Perception and Reasoning for Scene Understanding in Human-Robot Interaction Scenarios

Nikhil Somani†, Emmanuel Dean-León‡, Caixia Cai† and Alois Knoll† ⋆

Authors Affiliation: †Technische Universität München, Fakultät für Informatik.
‡Cyber-Physical Systems, fortiss - An-Institut der Technischen Universität München
Email: †{somani,caica,knoll}@in.tum.de, ‡{dean}@fortiss.org

Abstract. In this paper, a combination of perception modules and reasoning engines is used for scene understanding in typical Human-Robot Interaction(HRI) scenarios. The major contribution of this work lies in a 3D object detection, recognition and pose estimation module, which can be trained using CAD models and works for noisy data, partial views and in cluttered scenes. This perception module is combined with first-order logic reasoning to provide a semantic description of scenes, which is used for process planning. This abstraction of the scene is an important concept in the design of intelligent robotic systems which can adapt to unstructured and rapidly changing environments since it provides a separation of the process planning problem from its execution and scenario-specific parameters. This work is aimed at HRI applications in industrial settings and has been evaluated in several experiments and demonstration scenarios for autonomous process plan execution, human-robot interaction and co-operation.

1 Introduction

Industrial robotics is currently witnessing a phase where a lot of effort is directed towards applications of standard industrial robots in smaller industries with short production lines, where the environment is rather unstructured and rapidly changing. Scene understanding is an important component in the development of intelligent industrial robotics solutions. It provides information about the working environment which is used by reasoning modules and intelligent control algorithms to create an adaptive system. The separation of the problem space from the execution space (which contains scenario-specific parameters), is an important concept in these systems. A mapping between these spaces is provided by the perception and reasoning modules. On one hand, they provide an abstraction of the world which is used to learn tasks by demonstration and on the other hand, they provide scenario specific information which is used by the low-level execution and control modules for plan execution.

⋆ The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n 287787.

The perception problem in the industrial robotics context involves detecting and recognizing various objects and actors in the scene. The objects in the scene consist of workpieces relevant to the task and obstacles in the robot's path. The actors involved are humans, and the robot itself. One of the major contributions of this work is an object detection, recognition and pose estimation module, which uses 3D point cloud data obtained from low-cost depth sensors like the Kinect. The reasoning module used in this work is based on knowrob [1] and performs spatial and temporal reasoning about objects in the scene. An extension of this reasoning engine and creation of an Ontology specific to the industrial robotics domain are also contributions from this work.

Object detection, recognition and pose estimation using 3D point clouds is a well researched topic. The popular approaches for this task can be broadly classified as: local color keypoint [2], [3], local shape keypoint [4], global descriptors [5], geometric [6], primitive shape graph [7], [8]. Each of these approaches have their own advantages and disadvantages. For example, color based methods would not work on texture-free objects. Shape based methods can not distinguish between objects having identical shape but different texture. Global descriptors such as VFH [5] require a tedious training phase where all required object views need to be generated using a pan-tilt unit. Besides, its performance decreases in case of occlusions and partial views. The advantage of these methods, however, lies in their computational speed. Some other methods such as [7], [9] provide robustness to occlusions, partial views and noisy data. However, these methods are rather slow and not suitable for real-time applications in large scenes. In this paper, an extension to the ORR [9] method (PSORR) has been proposed, where the effort has been directed towards a solution which enhances its robustness to noisy sensor data and also increases its speed. Another object recognition and pose estimation algorithm has been proposed, which is complementary to the PSORR method with respect to the target object geometries.

To distinguish objects having identical geometry but different color, the point cloud is segmented using color information and then used for object detection. There are several popular approaches for Point cloud segmentation such as Conditional Euclidean Clustering [10], Region Growing [11], and graph-cuts based segmentation methods [12]. In this paper, a combination of multi-label graph-cuts based optimization [12] and Conditional Euclidean Clustering [10] is used for color-based segmentation of point clouds.

