Active Vision for Extraction of Physically Plausible Support Relations

Markus Grotz, David Sippel and Tamim Asfour

***Abstract*— Robots manipulating objects in cluttered scenes require a semantic scene understanding, which describes objects and their relations. Knowledge about physically plausible support relations among objects in such scenes is key for action execution. Due to occlusions, however, support relations often cannot be reliably inferred from a single view only. In this work, we present an active vision system that mitigates occlusion, and explores the scene for object support relations. We extend our previous work in which physically plausible support relations are extracted based on geometric primitives. The active vision system generates view candidates based on existing support relations among the objects, and selects the next best view. We evaluate our approach in simulation, as well as on the humanoid robot ARMAR-6, and show that the active vision system improves the semantic scene model by extracting physically plausible support relations from multiple views.**

I. Introduction

For autonomous robots, a semantic scene understanding is a key ability for a successful action execution, especially in unstructured scenes or cluttered table-top scenarios. Here, an increased awareness of the environment can facilitate task execution and further increase the success of a robot's manipulation. Fig. 1 gives an example of a cluttered table-top scenario with only a few objects, placed in a way that enforces a strict order of picking objects up. The manipulation order can be derived based on the support relation information between the objects. If the manipulation order is ignored the object stack would collapse.

In our previous work, we demonstrated and evaluated a system for the extraction of physically plausible support relations based on geometric primitives [1]. Model parameters for basic geometric shapes are iteratively fitted against a 3D input point cloud of the scene. A final step infers physically plausible support relations from the geometric shapes. Given the support relations between the objects, a robot is able to safely execute object manipulation tasks. Until now, camera view poses were chosen manually for the extraction of the scene model and the execution of manipulation actions. However, multiple views are necessary to address the limitations of a single view, such as occlusion of supported objects or missing support relations [2]. To extend our previous work, we present in this paper an active vision system to automatically determine a next best view (NBV) of a humanoid robot in order to explore a cluttered scene and extract a complete and consistent support relation graph describing support relations between the objects encountered in the scene.

Fig. 1: The humanoid robot ARMAR-6 in front of a cluttered table-top scenario. Knowledge about the support relations between the objects is required to safely manipulate the objects. Simply shifting the gaze is not sufficient due to occlusions. ARMAR-6 has to change the platform position as well in order to obtain a reasonably complete model of the scene.

Overall, the major contributions in this work can be summarized as follows: (a) Multi-view based scene modeling and semantic scene understanding, which is supported by (b) an active vision system to improve scene understanding by extracting support relations from multiple views by autonomous exploration, and (c) a real robot experiment validating our approach. The remainder of this work is structured as follows: After discussing relevant active vision approaches in section II, the proposed active vision system is detailed in section III and section IV as extension of the previous work. In section V, we evaluate proposed methods for various experimental setups, in simulation as well as on the real humanoid robot platform ARMAR-6. Finally, section VI concludes the paper and discusses future work.

II. Related Work

The term *active vision* was coined by Aloimonos et al. [3] in the late 1980s. *Active vision* is defined as modifying the camera view pose actively and purposefully with the goal to enhance the current perception. A similar definition for *active perception* was given by Bajcsy *et al.* [4]. Recently, *active perception* was revisited by Aloimonos, Bajcsy and

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project ROBDEKON (13N14678).

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany.
{grotz, asfour}@kit.edu

Tsotos in their survey paper [5]. The work of Chen *et al.* [6] surveys relevant active vision methods. An overview of more recent active vision methods is given in [7], extending the work of Chen *et al.* as well as including other aspects. Active vision has a strong link to the current task [8] and is required to support many robotic vision applications, for example SLAM, where an active vision system can improve registration by selecting regions of interest and changing the camera pose [9].

One particular active vision problem is the selection of the next best view (NBV), which determines the next camera pose to obtain a reasonably complete scene or object model. The NBV problem was pioneered by Connolly [10] for object modeling tasks and later improved by Pito [11] and Banta et al. [12]. It has also been widely addressed for eye-in-hand systems in industrial applications. In humanoid robots, however, selecting a suitable view pose is especially difficult due to the robot's position uncertainty, the unreachability of the view pose or the complexity of the current task. Therefore, the application of the NBV problem in humanoid robots is related, but differs in some respects when compared to industrial robot arms. The particular problem of unreachable view poses due to kinematic constraints was addressed by Foissotte et al. [13] for the humanoid robot HRP-2. Here, the goal was to compute whole robot postures to autonomously generate visual models of unknown objects. For humanoid robots, the processes of finding the NBV can be accelerated by filtering possible view candidates with inverse reachability maps as described in [14]. A general next best view system for the humanoid robot PR2 to explore a scene was presented by Potthast and Sukhatme [15]. The system is designed for occluded environments and estimates the visibility of occluded space. In [16] a trajectory optimization method to explore new and occluded regions for robotics grasping is presented. The view candidate evaluation of NBV is often costly and therefore methods utilize GPU to speed up the NBV calculation for humanoid robots [17] or for industrial robotic arms [18]. Xu et al. [19] presented a NBV system for shape classification. Their work is evaluated on the PR2 humanoid robot as well and uses a recurrent 3D attention model. The 3D attention model selects the NBV to improve object identification and scene exploration.

