TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Fusing multiple Kinects to survey shared Human-Robot-Workspaces

Claus Lenz, Markus Grimm, Thorsten Röder, Alois Knoll

TUM-I1214

Fusing multiple Kinects to survey shared Human-Robot-Workspaces

Claus Lenz, Markus Grimm, Thorsten Röder, Alois Knoll
Robotics and Embedded Systems
Department of Informatics
Technische Universität München
{lenz,grimmm,roeder,knoll}@in.tum.de

Abstract— **Knowledge about the static and dynamic situation is essential for joint workspaces of human and robot. Surveilling the joint area helps to realize a collaboration and co-existence of human and robot. Therefore, we propose in this paper a system that redundantly monitors the workspace to perceive obstacles within the workspace including the human worker. We use multiple, distributed range sensors (i.e. Microsoft Kinect) and de-centrally pre-process the data from the sensors. These data sets are further processed and segment and cluster unknown objects. The gained geometric information can then be used in the robot controller to allow the system to react accordingly.**

I. INTRODUCTION

In today's industrial applications humans and robots are often strictly separated in space or time to preclude collisions that may have severe consequences for the human. Both research and industry are currently struggling to overcome such limitations, because a large variety of applications would benefit from a direct cooperation of humans and robots or the co-existence of humans and robots in the same workspace.

Due to the missing information about moving objects in the shared environment it is possible that collisions between the robot and other objects may occur. If such a collision affects the human, this may have serious consequences with usually severe injuries of the human worker. Therefore, when humans and robots should work together as a team, the safety of the human worker must be guaranteed all the time. One possibility to achieve this would be to use safe hardware, for example the DLR Light Weight Robot [1], that is able to detect contacts with other objects.

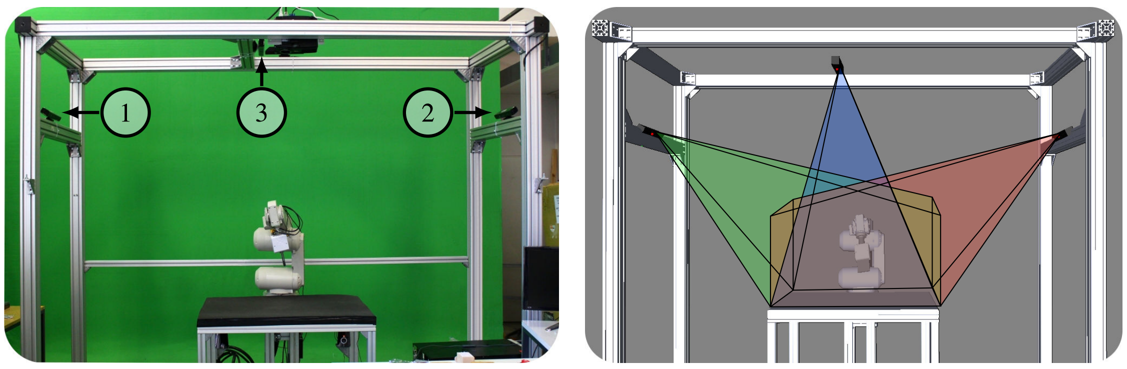
Additionally, without sensory equipment that perceives the environment, the cooperative teamwork between human and robot is only possible on a very low level, since the robot is not able to dynamically adapt to changes in the human worker's behavior or the environment. For efficient collaboration between humans and robots on a peer-to-peer level, it is necessary to observe the human's activity and collect information about the current situation. Based on this information the system can evaluate the actions of the human worker to coordinate its own tasks. Therefore, in order to enable safe and efficient human robot collaboration, the joint workspace needs to be perceived by several, distributed sensors like CCD cameras, range sensors or microphones. The information gained by these sensors builds up an internal representation of the working environment [2]. With the help of this environment model it is possible to avoid collisions between the robot and other objects in the shared space [3]. Aside from that it can be used to plan the next action steps of the robot according to the current situation, in order to reach an anticipatory behavior of the robot [4], [5].