2 Object Recognition and Pose Estimation

2.1 Shape Based Object Recognition from CAD models

There are two complementary approaches presented here. One is an extension of the ORR method [9] called Primitive Shape Object Recognition Ransac (PSORR), and the other is based on Primitive Shape Graph (PSG) matching. The results obtained are qualitatively similar for both approaches. The PSORR method is more suitable for handling arbitrary object geometries and objects having few primitive shapes while the PSG method is more suitable for large

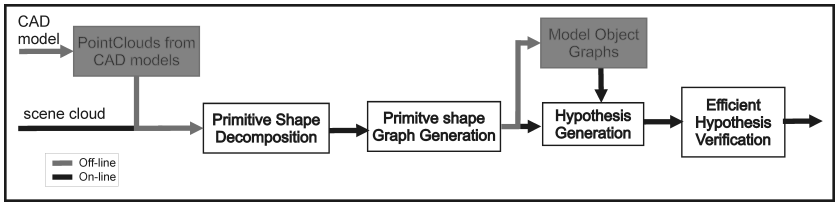

Fig. 1. Pipeline for Shape based perception.

models which decompose into a large number of stable primitive shapes. The pipeline for this module is shown in Fig. 1.

2.1.1 Primitive Shape Decomposition

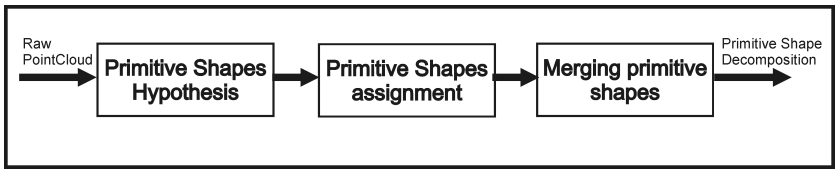

Fig. 2. Pipeline for Primitive Shape Decomposition.

The pipeline for this step is shown in Fig. 2. This step is very important for the algorithm because the hypothesis generation and pose estimation step are based on this decomposition. The hypothesis verification step, which is a major bottleneck in most algorithms such as ORR, can also be significantly simplified and sped-up using this decomposition.

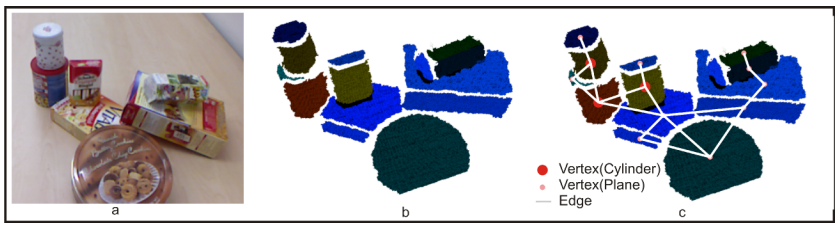

Fig. 3. Primitive Shape Decomposition example : (a) original Point Cloud (b) result of Primitive Shape Decomposition (c) Primitive Shape Graph representation.

The point cloud P is represented as a set of primitive shapes s_i containing points $p_i \subseteq P$ such that $\cup p_i \subseteq P$. The primitive shapes s_i could be planes, cylinders, etc. An example of such a decomposition is shown in Fig. 3, where the original scene cloud is shown in Fig. 3 (a) and its decomposition into primitive shapes is shown in Fig. 3 (b).

Primitive Shape Hypothesis

Hypothesis for primitive shapes are generated by randomly sampling points in the point cloud. Once the hypotheses have been generated, each point in the cloud is checked to determine whether it satisfies the hypotheses. The method used for generating a hypothesis and determining its inliers depends on the type of primitive shape.

- **Planes:** A plane hypothesis can be generated using a single point (X_0) with its normal direction ($\hat{n}$). To test if a point X lies on the plane $(X - X_0).\hat{n} = 0$, the distance of the point from the plane $|(X - X_0).\hat{n}|$ is used.
- **Cylinders:** A cylinder hypothesis can be generated using 2 points (X_0, X_1) with their normal directions $(\hat{n_0}, \hat{n_1})$. The principal axis of the cylinder is selected as the minimum distance line between the normal directions $\hat{n_0}$ and $\hat{n_1}$. The radius r is the distance of either point to this line. To test if a point X lies on the cylinder, the distance of the point from the cylinder's axis is used.