III. Scene Model

Our system starts by building a scene model from sensor data. Fig. 2 illustrates our system architecture to extract the scene model and to compute the next best view (NBV).

To build a scene model, RGB-D images are first registered with respect to each other. Then, the registered point cloud is segmented into plausible parts as a hint for possible objects. In the subsequent step, we approximate the shape of the objects in the scene, denoted as $\mathcal{O}$, with geometric primitives. These geometric primitives include boxes, spheres, or cylinders and are fit against each segment using a RANSAC based approach. For details, the reader is referred to our previous work [1].

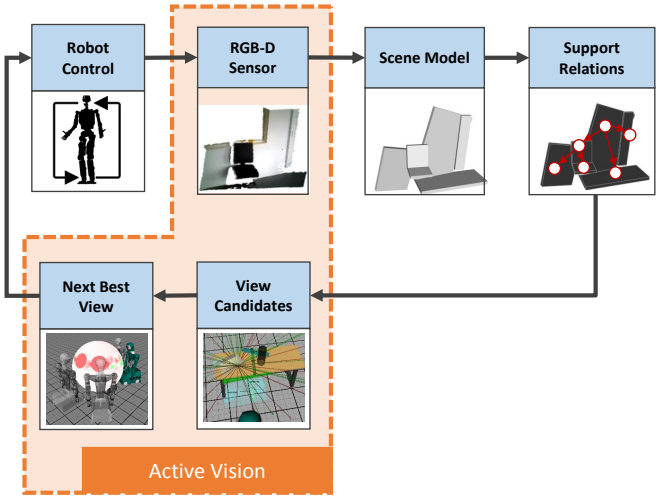


Fig. 2: The system architecture. A scene model is extracted from multiple views and support relations between the objects are inferred automatically. Our core contribution is an active vision system that determines automatically the robot's next best view to improve the extraction of physically plausible object support relations.

A. Support Relations

Next, we infer support relations based on the spatial extent of the geometric primitives. We follow the notation and definition of the support relations given in [1], [20]. For clarity we recap the most relevant definitions for this work. For two objects $\mathtt{A}, \mathtt{B} \in \mathcal{O}$ we denote $\mathtt{SUPP}(\mathtt{A}, \mathtt{B}) \iff \mathtt{A}$ supports $\mathtt{B}$. The support graph, spanning the geometric primitives, is a directed graph $\mathcal{G}_s = (V, E)$, where the vertices V map the set of objects $\mathcal{O}$ and the edges E model possible support relations between the objects. First, separating planes between objects are constructed at contact points. For each pair of objects in contact, a support relation is added from the object below the separating plane to the object above the plane, resulting in bottom-up support relations. In addition, uncertain support relations are detected purely on geometric information by computing a support area ratio. To this end, the object model's shape is projected to the ground floor and the overlap between the convex hull of each projected object is computed. The intersecting area is then combined into the set of polygons $\mathbb{P}_\mathtt{A} = \{P_\mathtt{A} \cap P_\mathtt{C} \mid \mathtt{A}, \mathtt{C} \in \mathcal{O},\ \mathtt{SUPP}(\mathtt{C}, \mathtt{A})\}$ representing the directly supported area of $\mathtt{A}$. The support polygon P_s is the convex hull of the polygons in $\mathbb{P}_\mathtt{A}$. Next, we compute the support area ratio $r_s \in [0, 1]$ defined as

$$r_s = \frac{\text{area}(P_s)}{\text{area}(P_\mathtt{A})} \ , \tag{1}$$

where $r_s = 1$ means that $\mathtt{A}$ is fully supported, and $r_s = 0$ means that $\mathtt{A}$ is not supported at all, i. e. $\mathtt{A}$ is floating. If $r_s < r_{s,\min}$ then we consider object $\mathtt{A}$ as unstable. In this work, we set $r_{s,\min} = 0.1$. For each unstable object $\mathtt{A}$ we add a new edge $e = (\mathtt{B}, \mathtt{A})$ labeled as uncertain to E. Otherwise, if $r_s > r_{s,\min}$ we consider $\mathtt{A}$ as well supported by the objects below it and therefore no edges are added. A major reason

