| | |
|---|---|
| Title: | A Reconfigurable robot workCell for fast set-up of automated assembly processes in SMEs |
| Acronym: | **ReconCell** |
| Type of Action: | Innovation Action |
| Contract Number: | 680431 |
| Starting Date: | 1-11-2015 |
| Ending Date: | 31-10-2018 |

| | |
|---|---|
| Deliverable Number: | D3.2 |
| Deliverable Title: | Workcell monitoring: methods and evaluation |
| Type (Public, Restricted, Confidential): | CO |
| Authors : | T. Ivanovska, M. Tamosiunaite, F. Hagelskjaer, R. Bevec, A. Ude, N. Krüger, and F. Wörgötter |
| Contributing Partners: | UGOE, SDU, JSI |

| | |
|---|---|
| Estimated Date of Delivery to the EC: | 31-10-2017 |
| Actual Date of Delivery to the EC: | 15-11-2017 |

# Contents

# 1 Executive Summary

The deliverable presents the monitoring module of ReconCell, which is is responsible for pose estimation and inspection before and during the assembly process, respectively. This monitoring module detects certain types of errors and notifies the robot to stop or continue the execution.

The work on the vision module is a collaborative project of UGOE, SDU, and JSI. SDU is designing a pose estimation framework, which detects the availability and the pose of the details in the tray before the assembly process starts. UGOE is developing a visual inspection part, which detects the errors during the assembly process. Both submodules are then integrated into the general ROS-based ReconCell framework by JSI.

As ReconCell should be easily reconfigured, the vision system should enable an easy and fast setup. This aspect was at the center of attention when SDU was developing the pose estimation submodule. The requirement was set that the user should not be asked to adjust parameters. However, the user has to select one among a set of stable poses. Through template matching the system suggests a set of possible stable poses and the user has to select the desired stable pose. The template matching, however, requires no tuning to detect the object in a new image.

In this deliverable, the monitoring tasks that have been proposed by the ELVEZ and the LogicData companies are presented and analysed by UGOE. A camera-in-hand hardware configuration is chosen for the visual inspection, taking into account the complexity provided by the general cell setup and use case tasks as well as the overall cost.

Thereafter, the choice of the software package is justified. We analyze several available commercial and free frameworks and libraries. It appears that the customized solutions, which are based on industrial image processing frameworks, cannot be naturally integrated into the robot operating system (ROS), and some work-around solutions would be rather inefficient in terms of execution time and the overall cycle time, respectively.

Therefore, we develop a visual inspection system which is not based on any of the existing commercial frameworks. The system is designed to monitor the assembly process and detect errors, which can appear during the execution within the reconfigurable work cell. The software system is based on the ROS and OpenCV frameworks. The algorithmic pipelines have been pre-programmed for the ELVEZ and LogicData monitoring problems. Our system is completely integrated into the ROS framework, which allows for the real-time processing. The evaluation demonstrates the efficiency of the proposed hardware components and software systems.

# 2 Introduction

The aim of the project is to design a robotic work cell, which is highly reconfigurable in terms of hardware and software. The hardware elements of the workcell, both those selected off-the-shelf and those developed specifically for the system, allow for fast cell setup and reconfiguration, while the software aims to provide a modular, robot-independent, Robot Operating System (ROS) based programming environment [8].A vital part of the system is the monitoring module, which includes both a pose estimation submodule that localizes the details in the tray for grasping by the robot and the visual inspection submodule, which detects errors during the assembly process. The visual inspection system must be user friendly, efficient, and fully integrated into the general framework.

SDU is responsible for pose estimation of objects in a tray and detection of the correct objects are present before the assembly process starts. There exist many different systems for pose estimation, both commercial (e. g., Halcon [2]) and open source (OpenCV [11]). These systems themselves have many different solutions for pose estimation, both 2D and 3D. For the initial pose estimation in the ELVEZ case, a simple 2D approach using template matching was selected. The 2D approach was selected both because a color image can be easier to understand from the point of the user, compared to a point cloud and the camera position is less dependent on the object. For a 2D approach the camera is often best positioned above the scene, as this gives the least perspective, whereas a depth cameras best position can very depending on the object. 3D approaches will be further explored in the ReconCell project. t was decided to use template matching as the approach as the objects for industrial cases seldom contain distinct 2D texture, thus precluding a feature-based approach. The implementation for the template matching is the LINEMOD [6] algorithm implemented in OpenCV. This library was chosen both because it keeps the overall cost down by providing an open source solution, but also because the fact that it is open source software enables new pose estimation solutions, that will be developed, to be integrated.