Primitive Shape Assignment

The hypotheses associated with each point in the cloud can be considered as labels for point. There may be multiple labels associated with each point and the labeling may be spatially incoherent. To resolve such issues and generate a smooth labeling, a multi-label optimization using graph-cuts is performed. In this setting, the nodes in the graph comprise all possible assignment of labels to the points. The data term indicating the likelihood of a label assignment to a point is inversely proportional to the distance of the point from the primitive shape. The smoothness term penalizes neighboring points having different labels and the penalty is inversely proportional to the distance between the neighboring vertices. Label swap energies are used for neighboring primitive shapes in a way that only neighboring primitive shapes labels can be swapped. This convex energy functional is then solved using the α - expansion, β -swap algorithms [12] which give the label assignment for each point in the cloud, such that the total energy is minimized.

Merging Primitive Shapes

Each primitive shape has a $fitness\_score$ associated with it which indicates how well the primitive matches the point clouds. It is based on the minimum

descriptor length(MDL) approach [13]. The fitness score of a primitive shape is defined as :

$$fitness\_score = \frac{inliers}{total\_points} + K * descriptor\_length \tag{1}$$

where, the first fraction represents the inlier ratio, i.e., the ratio of points which satisfy the primitive shape (*inliers*) to the total number of points in the input cloud (*total_points*), *descriptor_length* represents the complexity of the primitive shape (e.g. the number of values required to represent the shape). The constant K determines the relative weighting of the two factors. Higher values of K will support under-segmentation resulting in bigger, less accurate primitives, while low values will hamper robustness against over-segmentation, causing fewer merges and resulting in fragmented, over-fitted primitives.

The merging strategy is based on a greedy approach where pairs of primitive shapes are selected and merged if the combined primitive shape has a better fitness score than the individual primitive shapes. This continues till there are no more primitive shapes which can be merged.

2.1.2 Primitive Shape Graph(PSG) Representation

The primitive shapes detected in the previous step are now used to create a graphical representation of the point cloud. In this graph $G = (V, E)$, each primitive shape is a node $v \in V$ and neighboring primitive shapes are connected by an edge $e \in E$. An example of such a graph is shown in Fig. 3 (c).

2.1.3 Hypothesis Generation

PSORR method

An oriented point pair (u, v) contains two points along with their normal directions: $u = (p_u, n_u)$ and $v = (p_v, n_v)$. A feature vector $f(u, v)$ is computed from this point pair, as shown in Eq. 2.

$$f(u,v) = \left(\|p_u - p_v\|, \angle\left(n_u, n_v\right), \angle\left(n_u, p_v - p_u\right), \angle\left(n_v, p_u - p_v\right)\right)^T, \tag{2}$$

The central idea in the ORR method is to obtain such oriented point pairs from both the scene and model point clouds and match them using their feature vectors. For efficient matching of oriented point pairs, a Hash Table is generated containing the feature vectors from the model point cloud. The keys for this table are the three angles in Eq. 2. Each Hash Cell contains a list of models ($M_i \in M$) and the associated feature vectors. Given an oriented point pair in the scene cloud, this Hash Table is used to find matching point pairs in the model cloud. Each feature vector f has an associate homogeneous transformation matrix F associated with it, see Eq. 3.

$$F_{uv} = \begin{pmatrix} \frac{p_{uv} \times n_{uv}}{\|p_{uv} \times n_{uv}\|} & \frac{p_{uv}}{\|p_{uv}\|} & \frac{p_{uv} \times n_{uv} \times p_{uv}}{\|p_{uv} \times n_{uv} \times p_{uv}\|} & \frac{p_u + p_v}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

where $p_{uv} = p_v - p_u$ and $n_{uv} = n_u + n_v$. Hence, for each match f_{wx} in the hash table corresponding to f_{uv} in the scene, a transformation estimate can be obtained: see Eq. 4. This transformation estimate (T_i) forms a hypothesis $h_i = \{T_i, M_i\} \in H$ for the model (M_i) in the scene.

$$T = F_{wx} F_{uv}^{-1} \tag{4}$$

The raw point clouds are generally noisy, especially the normal directions. The original ORR method is sensitive to noise in the normal directions and hence, randomly selecting points to generate the feature vectors requires more hypothesis until a good oriented point pair is found. In the PSORR method, every node representing a plane in the scene PSG is considered as an oriented point (u) with the centroid of the plane as the point (p_u) and the normal direction as the orientation (n_u). The normal directions for these oriented points are very stable because they are computed considering hundreds of points lying on the plane. Therefore, we can use these centroids instead of the whole cloud to compute and match features, which leads to a significantly less number of hypotheses.