	Point Cloud		Support Graph Combination		
t	Input	Registered	PC only	SG only	PC + SG
1	$\mathcal{P}_1$	$\mathcal{P}^{1,1} = \mathcal{P}_1$	$\mathcal{G}_s^1$	$\mathcal{G}_s^1$	$\mathcal{G}_s^1$
2	$\mathcal{P}_2$	$\mathcal{P}^{1,2} = \mathcal{P}_1 \cup \mathcal{P}_2$	$\mathcal{G}_s^2$	$\mathcal{G}_s^{1,2}$	$\mathcal{G}_s^1 \cup \mathcal{G}_s^2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$\mathcal{P}_n$	$\mathcal{P}^{1,n} = \bigcup_{i=1}^{n} \mathcal{P}_i$	$\mathcal{G}_s^n$	$\mathcal{G}_s^{1,n}$	$\mathcal{G}_s^{n-1} \cup \mathcal{G}_s^n$

TABLE I: Schema of support graph combination methods.

why an object is considered unstable is an incomplete scene model, e. g. a supporting object is missing or the scene is not fully explored yet. Therefore, we are in particular interested in the support relations as a clue for incomplete data of the scene model.

B. Support Graph Combination

For the scene model and semantic scene understanding, the information from multiple views needs to be combined. A simple strategy to combine the support graph information is to extract the scene model and support relations after each single view from a global point cloud $\mathcal{P}^{1,t}$, containing all the registered previous views. We will refer to this case in the following as *Point Cloud only (PC only)*. To speed-up the process, the global point cloud is downsampled first since the geometric primitive fitting step is computationally expensive and the computation time scales with the input size of the point cloud. Overall, there are two major arguments against computing the support graph from a global point cloud: (a) the total runtime is significantly increased, and (b) the registration of the views might not be optimal and thus the RANSAC primitive fitting approach might fail to find all inliers for a geometric shape. Hence, we extract the geometric primitives and the support graph iteratively from each view and fuse the result with a global consistent support graph $\mathcal{G}_s^{1,t}$. To fuse an existing support graph $\mathcal{G}_s^t = (V^t, E^t)$ with the support graph from the current view $\mathcal{G}_s^{t+1} = (V^{t+1}, E^{t+1})$ we first match the vertices, i. e. the objects, as presented in [21]. To this end, we check the extracted object's shape, position, and orientation as well as the extent of the object. Similarly, we check if an edge $e = (\mathtt{A}, \mathtt{B}) \in E^{t+1}$ already exists in E^t. If $e \notin E^t$ then it will be added as a new edge. We further keep track of how many times an edge e has been extracted and matched. The number of matches will be denoted as $\#occ(e)$. This allows the algorithm to later validate the existence of the support relations that have been visible only a few times. We will refer to this case as *Support Graph only (SG only)*. Finally, we present a combination of the two methods *(PC + SG)*, where we first register the point cloud as proposed in the first approach, and then merge the extracted support graph as described in the second approach. Table I outlines the different support graph combination methods.

IV. Active Vision System

This section describes the active vision system as shown in our system architecture depicted in Fig. 2. Once the support graph is extracted from the first view the active vision system determines the next best view, i. e. the location of the platform and gaze direction. To this end, view poses are represented on a sphere with a fixed radius and later translated into a platform position and gaze direction. The center of the view sphere S is defined as the centroid of the extracted table-top segment from the initial view. In contrast to standard NBV methods, we do not sample equally on the view sphere. Instead we compute Points of Interest (PoI) based on the support graph as detailed in Algorithm 1. For each PoI x we compute a saliency value s to model the importance of a view pose. The idea is similar to the work of [22], where a saliency value is used to filter view candidates, which are then subjected to further evaluation. In this work, we consider semantic information, that is the support relations, instead of spatial information, i. e. the frontiers of the segmentation. Therefore, the point of interest

Algorithm 1: Points of Interest (PoI) Generation

Data: Support Graph $\mathcal{G}_s = (V, E)$, View sphere S, Previous view poses $\hat{v}_0, \ldots, \hat{v}_t$