UGOE is responsible for the monitoring task with visual quality control and error detection. The errors must be detected with vision components during the assembly process, and the whole system shall react accordingly. For instance, if the detail is damaged, the robot must stop the assembly process, remove the detail, and grasp a new one.

A common software for an industrial monitoring problem would be a customized solution based on such commercial packages as Halcon [2] or Cognex [12] frameworks. These frameworks are powerful and contain multiple efficient algorithms for pattern recognition and image analysis. Another possible option is to use other commercial frameworks (Matlab [4]) or free libraries such as OpenCV [1].

However, to our knowledge, there is no straightforward way to connect the ROS [8] interface with such frameworks as Halcon or Cognex software. Moreover, the use of a commercial software would increase the overall cost of the robot work cell. This leverages the use of the free and open source systems as ROS [8] in combination with the OpenCV library [1].

Therefore, we proposed a novel system, based on OpenCV and ROS libraries, which is designed for monitoring tasks in a robot workcell.

# 3 Tasks

## 3.1 Pose Estimation

ReconCell will deal with structured and semi-structured environments. Objects of interest in the assembly processes need to be localized and, specifically, for one application we had to determine the number of objects left in a tray to monitor the progress of the assembly. To avoid modifying the environment, a passive vision sensor is desired for this task.

Before the assembly process starts, the location and presence of the details on the tray is evaluated using template matching. The algorithm locates the position of the tray and whether objects are present in the tray. It then sends the position of the current requested object if it is present. If the object is missing, it shows an error until the tray is refilled. A new detection is then performed and the position of the object to be removed from the tray is sent further down the processing pipeline.

## 3.2 Monitoring

In general, there are two types of quality control tasks: binary decision and measurement. In the binary task, the algorithm needs to make a decision, for instance, if a detail is damaged or incorrectly placed. The measurement task requires us to perform detailed estimations such as finding the distance between two parts, measuring the height of a screw, calculating the angle of rotation of a part, etc.

Whereas the first task can be implemented using a camera, which has been only interally calibrated to produce undistorted images, the second task requires an extrinsic calibration [5] to obtain the measurement results in physical units, for instance, in mm.

Here we consider, among other things, the following quality control tasks:

1. Is a certain part of the housing damaged? (type 1)
2. Is the screw properly placed? (type 1)
3. Is the LWR drive properly placed? (type 1)
4. Is the light bulb holder properly placed? (type 1)
5. What is the angle of rotation of a hexagonal screw? (type 2)
6. What is the distance between two details? (type 2)
7. What is the orientation of the mounting holes? (type 2)
8. What is the height of the screw? (type 2)

# 4 Pose Estimation Solution

Before the assembly process starts, the location and presence of any details in the tray are evaluated. Using a single camera placed above the workspace, a template matching is performed. The algorithm locates the position of the tray and whether objects of interest are present in the tray. If the requested object is present, the position is returned to facilitate grasping, otherwise an error is issued until the tray is refilled and presented to the camera again. A new detection is then performed and the position of the object is determined again.

## 4.1 Hardware

The pose estimation is performed using a BASLER ACA1300-60GM camera with a Edmund Optics CFFL F1.4 f25 mm lens. The combination of a $1280 \times 1024$ pixel resolution and a 25 mm lens gives a focal length of 852 pixels. The objects are placed in a tray which is placed on the trolley. The camera is placed 1.5 m meters from the scene with its view direction orthogonal to the upper surface of the trolley. This gives a resolution of 1.7 mm per pixel. The trolley and the camera are shown in Figure 1.

Figure 1: The camera (left) and the trolley with objects (right) for the ELVEZ case ( and the chessboard used for defining the plane ).