The centroid for the scene cloud primitives might not match the model centroids in case of partial views. Hence, for the model cloud, the point pairs are generated by randomly sampling points from every pair of distinct primitive shape clouds.

PSG Matching for hypothesis generation

In cases where the PSG is rather large and the individual primitive shapes are

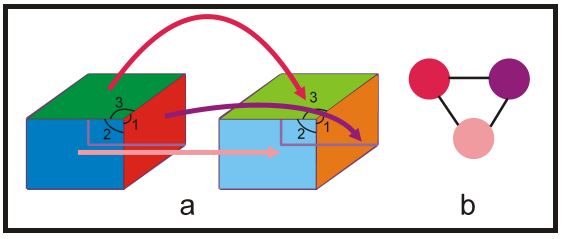

Fig. 4. PSG matching: (a) Matching of cliques 3 intersecting planes generates a full hypothesis. Each vertex assignment is a node, (b) Vertex assignments arising from the same clique are considered neighbors in the graph.

small, the speedups obtained by the PSORR method are not significant due to the additional cost of primitive shape decomposition. In this case, another approach is used where the scene PSG is matched with model PSG's and used to recognize the object and estimate its pose. Given both model and scene PSG's, the problem of object recognition becomes equivalent to constrained sub-graph matching, which is an NP-complete problem. However, the nature of the constraints on these graphs provide good heuristic solutions.

Some special cliques in this graph are minimal representations for object pose estimation, e.g. a clique of 3 intersecting planes, or a plane intersecting with a cylinder. A feature vector is computed for each of these cliques which can be used for matching. For a clique of 3 planes, the angles between the pairs of planes constitutes the feature vector. For a plane and cylinder intersection clique, the cylinder radius along with the angle between the plane normal and the cylinder axis direction constitutes the feature vector.

The clique matches between the scene and model point clouds generates full hypotheses $h_i \in H$, i.e., it gives the model (M_i) as well as the pose (T_i). Each of these hypotheses gives a set of partial matches for the scene and model graph vertices. Since they are full hypotheses, a fitness score can be computed for each of them which indicates the accuracy of the hypothesis.

The graph matching problem is identical to a vertex labeling problem. For each vertex V_s in the scene graph G_s, a match with a vertex V_m in the model graph G_m can be considered as a label. Hence, this problem can be posed as a multi-label optimization problem, where the *scene graph nodes* are the **nodes** and the *model graph nodes* are the **labels**.

This multi-label optimization problem is formulated as a Quadratic Pseudo-Boolean Optimization (QPBO) [14], [15] problem. In this setting, each vertex consists of a node and its possible label, see Fig.4(a). Thus, the maximum number of nodes in this graph can be $|V_s| \times |V_m|$. Since the node matches are obtained in pairs or cliques, the co-occurring node labels are considered as neighbors in this graph, see Fig. 4(b). The weights for these vertices are obtained from the fitness scores of the hypotheses. By solving this optimization problem, we get the optimal match between the model and scene graphs. This acts like a filtering step which ensures that conflicting hypotheses are removed.

2.1.4 Efficient Hypothesis Verification

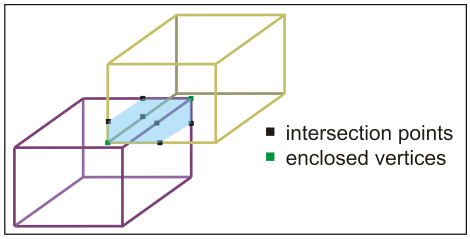

Fig. 5. MVBB intersection volume calculation for Efficient Hypothesis Verification.

Hypothesis verification consists of transforming the model point cloud according to the transformation estimate and calculating how much of it matches with the scene point cloud. Since we use a primitive shape decomposition of the scene

and model clouds, the hypothesis verification step can be simplified. The idea is to utilize this primitive shape decomposition and use it to speed up the point cloud matching step.