$PoI \leftarrow \emptyset$;
foreach $\mathtt{A} \in V$ **do**
  $U \leftarrow \{\mathtt{B} \in V | (\mathtt{B}, \mathtt{A}) \in E\}$;
  if $U = \emptyset$ **then**
    $\theta \leftarrow$ GetLargestArc(S, $\hat{v}_0, \ldots, \hat{v}_t$);
    $x \leftarrow$ ComputeContactPoints(A, $\frac{\theta}{2}$);
    $s \leftarrow$ ComputeSaliency(x);
    $PoI \leftarrow PoI \bigcup \{(x, s)\}$;
  foreach $\mathtt{B} \in U$ **do**
    $x \leftarrow \mathtt{A} + \frac{(\mathtt{B} - \mathtt{A})}{\frac{1}{2}||\mathtt{B} - \mathtt{A}||_2}$;
    $s \leftarrow$ ComputeSaliency(x);
    $PoI \leftarrow PoI \bigcup \{(x, s)\}$;
return PoI;

Algorithm 2: Next Best View Selection

Data: Support Graph $\mathcal{G}_s$, Sphere Center c_x, Voxel Map V

$S \leftarrow$ CreateViewSphere(c_x);
$PoI \leftarrow$ GeneratePoI($\mathcal{G}_s$, S, ...);
foreach $(x, s) \in PoI$ **do**
  $v \leftarrow$ ProjectToSphere(x);
  $r \leftarrow$ RayCast(x, v, V);
  $p \leftarrow$ PathPlanning(v);
  $h \leftarrow$ ComputePathCosts(v);
  if *IsIntersectionFree(r, V) and IsReachable(v)* **then**
    AddToViewSphere(v, S);
$\hat{v} \leftarrow \arg\max_{v \in V} c(v)$;
return $\hat{v}$;

already gives an indication of a good view pose and we can reduce the number of view candidates in a subsequent evaluation step. We note that this approach might not find an optimal next best view, but requires less candidates to evaluate. The implementation of the view sphere is similar to [23], but we use an egocentric sphere instead of an exocentric one. Overall, the sphere is discretized to 40.000 equally distributed points.

A. View Candidate Generation

Possible view candidates are generated based on points of interest in the scene. We propose two strategies to generate points of interest. In a first step, for each unsupported object PoIs are generated based on an object's extent and previous views. We first select the largest arc on the view sphere between previous positions of the robot. The point of interest is then defined as the intersection of the object and the line from the middle of the largest arc to the object's center. The underlying concept is that each object must have at least one support edge due to gravity and so far either the object is not fully visible or supporting objects are missing. We use a maximum saliency value to account for further exploration of the object and its area. In a second step, points of interest are computed based on the edges between objects. Here, the idea is to consider a relation in the support graph as more stable depending on the number of times it has been observed. In this case the saliency value $s(v)$ is computed with

$$s(v) = \lambda_s + (1 - \lambda_s) \cos\left(2\pi \cdot \frac{\#occ(e_x)}{\#views}\right), \qquad (2)$$

where e_x is the edge associated with the point of interest x, $\#occ(e_x)$ is the number of times the edge e_x was observed, $\#views$ the number of total views and λ_s is a parameter to define the importance of the support edge validation.

The points of interest are then projected on to the view sphere to represent possible views of the robot. Similar to [23], the saliency value is propagated to neighboring view poses on the view sphere with a decreasing value. Occlusions are mitigated by checking if the line of sight between the PoI x and the projected PoI v is free. We therefore keep a voxel grid of the environment. In case of occlusion the saliency value is inverted.

B. Next Best View Selection

We first discard unreachable view candidates on the sphere while taking into account the distance of a view candidate since moving the robot is time consuming. Therefore, a cost function

$$c(v) = s(v) - h(v) \qquad (3)$$

pivots the saliency $s(v)$ and the costs $h(v)$ of moving the robot for a given view v. The costs $h(v)$ correspond to the costs of reaching the position of view v with the robot's platform. Here, we set $h(v) = \sin(\frac{\alpha}{2})$, where α is the angle between the current view position and the view pose v with respect to the centroid of the view sphere. Algorithm 2 outlines the approach. The active vision system terminates if the current view pose does not contribute significantly to the support graph, i. e. the graph is not modified, or after n views are reached (we limit the total number of views to 10). To prevent the system from getting stuck we added an *Inhibition of Return* mechanism which forces the robot to attend new view poses. To this end, we keep track of previously visited view poses and add a negative saliency value to the sphere. The view pose on the sphere which maximizes the costs is chosen as the next best view.