The requirements for the plane detection, described in the following section, is a normal calibration plate containing a chessboard pattern with known dimensions.

## 4.2 Software

The pose estimation is based on template matching of a known CAD model. Using a CAD model makes it much easier to set up the system and create the templates. It is expected that companies have CAD models of their products available, and many of the other components in ReconCell already require a CAD model. Therefore, we consider the availability of CAD drawings to be a reasonable requirement for the pose estimation component also.

The objects used in the Elvez case, as well as any other objects used with this detection method, are expected to be placed on a planar surface, i.e. a table, which allows us to perform easy and user-friendly extrinsic calibration using a chessboard. Having acquired a simple setup

the last requirement for the pose estimation component, i.e. to be able to locate objects in 3D, can be performed.

By using geometric information from the CAD model, we can create a detection system, which is easy to set up. For many objects, so-called stable poses can be calculated. These are orientations of the object at which the object can be expected to maintain a stable placement on the table. With our system, the user can inspect the automatically generated stable poses and select the subset of stable poses that the system will encounter in the current scenario, thus constraining the search performed by the vision system. Several examples of stables poses of the tray are shown in Figure 2.

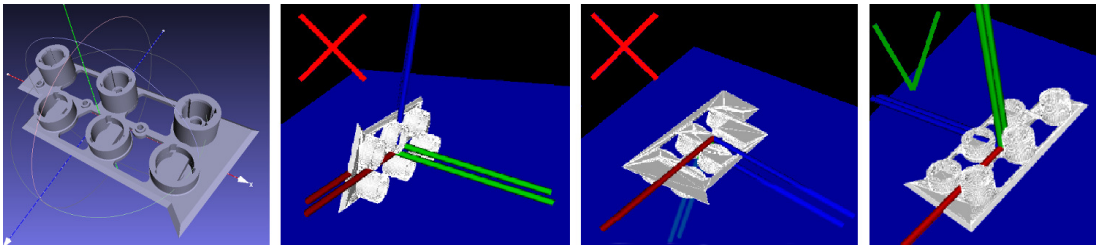

Figure 2: Left: CAD model of the tray used for Elvez. The following three images are examples of different stable poses for the tray, i.e. on the side, flipped and facing upwards. The object can be placed in any of these poses and will keep standing. As the red X's and the green tick indicate the final pose was selected for the Elvez use-case.

As we are doing template matching we want to create templates of the objects as they will appear in the camera. To create the templates we need the table plane a calibration plate with a known pattern is used. By placing it on the table, the coordinates are calculated and the templates for matching are created with a virtual camera at the same distance away from the tray as the table (see Figure 3).

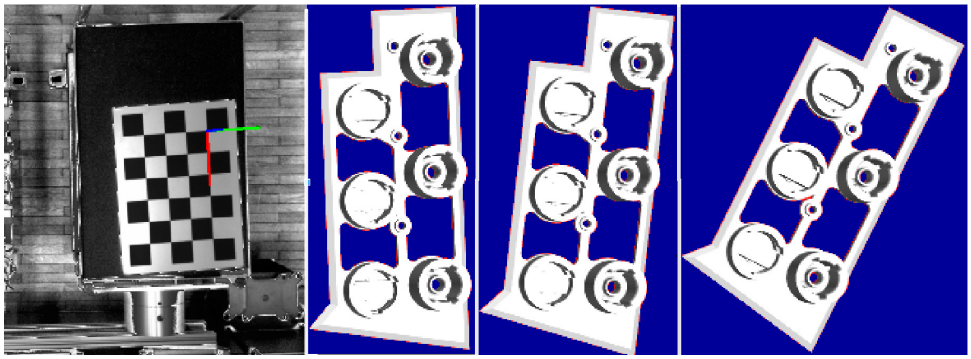

Figure 3: Chessboard placed on the trolley overlaid with the 3D coordinate system of the detected plane. To the right are samples of generated templates for the tray, oriented using the stable pose computed in the previous stage.