The aim of this work is to support collaborative human-robot scenarios using multiple distributed range sensors to redundantly surveil the joint workspace. The system perceives its surroundings via the three dimensional data sets acquired by range sensors in order to build up an internal representation of the working environment. The gained information can then be used to avoid collisions between the robot and its surroundings. Since the robot is also acting in this space, we have introduced an adaptive suppression technique to cross out data points resulting from the robot. Our approach distributes the workload among different systems, fuses available views of the environment and detects unknown objects in the working area.

II. RELATED WORK

The approach presented in [6] uses three color cameras, each connected to a separate PC. The three PCs work parallel and determine the position of the human worker and the robot in the scene. This information (still in 2D) is then transferred to a fourth computer that creates a three dimensional model of the scene. Based on this perceived model of the environment and a geometrical model of the robot, the system calculates the spatial distance between the human worker and the robot and intervenes if this distance is too short. The disadvantage of this approach is that the extraction of the human worker in the images is based on skin color detection. Hence, the system can not operate properly if the worker wears, for example, long clothes or gloves. In addition to that, the system has no means to identify other obstacles in the environment.

In [7] a stereo camera system is used to reconstruct the scene in 3D. Further, 3D models of humans and robots are used as a basis to extract features of the single elements in the scene. Based on these features, historical data, and additional pre-configured rules, the system tries to predict the following state of elements in the scene. This information is used to detect potentially dangerous situations which are resolved by appropriate, pre-defined reactions of the robots. However, the system can only perceive obstacles that have been defined and modeled in advance. Thus, this approach can hardly cope with complex dynamic scenarios and unknown obstacles.



(a) The *JAHIR* set-up (b) Visualization of the *JAHIR* set-up

Fig. 1. **Positions of the Kinect sensors in the *JAHIR* set-up** - The Kinect devices 1-3 are installed on the scaffold surrounding the workbench (a). The devices are also integrated into the three-dimensional simulation of the set-up (b). The view areas of the devices that are of interest for the workspace surveillance are also highlighted in (b)

To detect all kinds of obstacles, range sensors (including PMD cameras [8]) are more appropriate, because geometric information is delivered directly in the three-dimensional space for each pixel of the depth camera.

A system, that is closely related to ours, is presented in [9], where also geometric scene data is filtered using known foreground objects and a suppression technique for the moving robot. Opposite to the usage of a single PMD camera with a low pixel resolution, we are using multiple high resolution Kinect cameras to redundantly cover the workspace while still remaining at high update rates.

The approach presented in this paper closely related to the system presented in [10], [11] where the shared workspace is monitored by multiple stationary color cameras and range sensors. The sensors are connected to multiple slave computers that pre-process the raw sensory data, detect object point clusters, compute convex hulls for each cluster, and synchronously transfer the data to a master computer. The master server fuses the acquired information and creates a 3D-model of the surroundings. This way it is possible to calculate the distance between the robot and any kind of obstacle. Our approach differs mainly in the following issues: we have integrated an automatic extrinsic and intrinsic calibration mechanism so that e.g. slightly moved cameras cause no problems. Additionally, we can seamlessly add more sensors and distribute the modules among the distributed computer structure. Further, due to the use of Kinect devices, we have a lot more data that needs to be analyzed which increases the density of the overall measurement.

III. BACKGROUND

In this paper, we use the *JAHIR* system as platform [12], [13]. This system has been designed as a generic robotic system to analyze and show a variety of concepts regarding collaboration aspects between human and robot. As depicted in Figure 1 (a), a standard position controlled industrial robot is placed on a working table. Human and robot can jointly use a workbench and partly share the same workspace. In this way, both human and robot have areas where they can work for their own and on the other side where both partners can work together. Hence, human and robot are brought closely together for a diversity of collaborative assembly applications.