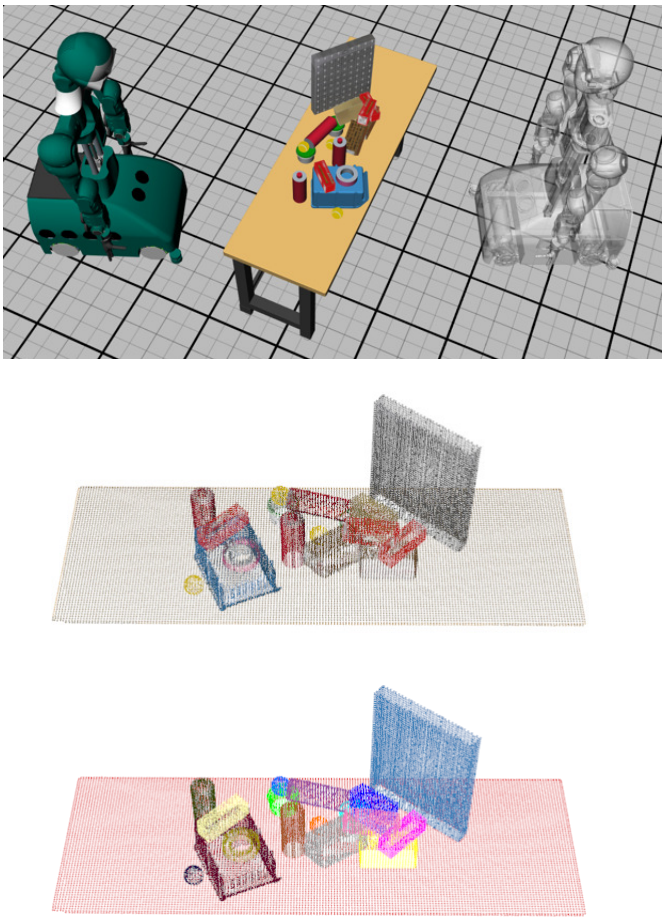


Fig. 3: ARMAR-6 simulated experiment. *Top:* A cluttered table-top scenario. *Middle:* The registered point cloud of the cluttered table-top. *Bottom:* The ground truth segmentation of the scene.

V. Evaluation

For all experiments the humanoid robot ARMAR-6 [24] is used. ARMAR-6 features a holonomic platform and has two degrees of freedom in total for the head: yaw and pitch. Further, the robot is equipped with several sensors for perception. For the experiments we utilize ARMAR-6's Carmine 1.09 RGB-D sensor. The working range of the depth measurement was limited to 3 m in the experiments to reduce sensor noise. The segmentation, as a hint for the geometric primitive fitting, was manually refined to avoid bias of the RANSAC based geometric primitive fitting and to make the experiment reproducible. We set $\lambda_s = 0.75$ for the saliency

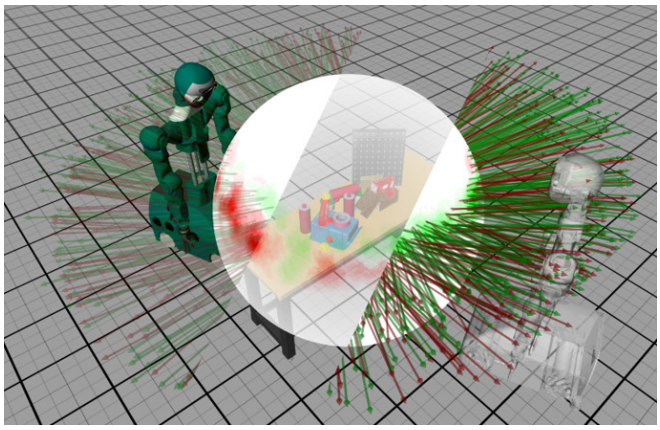


Fig. 4: The view sphere including the projected point of interests of the simulated experiment. The rays project the point of interest to the sphere and a saliency value models the interest of the view pose. Parts of the sphere have been made transparent for visualization purposes. The selected next best view is visualized in gray.