Using the LINEMOD [6] algorithm and the templates generated from the CAD drawing of the object, we can efficiently find the the 3D position and orientation of the object. We have additionally implemented an iterative procedure to further refine the pose estimates produced by LINEMOD to get object poses that are better suited for grasping. And example of our detections is shown in Figure 4

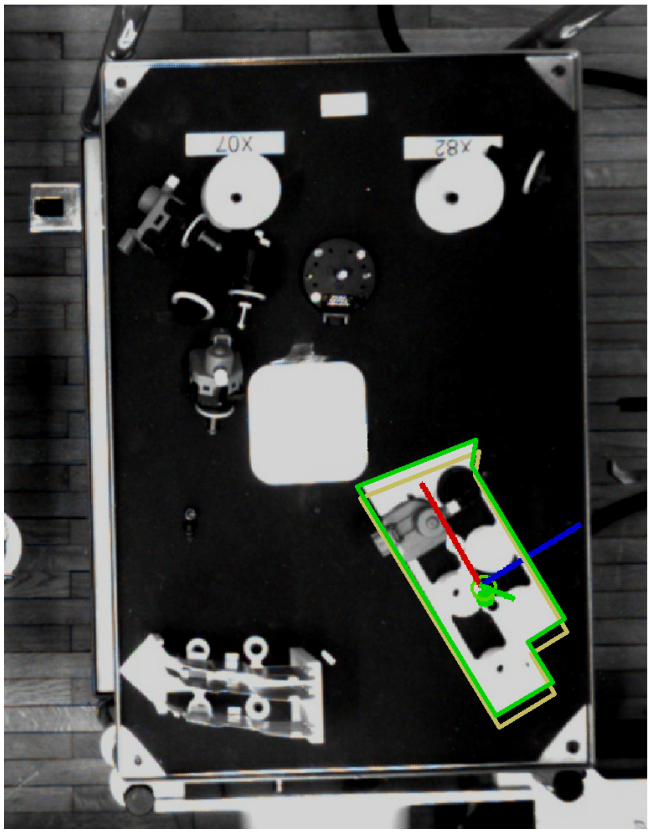

Figure 4: An example of a template-based detection is shown in yellow. The result after applying our refinement method is shown in green.

In the Elvez use case, the detected parts are trays, which contain the actual objects necessary for grasping. For all these objects, the relative position in the tray is known and a CAD model is available. Thus, we can create a virtual template of the object's position in the real scene from the already found 3D position of the tray. To detect whether the object is actually present in the real scene, a matching is performed using the generated template and the real image. This procedure is demonstrated in Figure 5.

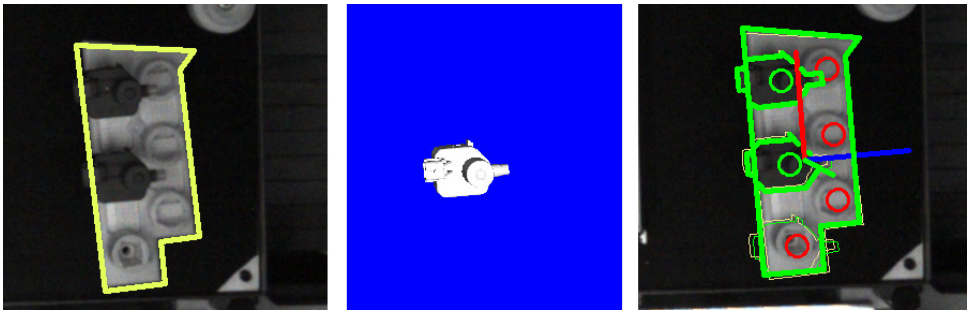

Figure 5: From left to right: Initial detection of tray, virtual template of an object in its expected position in the tray and actual detection of objects in the real image.

As a final note about the pose estimation component, we are also working on using machine learning techniques to discriminate between when an object is present in the tray or not. Figure 6 shows some of the images from which we are training a classifier to distinguish between

empty or non-empty tray regions. This is ongoing work, but so far it has shown promising results.