In a shared workspace, collisions need to be avoided for static (for example the workbench) and dynamic (i.e. moving obstacles) objects. In order to surveil the entire shared area of human and robot in any situation, multiple sensing devices are required that capture the scene from different points of view. A good and redundant coverage from different directions is necessary to be able to gain information about the surrounding even if obstacles are occluded in one or more cameras.

As depicted in Figure 1, we have used in this paper three Kinect devices. To reach a good coverage of the work environment two Kinects are mounted left ① and right ② of the worktable, both at a height of 2 m on the cage surrounding the workspace. The third one ③ is fixed to a crossbar directly above the table in order to perceive the environment from a bird's perspective like it is visualized in Figure 1 (b).

It has turned out that this arrangement of the Kinect devices is a good choice for surveilling the jointly used workbench of human and robot, as most of the human's work area can be monitored without having occlusions in all cameras. This arrangement has been chosen heuristically. A mathematical solution to reach an optimal sensor placement is e.g. presented in [14].

Since the robot controller has a dynamic internal environment model with which distances can be computed, it can avoid collisions in a reactive way as presented in previous work [3]. Additionally, every sensing or processing unit can add, update or remove objects in the internal representation via a defined communication channel. Please refer to [3], [2] for further details.

IV. APPROACH

The surveillance component is supposed to be flexible, extendable and distributable. Hence, we have divided it into several components, that are connected via the network. All modules can be started distributed on different machines.

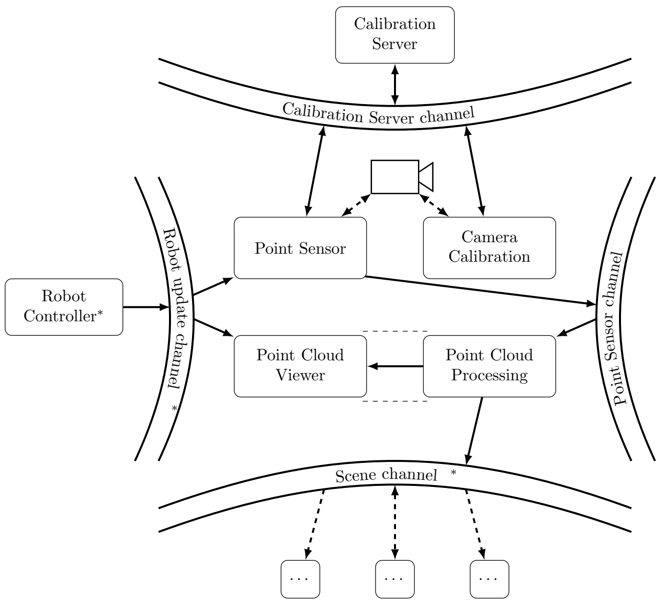


Fig. 2. Extract of the overall system architecture. The *Calibration Server* channel enables communication between the *Camera Calibration* tool and the centralized *Calibration Server*. Further, the *Point Sensor* communicates with the *Calibration Server* via this channel to request its calibration parameters. The acquired point clouds are interchanged between the *Point Sensor* modules and the *Point cloud processing* module via the *Point Sensor* channel. The *Point cloud processing* module broadcasts the detected objects via the *Scene* channel. Both the *Point Cloud Viewer* and the *Point Sensor* subscribe to the *Robot update* channel. The *Robot Controller* broadcasts the current joint positions of the robot via this channel

Figure 2 details the information flow between the modules (Calibration Server, Camera Calibration, Point Sensor, Point Cloud Viewer, Point Cloud Processing) and the other components.