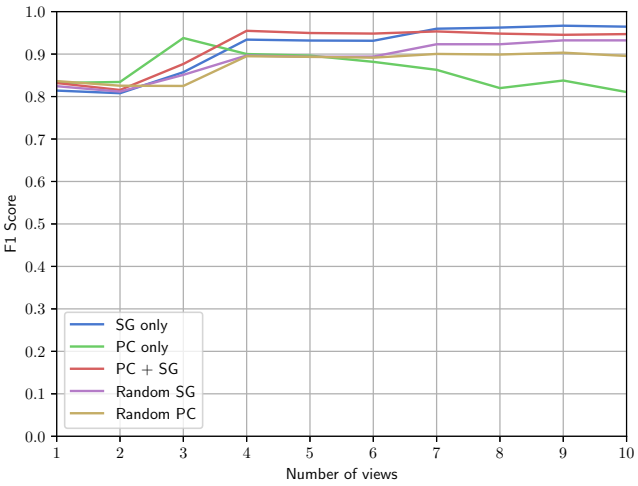


Fig. 5: The F_1 score for the first 10 next best views of the simulated experiment. The extracted support graph from the scene is compared to a ground truth support graph. Different methods include the combination of each view based on the spatial information (*PC only*), the support graph (*SG only*) or both (*PC + SG*). The active vision system was compared to random placement by fusing the information on the support graph (*Random SG*) and spatial information (*Random PC*).

computation, as it shows a good balance between exploration and validation of support relations. The radius of the view sphere can be parameterized and was set to $1.5\,\mathrm{m}$. The leaf size of the voxel grid for occlusion checking was set to $1\,\mathrm{cm}$.

A. Evaluation in Simulation

We qualitatively evaluate our approach in simulation. Fig. 3 visualizes the scenario setup. The view sphere after the initial view is visualized in Fig. 4. We compare the ground truth of a manually created support graph $\mathcal{G}_s^{GT} = (V^{GT}, E^{GT})$ with the result $\mathcal{G}_s^i = (V^i, E^i)$ of the i-th view from our proposed method. The F_1-score models the accuracy of the extracted support graph. We set

$$\text{precision} = \frac{\left|V^{GT} \cap V^i\right| + \left|E^{GT} \cap E^i\right|}{|V^i| + |E^i|} \tag{4}$$

and

$$\text{recall} = \frac{\left|V^{GT} \cap V^i\right| + \left|E^{GT} \cap E^i\right|}{|V^{GT}| + |E^{GT}|} \quad . \tag{5}$$

The true positives are vertices and edges that exists in $\mathcal{G}_s^{GT}$ as well as in $\mathcal{G}_s^i$. The false positives are the number of vertices and edges in $\mathcal{G}_s^i$ but not in $\mathcal{G}_s^{GT}$. False positives can occur due to an erroneous RANSAC model fitting. False negatives are the vertices and edges missing in $\mathcal{G}_s^i$. The result of the F_1-score for different matching approaches, as described in section III-B, is reported in Fig. 5. Different methods include the combination of each view based on the spatial information (*PC only*), the support graph (*SG only*) or both (*PC + SG*). The active vision system was compared to random placement of the robot while fusing the information on the support graph (*Random SG*) and spatial information (*Random PC*). Notably, all approaches yield an increase with respect to the F_1-score after the second view. However, extracting the support graph only from a single registered point cloud results in a decline of the F_1-score after the fourth view. One reason for this is the fitting of the geometric primitives, which works with a fixed error threshold of the fitted geometric model. Furthermore, the F_1-score increases with the random placement of the robot as well. However, the random placement of the robot does not consider the distance to reach the next position. Therefore, the total distance traveled by the robot during exploration might be significantly larger than with the proposed methods. This is not taken into account by the F_1- score.

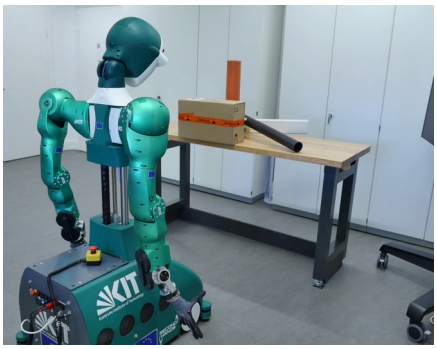
B. Real World Evaluation

The real world evaluation is similar to our evaluation in simulation. This time, however, the noise of the sensor and the registration error injects noise into the system. We chose to fuse the support graph with the second approach *(SG only)* as it performs well in simulation while reducing computation time. Fig. 6 depicts three selected next best views at different timestamps of the experiment. The scene is relatively simple but due to occlusion requires multiple views to extract a complete support graph of the scene.

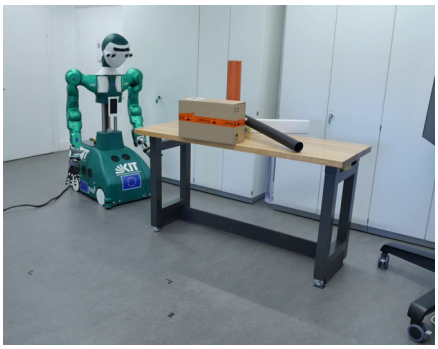
As one can observe from the first view (first column of Fig. 6), no support relations are extracted due to an occluding object. The active vision system therefore generates PoIs between each object pair and the robot attends more next best views. In the second column, the support graph is still incomplete, but the most important support relations are discovered. Finally, in the third column the next best view discovers a missing support relation.