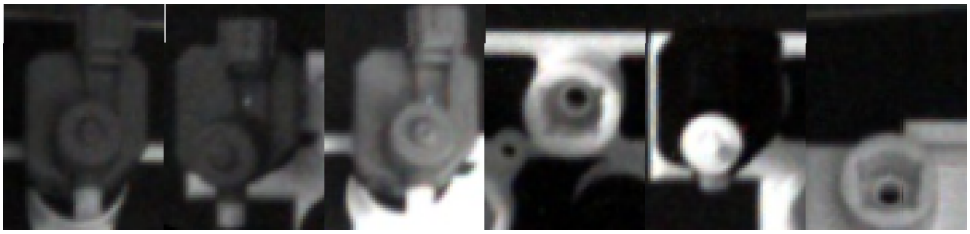

Figure 6: Example of different test images for detecting the presence of objects in the tray. The first three from the left are positives, i.e. with objects present in the tray. The last three are, respectively, empty region, region with a wrong object and again an empty region.

# 5 Monitoring Solution

Here we present the hardware components as well as the software components, which have been applied for the inspection module.

## 5.1 Hardware

The monitoring of the execution and detection of errors is performed in the workcell using a 2D color camera (Basler acA4600-7gc), which is mounted together with a light ring in the robot hand as shown in Figure 7. This high resolution industrial camera produces 14MP images and has been selected to be able to catch even minor detail damages and assembly inaccuracies. However, it has to be noted, that since the frame rate of the camera is only 7 frames per second (fps), it is practically only possible to capture still images, when the robot is not moving.

Then, the user defines critical measurement points: where the objects, which need to be measured, will be located at different points in time during the assembly process and what poses of the robot arm are needed to make the measurements. The camera plane, i. e., the plane that is perpendicular to the camera centerline, should be located parallel to the object plane.

Figure 7: The industrial camera with a light ring mounted on a hand of the UR-10 robot. Left: a general view. Right: a close-up view.

## 5.2 Software

We have developed a framework, based on OpenCV [1] using ROS interface [8], C++, and Qt [13].

The image data stream is published through the ROS module for the Basler camera. As soon as the request for monitoring arrives to the Monitoring ROS module, it collects the data from the camera, executes the corresponding processing, and returns the status of the operation (a boolean value, indicating, if the detail is damaged or not). Additionally, a graphical user interface is developed to demonstrate the monitoring routine.

### 5.2.1 The processing pipeline

The processing pipeline consists of two parts.

First, ideal or template images are acquired (cf. Figure 8) and the corresponding regions of interest (ROIs) are predefined by the user and stored in a specific location. After the templates are defined, the actual processing of new image data can be executed.

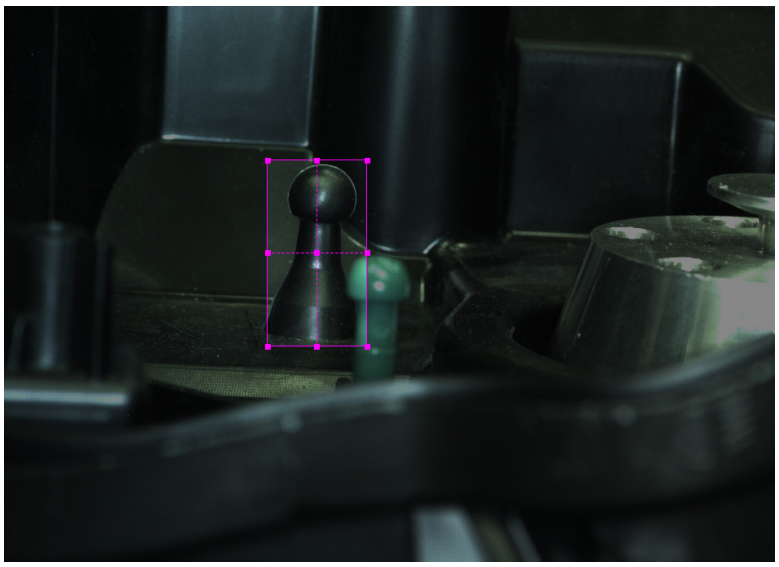

Figure 8: The process of template selection.

Second, the template ROI is detected in the newly acquired image. The search of the template starts within a neighborhood of the ROI location in the ideal image. The search is based on the assumption that the ideal template and the actual image data are always acquired under very similar conditions, namely, the same controlled lighting and the pose. The template image is overlaid with the corresponding patches of the original image and in each point in a certain neighborhood of the template location a normalized correlation coefficient is computed. The location of the global maximum is considered as the template location (cf. Figure 9).