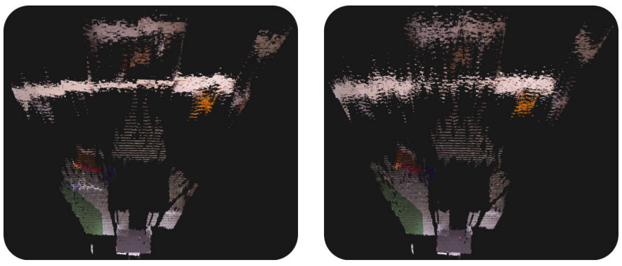
A. Automatic calibration of sensors

A *Calibration Server* acts as central entity for calculating and publishing the poses of the cameras in the scenario. It allows both intrinsic and extrinsic camera calibration for diverse calibration patterns (e.g. chessboard patterns, circle patterns, ring patterns, etc.) and different types of cameras. Supplementary, point clouds from different sensors are registered to reduce alignment errors caused by inaccuracies during the calibration process. The transformation between an initial and the registered point cloud of a sensor is used to update the sensor's extrinsic calibration.

Since the RGB and depth image streams provided by the *OpenNI* driver are pixel aligned, the Kinect devices can be easily extrinsically calibrated one of the integrated sensors (infrared sensor or RGB camera). In the set-up a regular chessboard pattern is used for extrinsic calibration, where one corner square contains a red circle to mark the world origin. At start-up of the camera modules, the camera can extrinsically calibrate themselves and find automatically their position within the world. Hence, it is inherently possible to attach and add more sensors to add more information and data sources.

B. Local sensor data pre-processing

The pre-processing of the raw sensory data of a single range sensor is implemented in the *Point Sensor* module using *OpenCL* with an extensive use of the graphics hardware. The main task here is to detect objects in the surroundings, which can form obstacles for the robot. At start up of the system a model of the static environment (*background model*) is computed by averaging multiple depth images along with the average standard deviation for each measurement point to obtain an estimate for the noise behavior of the sensor. This is especially of interest if multiple Kinects are used, because the measurement noise of the depth estimation increases when the projected patterns of multiple Kinects overlap (see Fig.3 for an example).



(a) One active Kinect (b) Three active Kinects

Fig. 3. **Usage of multiple overlapping Kinects** - Both point cloud data sets were acquired by one Kinect. Compared to (a), the point cloud in (b) contains more measurement noise due to the illumination of the scene with multiple Kinects

Additionally, using multiple Kinect devices the point cloud of a single sensor becomes less dense and contains fewer points. This is shown in Figure 4 where up to three Kinects are switched on. Despite the drop of about 15 % of points in one sensor, we get approximately 3×200.000 data points with 3 Kinects switched on, which is quite impressive.

For the determination of the dynamic environment the system then differentiates whether a point originates from an unknown object based on the *background model* and the average standard deviation. A point d_i in the depth image is defined to belong to the static environment if its value is within the deviation interval $background_i \pm \bar{\sigma}$.

The system uses bounding box tests for the determination

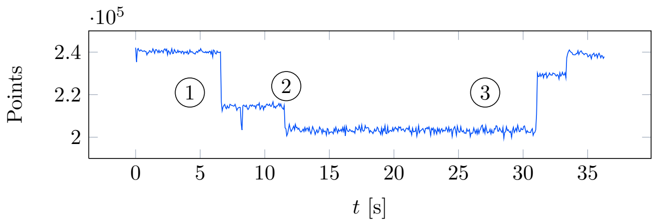


Fig. 4. Increasing image noise. The graph shows the progression of the number of acquired points of one Kinect device over time if none, one or two additional Kinects are turned on and illuminate the scene. One can see that if two additional sensors are up and running (3), the number of valid points is reduced by about 15 % compared to the original value (1)

Device	Empty scene	Robot active	Human active	Human and Robot active
Left	17.76	4.02	1752.4	1786.1
Right	23.87	16.17	3428.6	2678.1
Top	13.86	8.70	10141.9	8519.4

TABLE I

TRANSMITTED POINTS PER FRAME - THE TABLE SHOWS THE AVERAGED AMOUNT OF POINTS TRANSMITTED IN DIFFERENT SCENARIOS

of the dynamic environment. An offline created environment model contains the bounding boxes of the robot model and the model of the workbench. The bounding box of the workbench can be extended in each direction to cope with sensor noise. Additionally, the robot is also suppressed by applying bounding box test of the robot bodies. The bounding boxes of the robot are statically enlarged , because of protruding wires on the robot's links. This makes it possible to remove all points from the acquired point cloud data set that originate from the robot's surface or from the cables. This works fine as long as the robot remains in its position. If it moves, the bounding boxes of the robot are updated correctly, but the depth images are lagging behind due to the depth reconstruction within the Kinect device and thus not all points can be reliably filtered (see Figure 5).