VI. Conclusion

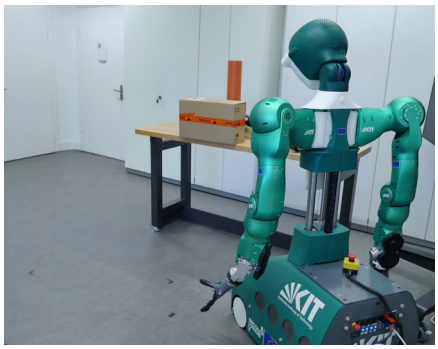
We presented an integrated active vision system to support the creation of a semantic scene model including the



(a) First view

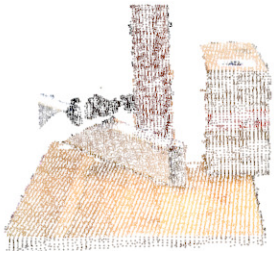


(b) Second view

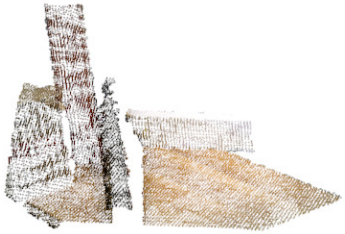


(c) Third view

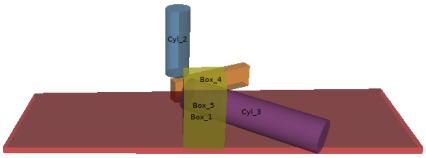
(d) First view (point cloud)



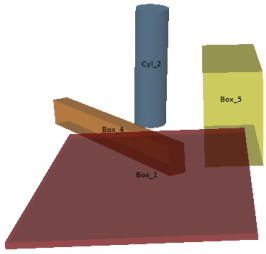
(e) Second view (point cloud)



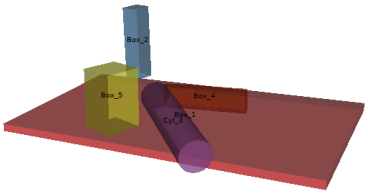
(f) Third view (point cloud)



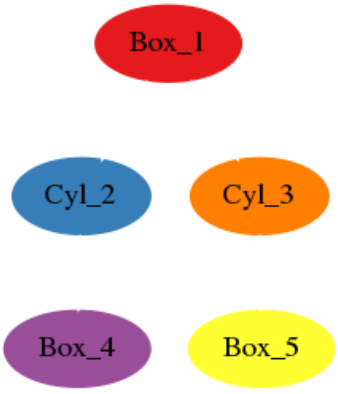
(g) First view (geometric primitives)



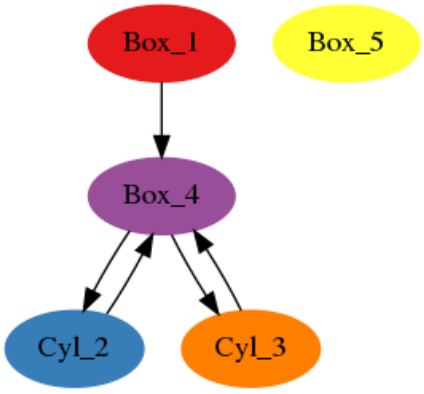
(h) Second view (geometric primitives)



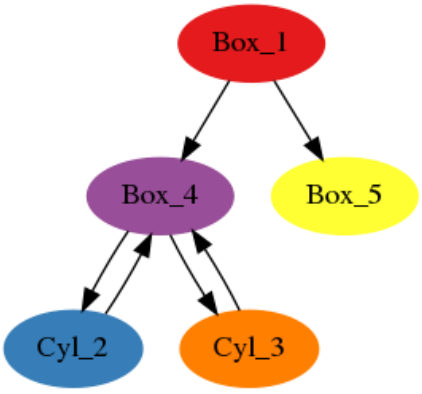
(i) Third view (geometric primitives)



(j) First view (support graph)



(k) Second view (support graph)



(l) Third view (support graph)

Fig. 6: A real world experiment with the humanoid robot ARMAR-6 showing three next best views at different timestamps of the experiment. *First row*: Scene and position of the robot (a-c). *Second row*: Current point clouds (d-f). *Third row*: Extracted geometric primitives (g-i). *Fourth row*: Extracted support graph (j-l).