Since the model and scene clouds are decomposed into primitive shapes and represented as PSG's, matching these point clouds is equivalent to matching all the primitive shapes in their PSG's. A Minimum Volume Bounding Box (MVBB) [16] is computed for each of these primitive shapes. Matching these primitive shapes can then be approximated by finding the intersection of their MVBB's. The i-th MVBB comprises 8 vertices $v^i_{1,..,8}$, which are connected by 12 edges $l^i_{1,..,12}$ and forms 6 faces $f^i_{1,..,6}$. To find the intersecting volume between MVBB's i and j, the points p^i at which the lines which form the edges of MVBB i intersect the faces of MVBB j are computed. Similarly, p^j are computed. Vertices v^i of the first MVBB which lie inside the MVBB j and vertices v^j of the second which lie inside the MVBB i are also computed. The intersection volume is then the volume of the convex hull formed by the set of points $\left(p^i \cup p^j \cup v^i \cup v^j\right)$. Fig. 5 shows an example of this algorithm, where the volume marked light blue is the intersection volume of the two MVBB's.

The fitness score for this match is the ratio of the total intersection volume to the sum volumes of the primitive shapes in the model point cloud. This score is an approximation of the actual match but the speed-ups achieved by this approximation are more significant compared to the error due to approximation.

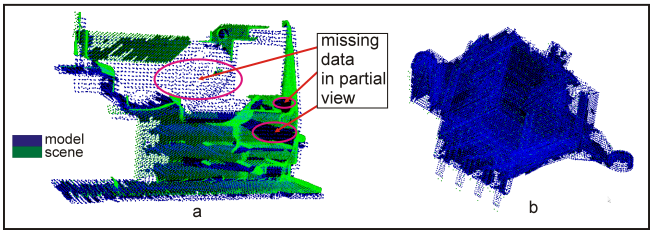

Fig. 6. Example of object recognition and pose estimation using PSORR algorithm: (a) scene cloud containing partial view of object (b) scene cloud containing sparse full view of object.

Fig. 6 shows examples of results obtained using the PSORR algorithm. Fig. 6 (a) shows the case when a partial view of the object is present in the scene. Fig. 6 (b) shows the case where a very low resolution full view of the object is present in the scene. In both cases, the algorithm is able to recognize the object and estimate the pose accurately.

The PSG matching algorithm was tested for full object views at similar cloud resolutions. Fig. 7 shows the results obtained for this experiment. The objects chosen for this experiment were larger than the ones used for the PSORR

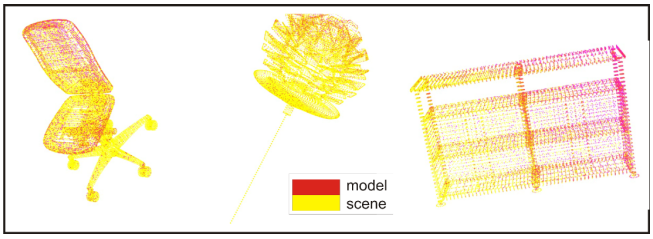

Fig. 7. Results for PSG matching algorithm using full views of object at similar resolutions.

algorithm and their primitive shape decomposition results in a PSG with greater number of nodes (> 20).

2.2 Combining shape and color information

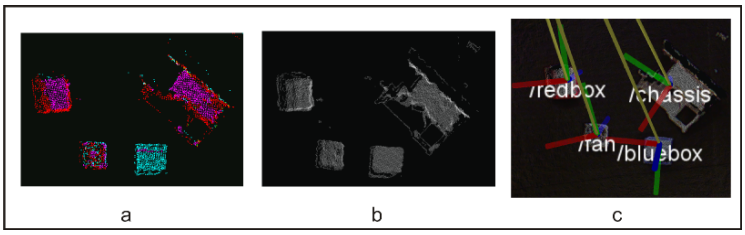

Fig. 8. Example of object recognition using a combination color and shape information: (a) Color Based segmentation (b) Detected Object Clusters (c) Final result of Object Recognition using shape and color information.

A combination of multi-label graph-cuts based optimization [12] and Conditional Euclidean Clustering [10] is used for color-based segmentation of point clouds. Fig. 8 shows an example of object recognition using a combination of color and shape information, where the point cloud is first segmented using color information. Each of these segmented objects is then recognized using the PSORR method described in Sect. 2.1.3. Fig. 8 (a) shows the color based segmentation, Fig. 8 (b) shows the clustered objects and Fig. 8 (c) shows the final recognized objects along with their poses.

