the user through the complete setup from image acquisition to the final position. Here the simulation of sensors and systems before setting up costly hardware solutions could play an important role.

This paper shows that when evaluating the quality of pose estimation systems, the aspect of setup time might be an important criterion besides the percentage of recognized objects and the pose accuracy.

ACKNOWLEDGEMENTS

This work has been supported by the H2020 project ReconCell (H2020-FoF-680431).

REFERENCES

Adaptive-Vision. Adaptive vision. https://www.adaptive-vision.com/en/. Accessed: 2017-08-17.

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. Ieee.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.

Buch, A. G., Petersen, H. G., and Krüger, N. (2016). Local shape feature fusion for improved matching, pose estimation and 3d object recognition. *SpringerPlus*, 5(1):297.

Cognex. Cognex vidi suite. http://www.cognex.com/products/deep-learning/dl-software/?id=19178&langtype=1033. Accessed: 2017-09-15.

Collet, A., Martinez, M., and Srinivasa, S. S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306.

Drost, B., Ulrich, M., Navab, N., and Ilic, S. (2010). Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005. Ieee.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Gossow, D., Weikersdorfer, D., and Beetz, M. (2012). Distinctive texture features from perspective-invariant keypoints. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2764–2767. IEEE.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., and Wan, J. (2014). 3d object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287.

Gupta, S., Arbeláez, P., Girshick, R., and Malik, J. (2015). Aligning 3d models to rgb-d images of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740.

Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012). Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888.

Itseez (2015). Open source computer vision library. https://github.com/itseez/opencv.

Johnson, A. E. and Hebert, M. (1998). Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16(9):635–651.

Jørgensen, T. B., Iversen, T. M., Lindvig, A. P., Schlette, C., Kraft, D., Savarimuthu, T. R., Roßmann, J., and Krüger, N. (2017). Simulation-based optimization of a pose estimation system with application to detection of partially filled trays. *submitted to iros Vancouver september 24-28, 2017.*

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Krüger, N., Ude, A., Petersen, H. G., Nemec, B., Ellekilde, L.-P., Savarimuthu, T. R., Rytz, J. A., Fischer, K., Buch, A. G., Kraft, D., et al. (2014). Technologies for the fast set-up of automated assembly processes. *KI-Künstliche Intelligenz*, 28(4):305–313.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A scalable tree-based approach for joint object and pose recognition. In *AAAI*.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289.

Lewis, J. P. (1995). Fast template matching. In *Vision interface*, volume 95, pages 15–19.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Matlab (2016). Matlab and computer vision toolbox release 2016b.

Mian, A. S., Bennamoun, M., and Owens, R. (2006). Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1584–1601.

Miksik, O. and Mikolajczyk, K. (2012). Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684. IEEE.

MvTec. Halcon solution guide. http://download.mvtec.com/halcon-12.0-solution-guide-iii-c-3d-vision.pdf. Accessed: 2017-06-10.

MvTec. Mvtec. http://www.mvtec.com. Accessed: 2017-08-16.

Rad, M. and Lepetit, V. (2017). Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. *arXiv preprint arXiv:1703.10896*.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Sørensen, S. Y., Østergaard, S. M. F., and Pedersen, C. (2015). Robotter i global kamp.

Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *European Conference on Computer Vision*, pages 356–369. Springer.

Tombari, F., Salti, S., and Di Stefano, L. (2011). A combined texture-shape descriptor for enhanced 3d feature matching. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 809–812. IEEE.

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10.