By applying all filters to remove the known environment, the number of points that need to be broadcasted for further processing can be reduced to a minimum. Table I shows how many points are filtered by a *Point Sensor* instance, keeping in mind that the Kinect acquires more than 300.000 points per frame. The point transfer rates show that there is almost no data traffic if nothing happens in the workspace. If the human worker is active, the number of points rapidly increases but stays at a low percentage (about 3 %). It is also notable that fewer points are transmitted if only the robot is moving due to the dynamically enlarging bounding boxes.

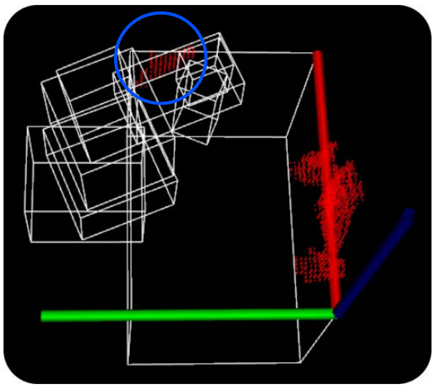


Fig. 5. **Camera delay** - The robot is moving downwards in the image. The bounding boxes of the robot are updated correctly in time but they do not fit with the information gained by the Kinect device as the camera introduces a short delay. This delay is caused by the internal processing of the camera

To resolve this problem, a mechanism has been integrated that dynamically enlarges the bounding boxes of the robot if it is moving.

C. Global data consolidation and clustering

The *Point Cloud Processing* module combines and processes the point clouds published by several *Point Sensors* in the distributed set-up using *PCL* [15]. On the basis of the fused point cloud, the object point clusters are extracted and for each detected object a convex hull is computed. Afterwards, the current object hulls are broadcasted via the *Scene* channel to integrate them into the global environment model. Additionally a *Point Cloud Viewer* for displaying both raw and processed point cloud data has been developed. It visualizes the point clouds broadcasted by the *Point Sensor*s, the extracted object point clusters and the convex hulls [16].

The point clouds are sampled down utilizing a fixed width octree structure. Since the resultant point clouds contain only the occupied voxel centroids, this approach affects the accuracy of the entire system: A point, whose distance to the centroid of its encapsulating cube is maximum, is mapped to the centroid. The point is shifted by ca. 0.65 cm in this worst-case situation. Hence, a computed convex hull can be smaller than the original object. But since the accuracy of the depth measurement is only about one centimeter, the impact of this technique on the overall system accuracy is only little.

The clustering method to find connected objects uses radius searches to group the points into clusters. We have used a distance threshold of 3 cm as a compromise between object point clusters being too small, and thus may approximate only a part of an object, or being too liberal and spacious. So it is advisable to choose this parameter generously. Note that if the radius is too small, an obstacle may be split into two clusters and the resultant convex hulls may not overlap.

In addition to that, an object point cluster must contain at least 30 points, otherwise it is ignored. A fused point cloud of all three Kinects in the set-up contains outliers which can artificially enlarge an object. The minimum cluster size takes this blur-effect into account. In a test scenario, the system was able to detect a cardboard box ($6{\times}10{\times}4$ cm) standing or lying on the workbench.

The computation of the convex hull for a given point set approximates the surface of the underlying sampled object. Since all sampled points are located on or inside the convex hull, this structure enables efficient collision tests. The convex hull approach additionally compensates missing data from object surfaces.