3 Reasoning Module

The reasoning module is responsible for representaion of the scene at a semantic level based on an Ontology. Using the semantic representation, this module

can perform first-order-logic reasoning to draw inferences about the scene and its state. This module consists of: a Knowledge Database (KDB) which is an unbounded Ontology that describes the restricted domain of our applications scenarios, a Knowledge Representation (KR) which contains the semantic map of the entities present in the scene, a reasoning engine which can execute first-order-logic queries on the KR.

3.1 Knowledge Database(KDB)

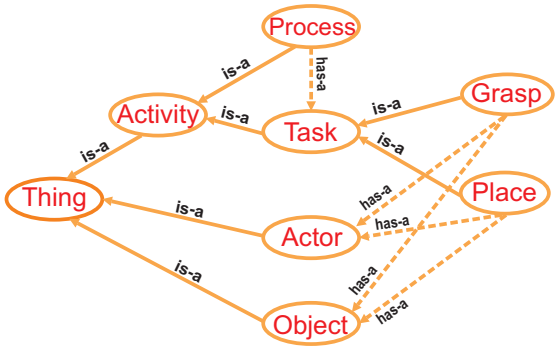

Fig. 9. Knowledge Database

This component is the taxonomy used to describe all entities at a semantic level in the restricted universe of this domain. This is obtained by defining classes in an ontology knowledge base. These ontology classes represent static objects, humans, tools, tasks and plans. Each of these classes contains *data properties(DP)* which store information associated to the instances of the class, e.g. position, dimensions, shape, appearance, etc. The relationships between the different classes and instances of the classes are represented as *object properties* in the ontology class. We use the OWL (Web Ontology Language) format in order to be compatible with Knowrob [1], a knowledge processing framework, which we extend and use as the reasoning module. An example of the Ontology Classes which are used for the *Pick_And_Place* plan in Sec. 4 is shown in Fig. 9.

3.2 Knowledge Representation

The Knowledge Representation is a data container that stores instances of the KDB with data and object properties, and forms the semantic representation of the world state. It includes instances of objects, actors, tasks and plans. As evident from the name, this component acts as a representation of the knowledge about the world which the system possesses. An example of the Ontology Instances which exist in the Knowledge Representation while executing the *Pick_And_Place* Plan in Sec. 4 is shown in Fig 10.

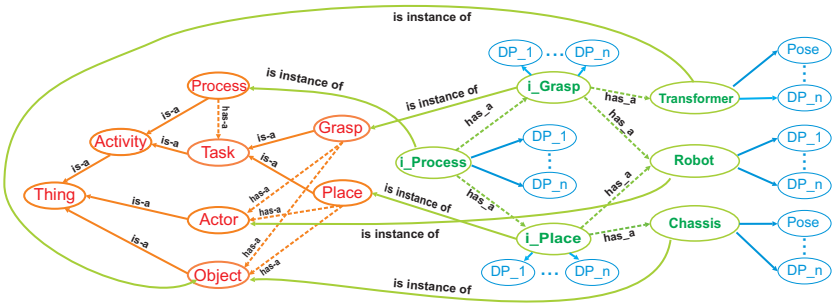

Fig. 10. Knowledge Representation

3.3 Reasoning

The reasoning module receives the system state at a numeric level from the perception modules, converts it to a semantic form and incorporates it into the Knowledge Representation. It creates individuals of the ontology from the KDB and stores them in the Knowledge Representation. The data properties of these individuals are set using the numeric level information obtained from the WSG module. This module contains computables[1] which perform spatial, temporal and object reasoning to infer the object properties for the individuals of the ontology. In other words, this module infers the semantic relationships between individuals. This module is based on the knowrob reasoning engine and the computables are written in first order logic using *PROLOG*.

4 Human-Robot Interaction Applications

A mixed reality interface is created using scene perception and reasoning modules, targeted towards human-robot interaction applications. This interface can be used for teaching process plans at a semantic level (see Fig. 12 (a,b,c)), and execute them in different scenarios without requiring any modifications (see Fig. 12 (d,e,f)). This interface can also be used for executing process plans with both human and robot tasks, see Fig. 12 (g,h,i). Fig. 11 shows an example with the different phases of this interface, where it can be noted that the generated process plan contains semantic names of the objects and not the numeric level data in the form of poses taught to the robot.