extraction of physically plausible support relations based on multiple views of the scene. View candidates are generated based on their support relations and thereby explore the scene for missing supported objects. Our experiments show that multiple views are necessary to mitigate the effect of occluded objects. Both the evaluation in simulation as well as the real world experiment show a completion of the support graph after a few attended next best views. Based on annotated ground truth data, the F_1 score reaches $94\,\%$ after the fourth next best view and converges to a complete model iteratively. The real world experiment demonstrates the necessity of the active vision method for cluttered table-top scenarios. Future work will focus on incorporating other semantic data to further improve the next best view. In addition, we will evaluate the method on more challenging scenarios and extend the evaluation with manipulation experiments.

REFERENCES

[1] R. Kartmann, F. Paus, M. Grotz, and T. Asfour, "Extraction of physically plausible support relations to predict and validate manipulation action effects," *IEEE Robotics and Automation Letters*, pp. 3991–3998, 2018.

[2] S. Panda, A. H. Abdul Hafez, and C. V. Jawahar, "Single and multiple view support order prediction in clutter for manipulation," *Journal of Intelligent & Robotic Systems*, vol. 83, no. 2, pp. 179–203, 2016.

[3] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 333–356, 1988.

[4] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[5] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, pp. 177–196, 2017.

[6] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[7] E. Potapova, M. Zillich, and M. Vincze, "Survey of recent advances in 3d visual attention for robotics," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 1159–1176, 2017.

[8] J. Aloimonos, "Purposive and qualitative active vision," in *International Conference on Pattern Recognition*, pp. 346–360, 1990.

[9] S. Frintrop and P. Jensfelt, "Attentional landmarks and active gaze control for visual slam," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1054–1065, 2008.

[10] C. Connolly, "The determination of next best views," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 432–435, 1985.

[11] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[12] J. E. Banta, L. R. Wong, C. Dumont, and M. A. Abidi, "A next-best-view system for autonomous 3-d object reconstruction," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, no. 5, pp. 589–598, 2000.

[13] T. Foissotte, O. Stasse, A. Escande, and A. Kheddar, "A next-best-view algorithm for autonomous 3d object modeling by a humanoid robot," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 333–338, 2008.

[14] S. Oßwald, P. Karkowski, and M. Bennewitz, "Efficient coverage of 3d environments with humanoid robots using inverse reachability maps," in *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, pp. 151–157, 2017.

[15] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 148–164, 2014.

[16] G. Kahn, P. Sujan, S. Patil, S. Bopardikar, J. Ryde, K. Goldberg, and P. Abbeel, "Active exploration using trajectory optimization for robotic grasping in the presence of occlusions," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4783–4790, 2015.

[17] S. Oßwald and M. Bennewitz, "Gpu-accelerated next-best-view coverage of articulated scenes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 603–610, 2018.

[18] R. Monica, J. Aleotti, and S. Caselli, "A kinfu based approach for robot spatial attention and view planning," *Robotics and Autonomous Systems*, vol. 75, pp. 627–640, 2016.

[19] K. Xu, Y. Shi, L. Zheng, J. Zhang, M. Liu, H. Huang, H. Su, D. Cohen-Or, and B. Chen, "3d attention-driven depth acquisition for object identification," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–14, 2016.

[20] R. Mojtahedzadeh, A. Bouguerra, E. Schaffernicht, and A. J. Lilienthal, "Support relation analysis and decision making for safe robotic manipulation tasks," *Robotics and Autonomous Systems*, vol. 71, pp. 99–117, 2015.

[21] M. Grotz, P. Kaiser, E. E. Aksoy, F. Paus, and T. Asfour, "Graph-based visual semantic perception for humanoid robots," in *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, pp. 869–875, 2017.

[22] R. Monica and J. Aleotti, "Contour-based next-best view planning from point cloud segmentation of unknown objects," *Autonomous Robots*, vol. 42, no. 2, pp. 443–458, 2018.

[23] M. Grotz, T. Habra, R. Ronsse, and T. Asfour, "Autonomous view selection and gaze stabilization for humanoid robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1427–1434, 2017.

[24] T. Asfour, L. Kaul, M. Wachter, S. Ottenhaus, P. Weiner, S. Rader, R. Grimm, Y. Zhou, M. Grotz, F. Paus, D. Shingarey, and H. Haubert, "Armar-6: A collaborative humanoid robot for industrial environments," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 447–454, 2018.