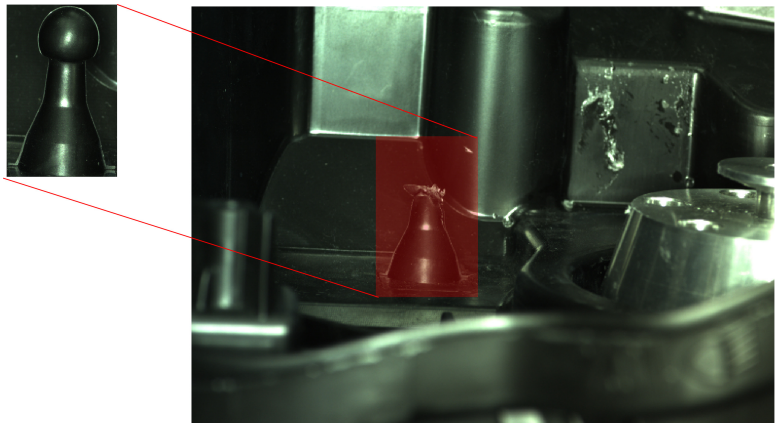

Figure 9: The process of template detection is schematically shown. The template image (left) is sought in the acquired image (right), and the region that is the most similar to the template image is marked red.

Finally, the detected ROI of the same size as the template image is extracted from the acquired image and passed for further processing.

Third, the subimage preprocessing is executed. In the preprocessing step, we perform illumination correction and denoising [7]. It is crucial to apply these techniques not on a full acquired image, but on the extracted subimage to reduce the computational cost and the overall execution time. The subimage is converted to the L*a*b color space [9], which is designed to approximate human vision. The luminance channel (L) is extracted and the adaptive histogram equalization [10] is applied to it. Thereafter, the image is smoothed with median filter to preserve the edges. Fourth, the use case-specific checks are executed. Usually, they include a segmentation step, for example, using color global or local, automatic or manual thresholding. For the automatic thresholding, the Otsu thresholding method [14] is applied. Thereafter, connected components [7] are checked, and the component of interest is found and analyzed.

### 5.2.2 Framework modes

The framework has the three following modes:

- Template selection (online or offline);
- Offline processing pipeline testing;
- Online monitoring.

Note that during the offline acquisition, the image is taken from the hard drive. In the online acquisition, the image is obtained from the camera through the ROS interface.

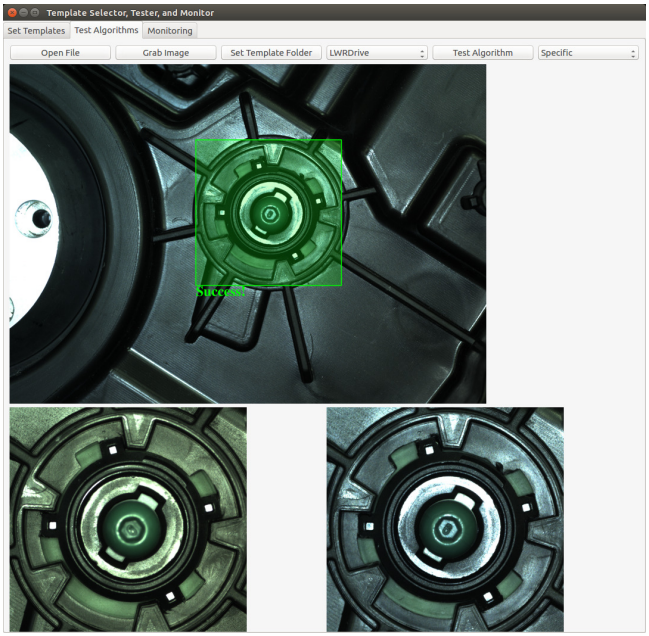


Figure 10: The detection result is positive.

After the template selection, the user can test the developed detection pipelines and check different parameter settings on an image, acquired from the camera directly or saved on a hard drive. An example of the offline testing mode is shown in Figure 10.