4.1 Teaching Process Plans

An articulated Human Tracker provides estimates of the hand positions which are used to control the projected GUI, see Fig. 12 (a). This module enables the user to physically interact with the robot, grab it and move it to the correct

[1] Computables are used to obtain on-demand semantic relations between individuals instead of incorporating every possible knowledge in the ontology

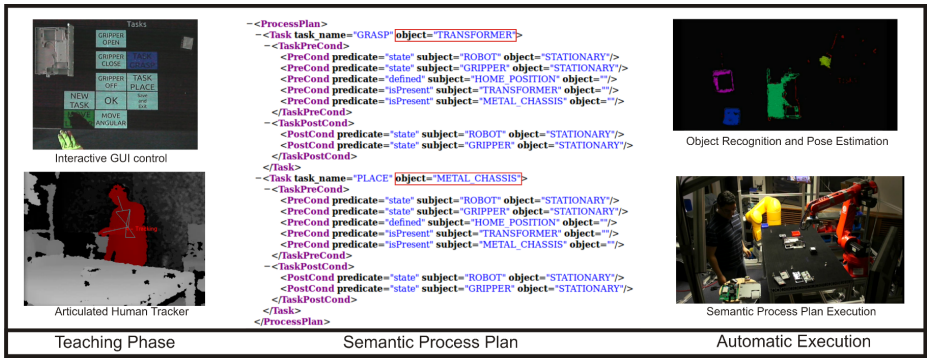

Fig. 11. Overview of Intuitive Interface for Human-Robot Collaboration.

position for grasping and placing objects, see Fig. 12 (b-c). The perception module (Sect. 2.2) detects the objects present in the scene and the reasoning engine (Sect. 3) associates objects with the taught poses to automatically generate a semantic script of this process plan in STRIPS [17] format, see Fig. 11.

4.2 Automatic Plan Execution

The user can place the objects to be assembled anywhere in the working area to begin the plan execution. The system first checks if all pre-conditions for the task are satisfied and informs the user in case something is missing, see Fig. 12 (d). The human can physically interact with the robot during the execution and move it by grabbing its end-effector, see Fig. 12 (e). The user can also add obstacles in the path of the robot, which are detected using the perception module and avoided during plan execution, see Fig. 12 (f). All these interactions and changes in the scenario don't require modifications in the process plan script because object positions and obstacles are scenario-specific entities and, like the physical interaction, are handled at the low-level execution. This is the main advantage of decoupling the Problem Space from the Execution Space. The process plan is generated using only information from the Problem Space. The execution specific parameters are provided by the perception module at runtime.

4.3 Assembly task with Human-Robot Co-operation

In this demonstration, we highlight another important advantage achieved using a semantic description of the process plans - possibility of symbiotic human-robot collaboration, which is one the primary goals in our target applications. Once the robot is taught the *Pick_And_Place* process plan, it can be instructed to perform this plan on different objects. This extension is not as straightforward in conventional industrial robot programming languages, which require explicit object positions and trajectories for creating robot programs. In this example, the process in mind is the assembly of a power converter box. This process

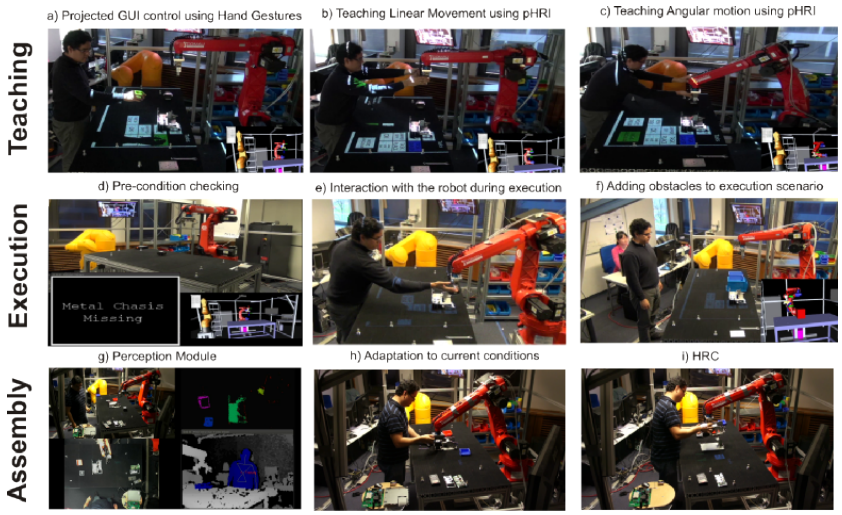

Fig. 12. a,b,c) Teaching Application. d,e,f) Execution and Plan generation of taught Task. g,h,i) HRC in an assembly process.