V. EVALUATION

The used system consists of three computers, each having a INTEL i7-2600 3.4 GHz processor, 16 GB RAM and a Nvidia GeForce GTX 560-1GB graphics card. Two of the Kinects are connected to the first PC, the third one is connected to the second PC, because each Kinect device need to be connected to a separate USB controller and the PCs used have only two. The *Point Cloud Processing* module are also executed on the first system. The third PC is used to control the robot and to broadcast the current joint positions.

A. *Data acquisition and pre-processing*

The workspace surveillance component perceives the environment through multiple range sensors. Each *Point Sensor* instance publishes the acquired information via the *Point Sensor* channel. The performance of a *Point Sensor* is limited upwards by the update rate of the used sensing device. The Kinect device acquires up to 30 frames per second. Since the computation time for a single frame depends on the amount of points which are in the inside of the observed area, the performance of the system (i.e. the frame rate) may vary over time, depending on the number and volume of the objects in the scene. In addition to that, as each device captures the scene from different viewing directions, the performance is also different for each *Point Sensor*.

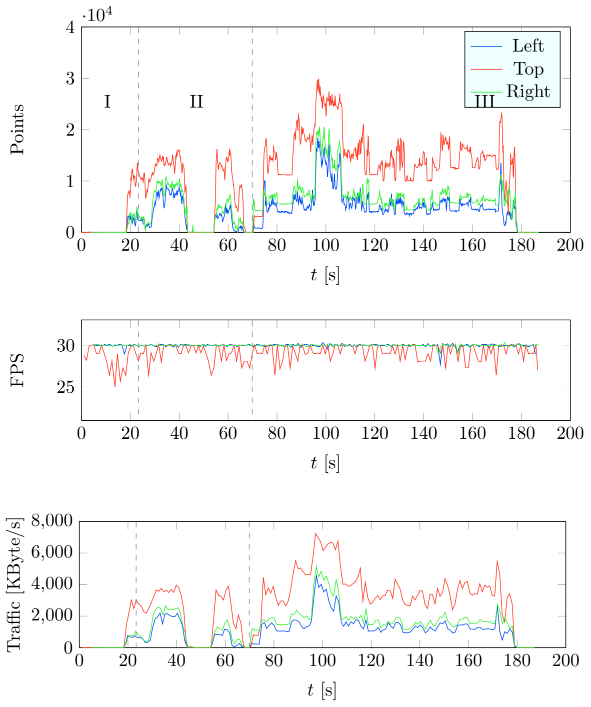


Fig. 6. **Data acquisition** - (a): Points per sensor over time. (b): Computed frame rates per sensor over time. (c): Transfer rates per second over time

The first graph in Figure 6 shows the number of points per frame and range sensor over time. During the first 18 seconds the workbench is empty and the worker is outside of the observed area (I). The following sequence (II, sec. 18 to 50) contains a short test if all sensors are up and running. After that, the worker performs an example assembly task where the robot fetches required parts and hands them over to the human worker (III). At about $t = 95\,\mathrm{s}$, the worker fetches a toolbox, searches for the right tool and puts the box away again.

The frame rates of the *Point Sensor*s are given in the second graph of Figure 6. The Kinects mounted left and right on the scaffold have almost constant at about 30 frames per second (Ø 29.9 FPS). The frame rate for the third Kinect varies more over time, but as the average is at about 29.0 FPS, it is also near the maximum attainable performance. The reason for this deviation is, that the third *Point Sensor* needs to publish the point clouds over the network while the other two communicate directly over a *localhost* connection.

The third graph in Figure 6 shows the progress of the network traffic over time in kilobytes per second. The network traffic is proportional to the number of published points. In the example scenario it is always below 10 MByte/s and therefore within the bounds of what is manageable by modern network cards.

B. *Point cloud processing*

As described previously, the task of the *Point Cloud Processing* module is to combine the most recent point clouds of the *Point Sensor*s, downsample the merged point cloud, extract the object point clouds and compute the convex hull for each cluster. The evaluation presented in this section is based on the example scenario which was introduced in the previous part.