In the current version of our framework, the processing pipelines are pre-programmed. In the next version the user will be able to generate and save the processing pipelines.

# 6 Evaluation of the Pose Estimation Module

The aim of the pose estimation for ReconCell is to ensure ease-of-use and simple setup. One of the most user-friendly commercial systems on the market for this is the Halcon library [2]. When setting up an equivalent template matching in Halcon, even if using initial preset parameters, the user still needs to manually move the 3D model to the desired orientation and distance in the setup. This setup requires knowledge of Halcon and general skills with CAD models, limiting the scope of users.

In our solution, this setup has been reduced to placing a chessboard on the table and selecting the desired stable pose for the part to be localized. We believe this greatly reduces the setup complexity, allowing also less-expert users to set up new detections when the objects change.

# 7 Evaluation of the Monitoring Module

## 7.1 Hardware components

Today the market of industrial imaging offers multiple 2D and 3D cameras. For instance, such companies as "The Imaging Source" [1] or Basler [2] present industrial cameras, which can be successfully applied for the monitoring task. Apart from that, these manufacturers support a multi-platform application programming interface (API).

Moreover, there is a specialized ROS module for Basler cameras, which is naturally built in the general software architecture of the reconfigurable work cell [3].

We have selected a color high resolution Basler camera, which can only be used for acquisition of still images, i. e. the robot moves to a certain position, stops, and the image is taken. For such tasks as video tracking, this camera is not suitable.

For the selection of the appropriate lens, several parameters have been taken into account. The main ones are the object size, the working distance, and the sensor size. The sensor size is defined by the camera. Since the size of the imaged objects can be rather small, e. g. $10 \times 20\ mm^2$, we have selected a $f25$ mm lens with a working distance 200 mm. This working distance is realistic, taking into account the bulky robotic hand, fixtures, details, as well as light mounting.

## 7.2 Software components and algorithms

The general project architecture suggests to use the ROS framework. Therefore, we looked for a software package or library that can be integrated with ROS interface in a straightforward way.

To our knowledge, the processing using the well known software as Halcon or Cognex can only be organized using the procedure schematically shown in Figure 11. The image must be obtained from the camera using the ROS-based framework and saved on a hard drive. Thereafter, the image can be read and processed by a software package. The ROS node keeps listening to the specified folder for the results. As soon as the image processing is finished and saved, the ROS node reads the results and sends a notification to the robot.

This solution is rather time-consuming and inefficient. Hence, we have developed our own system, which is implemented as several ROS nodes. Our framework is based on OpenCV library, i. e., the main algorithm implementations are tested and optimized. Several algorithms are combined into processing pipelines, suitable for the example use case. The main idea of our software package is to use fast and efficient algorithms that do not require any prior training due to the fact that for each task in each use case the collection of a big amount of data seems infeasible.

Moreover, we observed that measurement problems are very different even for the same use case. Hence, only limited generalization with respect to processing algorithms is possible. Namely, one is not able with the same algorithm to measure the height of a metal screw and detect, if some other plastic detail is intact. The general evaluation pipeline consists of the following steps:

1. Template detection

[1] https://www.theimagingsource.de
[2] https://www.baslerweb.com

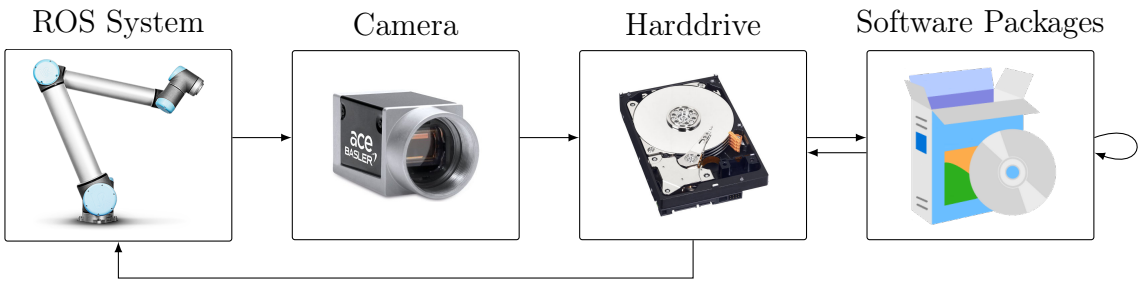


Figure 11: The image processing pipeline, when a software package is not straightforwardly combined with a ROS-based system.