consists of a number of steps, involving several actors and objects which are identified by the perception/reasoning module, Fig. 12 (g), some of which are complex high precision assembly tasks suitable for the human, while some involve lifting heavy objects which are more suitable for the robot. In the situation where precision assembly is required for a heavy object, a co-operative task is performed where the robot grasps the object and the human guides it by physically grasping the robot end-effector and moving it to the desired place position, Fig. 12 (i). The *Low-Level Execution Engine* switches between motion modalities and control schemes according the current conditions (external perturbations) of the scene, Fig. 12 (h). Thus, in this experiment, we demonstrate the use of this interface for human tasks, robot tasks and co-operative tasks which require both actors. This experiment also highlights that it is relatively easy to understand, edit or even create such a plan from scratch since it is at a semantic level and is abstracted from scenario or execution specific details.

A video illustrating results for the algorithms presented in this paper and its use in the applications mentioned above can be found at :
http://youtu.be/UBy2ceB8ssA.

5 Conclusion and Future Work

The main contribution of this work has been the development of a shape based object detection and recognition module which can handle sensor noise, occlusions and partial views. This module can be trained from CAD models or scanned 3D objects. In the current implementation, planes and cylinders were used for

primitive shape decomposition of point clouds. This could be easily extended for other shape primitives such as torus, spheres or other conics. The primitive shape merging phase supports primitives in general as long as a fitness score and model complexity can be defined. The reasoning framework presented in this work can also be extended to include more computables and perform more complicated reasoning about the scene.

References

1. Tenorth, M., Beetz, M.: Knowrob 2014; knowledge processing for autonomous personal robots. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. (Oct.) 4261–4266
2. Scovanner, P., Ali, S., Shah, M.: A 3-dimensional sift descriptor and its application to action recognition. In: Proceedings of the 15th international conference on Multimedia. MULTIMEDIA '07, New York, NY, USA, ACM (2007) 357–360
3. Sipiran, I., Bustos, B.: Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. Vis. Comput. **27** (2011) 963–976
4. Zhong, Y.: Intrinsic shape signatures: A shape descriptor for 3d object recognition. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE. (2009) 689–696
5. Rusu, R.B., Bradski, G., Thibaux, R., Hsu, J.: Fast 3d recognition and pose using the viewpoint feature histogram. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ. (2010) 2155–2162
6. Hu, G.: 3-d object matching in the hough space. In: Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century. Volume 3. (1995) 2718–2723 vol.3
7. Schnabel, R., Wessel, R., Wahl, R., Klein, R.: Shape recognition in 3d point-clouds. In Skala, V., ed.: The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2008, UNION Agency-Science Press (2008)
8. Schnabel, R., Wahl, R., Klein, R.: Efficient ransac for point-cloud shape detection. Computer Graphics Forum **26** (2007) 214–226
9. Papazov, C., Haddadin, S., Parusel, S., Krieger, K., Burschka, D.: Rigid 3D geometry matching for grasping of known objects in cluttered scenes. International Journal of Robotic Research **31** (2012) 538–553
10. Hastie, T., Tibshirani, R., Friedman, J.: 14.3.12 Hierarchical clustering The Elements of Statistical Learning. 2nd ed. edn. New York: Springer, ISBN 0-387-84857-6 (2009)
11. Gonzalez, R.C., Woods, R.: Digital Image Processing. 2nd edn. Prentice Hall, New Jersey (2002)
12. Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast approximate energy minimization with label costs. Int. J. Comput. Vision **96** (2012) 1–27
13. Leonardis, A., Gupta, A., Bajcsy, R.: Segmentation of range images as the search for geometric parametric models. Int. J. Comput. Vision **14** (1995) 253–277
14. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Appl. Math. **123** (2002) 155–225
15. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary mrfs via extended roof duality. In: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on. (2007) 1–8
16. Barequet, G., Har-Peled, S.: Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. J. Algorithms **38** (2001) 91–109

17. Fikes, R.E., Nilsson, N.J.: Strips: A new approach to the application of theorem proving to problem solving. Technical Report 43R, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025 (1971) SRI Project 8259.