As depicted in Figure 7, the large size of the merged point clouds would make it difficult to reach real time performance when searching for object point clusters. By constructing a downsampled version of a point cloud, the number of points is reduced(first and second graph). The frame rate of the system (third graph) represents the number of frames processed by the clustering of the *Point Cloud Processing* module. This includes downsampling the point cloud, extracting object point clusters and computing the convex hulls.

The performance of the clustering directly depends on the size of the acquired point clouds. In the presented scenario, the system was able to publish a new set of detected objects after 100 ms at the latest (i.e. in the case of having the worst update rates with 10 FPS at $t = 40\,\mathrm{s}$). In the periods where the human worker is actively operating in the monitored space, according to the measurements, there are at least 10.000 points per operating cycle that are processed by system. If there are more than 40.000 points per frame, the system's frame rate collapse to 10 to 15 frames per second, otherwise it is more or less constant at 30 FPS.

The frame rate also depends on the number and size of the object point clusters. This effect can be seen in the graphs for example at about $t = 148\,\mathrm{s}$: Although the number of points is comparable with the point cloud size at $t = 40\,\mathrm{s}$, the frame rate is only at around 10 FPS. The number of clusters makes the difference here (4 clusters compared to 2).

In the latter two cases, there is either a lot of activity or there are big obstacles within the workspace. In these cases, the robots' velocity should be reduced. Hence, with a reduced velocity, the reduced framerate can be compensated.

VI. Conclusion

In this paper we have presented an approach for monitoring of the joint workspace in human-industrial robot collaboration scenarios. The developed system uses multiple, distributed range sensors (MS Kinect) to perceive unknown objects and obstacles in the work area of the robot. To

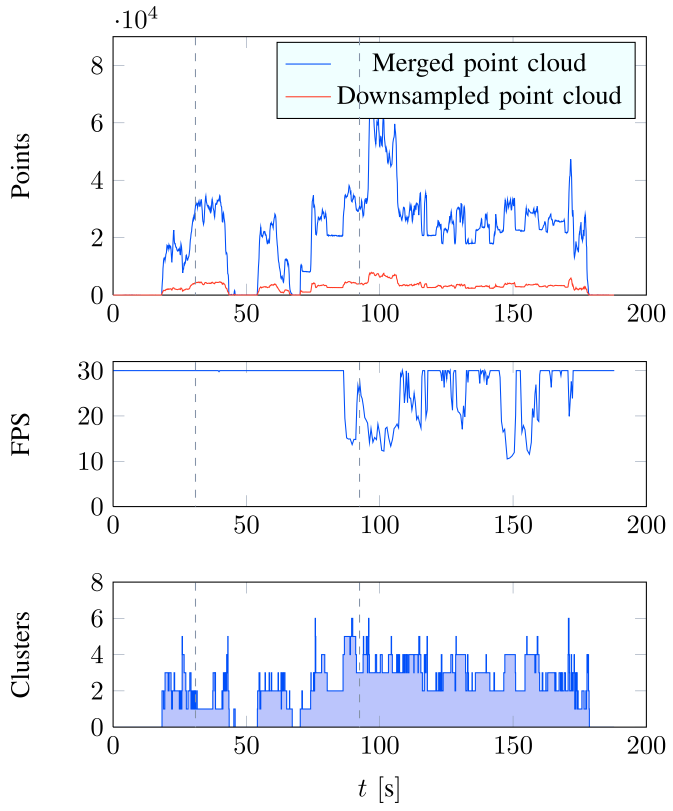


Fig. 7. **Point cloud processing** - From top to bottom: Point cloud sizes over time; Number of processed frames per second; Number of detected object point clusters over time

ensure efficient and reliable processing of the acquired data sets, the system is divided into several, decentrally organized components. Objects detected by the system are broadcasted without any additional overhead or adaptions and can be used to control the robot velocity using the distances or to avoid collisions.