2. ROI selection
3. ROI pre-processing (denoising and illumination correction)
4. Segmentation and object extraction
5. Object evaluation (distance or angle measurement, binary decision)

The first three steps are rather general, and we use them for all measurement problems. Of course, prior to measurements, the user has to set up the templates for each problem.

For the first step, the template matching algorithm provided by OpenCV is applied. We observed that only normed metrics (for instance, `CV_TM_CCORR_NORMED`) work reliably for our problems. This straightword template matching is applicable, when the assumption about the controlled lighting conditions holds. Moreover, such an approach is preferrable for textureless objects.

The latter two steps are task-dependent. For segmentation and object extraction we usually apply some prior knowledge about the object appearance, e. g., color, shape, and location on the image. Here, such algorithms as color clustering or thresholding, and shape analysis (Hough transformation, Harris corner detection, Connected component analysis) are applicable.

For each measurement problem, we collected test images, acquired in a test environment as well as the real images, obtained during the assembly procedure. The parameters were pre-selected and optimized for the implemented use case.

A new use case, i. e. with new measurement tasks, the algorithmic pipelines and their correspondent parameters would have to be again selected and optimized. However, we assume that a set of image processing tools provided by the OpenCV library, such as pattern matching, denoising, segmentation, and feature extraction algorithms would be fully sufficient for a monitoring task.

# 8 Conclusions

In total, the vision system in ReconCell is presented as two frameworks, namely, the pose estimation and the visual inspection, which monitor the cell before and during the assembly process. These frameworks have been developed by SDU and UGOE, respectively.

Here, the selection of the hardware components as well as the software tools is presented, thoroughly described and analysed. It was shown that the systems are efficient, easy-to-use, fully integrated with the ROS framework and, therefore, are fully suitable for a highly reconfigurable robot work cell.

SDU and UGOE will extend their frameworks to completely fulfill the requirements until the end of the project. The extensions will include further methods, designed for the third use case as well as general-purpose techniques, which would allow for new (unknown) use cases to be addressedok . UGOE will also continue investigation of the tracking problem as agreed with the reviewers during the review meeting.

# References


[1] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 25.11 (2000), pp. 120–123.

[2] W. Eckstein and C. Steger. "The Halcon vision system: an example for flexible software architecture". In: *Proceedings of 3rd Japanese Conference on Practical Applications of Real-Time Image Processing*. 1999, pp. 18–23.

[3] T. Gašpar et al. "Rapid hardware and software reconfiguration in a robotic workcell". In: *18th International Conference on Advanced Robotics (ICAR)*. Hong Kong, 2017, pp. 229–236.

[4] D. C. Hanselman and B. Littlefield. *Mastering matlab 7*. Pearson/Prentice Hall, 2005.

[5] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[6] S. Hinterstoisser et al. "Gradient response maps for real-time detection of textureless objects". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (2012), pp. 876–888.

[7] B. R. Masters, R. C. Gonzalez, and R. Woods. "Digital image processing". In: *Journal of biomedical optics* 14.2 (2009), p. 029901.

[8] M. Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Kobe. 2009.

[9] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. "Color transfer between images". In: *IEEE Computer graphics and applications* 21.5 (2001), pp. 34–41.

[10] A. M. Reza. "Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement". In: *The Journal of VLSI Signal Processing* 38.1 (2004), pp. 35–44.

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

[12] J. R. Scola, V. N. Ruzhitsky, and L. D. Jacobson. *Machine vision system for object feature analysis and validation based on multiple object images*. US Patent 6,175,644. 2001.

[13] M. Summerfield. *Advanced Qt Programming: Creating Great Software with C++ and Qt 4*. Pearson Education, 2010.

[14] J. Zhang and J. Hu. "Image segmentation based on 2D Otsu method with histogram analysis". In: *International Conference on Computer Science and Software Engineering (CSSE)*. IEEE. 2008, pp. 105–108.