The system finds and segments individual object point clusters in the acquired point clouds. For each detected object point cluster, a convex hull is computed which approximates the shape of the underlying object. The detected objects are then integrated into the global environment model. With the help of the integrated filtering methods, the system is able to distinguish between the static environment, the robot and other obstacles in the surroundings at high update rates.

Although the evaluation of the system looks promising, there is still much room for improvements. A major challenge of the developed system is that the computed convex hulls are sometimes too large or too small and thus may give only a rough estimate of an object's surface. Additionally, the object geometry might be arbitrarily big in parts of total occlusion in all cameras. As this problem also affects the approximation of the human worker, future work should concentrate on this issue.

VII. Acknowledgements

This work is supported by the DFG cluster of excellence *CoTeSys*[1]. The authors also want to thank previous reviewers for their support in improving our work.

References

[1] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR Lightweight Robot arm - a new reference platform for robotics research and manufacturing," *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1 –8, June 2010.

[2] F. Wallhoff, J. Blume, A. Bannat, W. Rösel, C. Lenz, and A. Knoll, "A skill-based approach towards hybrid assembly," *Advanced Engineering Informatics*, vol. 24, no. 3, pp. 329 – 339, 2010, the Cognitive Factory. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474034610000406

[3] C. Lenz, M. Rickert, G. Panin, and A. Knoll, "Constraint task-based control in industrial settings," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE/RSJ. St. Louis, MO, USA: IEEE, Oct. 2009, pp. 3058–3063.

[4] M. Huber, A. Knoll, T. Brandt, and S. Glasauer, "When to assist?-Modelling human behaviour for hybrid assembly systems," in *Proceedings of ISR/ROBOTIK 2010*. VDE VERLAG GmbH, 2010.

[5] C. Lenz, A. Sotzek, T. Röder, H. Radrich, M. Huber, S. Glasauer, and A. Knoll, "Human workflow analysis using 3d occupancy grid hand tracking in a human-robot collaboration scenario," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3375–3380.

[6] J. Kruger, B. Nickolay, and O. Schulz, "Image-based 3d-surveillance in man-robot-cooperation," in *Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on*, June 2004, pp. 411 –420.

[7] J. Bosch and F. Klett, "Safe and flexible human-robot cooperation in industrial applications," in *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, October 2010, pp. 107 –110.

[8] A. Prusak, H. Roth, and R. Schwarte, "Application of 3d-pmd video cameras for tasks in the autonomous mobile robotics," in *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

[9] B. Winkler, "Safe Space Sharing Human-Robot Cooperation Using a 3D Time-of-Flight Camera," *International Robots and Vision Show*, 2007.

[10] M. Fischer and D. Henrich, "3D collision detection for Industrial Robots and Unknown Obstacles using Multiple Depth Images," in *German Workshop on Robotics*, June 2009.

[11] ——, "Surveillance of robots using multiple colour or depth cameras with distributed processing," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, 30 2009-September 2 2009, pp. 1 –8.

[12] C. Lenz, S. Nair, M. Rickert, A. Knoll, W. Rosel, J. Gast, A. Bannat, and F. Wallhoff, "Joint-action for humans and industrial robots for assembly tasks," in *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, August 2008, pp. 130 –135.

[13] C. Lenz, "Context-aware human-robot collaboration as a basis for future cognitive factories," Dissertation, Technische Universität München, München, Dec. 2011. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20111102-1078095-1-5

[14] F. Flacco and A. De Luca, "Multiple depth/presence sensors: Integration and optimal placement for human/robot coexistence," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3916–3923.

[15] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[16] E. Mücke, "Computing Prescriptions: Quickhull: Computing Convex Hulls Quickly," *Computing in Science Engineering*, vol. 11, no. 5, pp. 54 –57, September-October 2009.

[1]`www.cotesys.org`