An Odometry-Based Approach for Indoor Object Viewpoint Recognition

Luigi F. Tedesco[1], Céline Craye[2], Jean-François Goudou[3] and David Filliat[4]

Abstract— **Object recognition capability is a essential condition for giving autonomy to mobiles robots in human made environment. However, achieving this goal by means of visually representing objects is a hard task and using all possible sources of information is a must. Here we present a procedure to incorporate the notion of continuity and overcome ambiguous points of view. By observing objects from different perspectives bound with a Markovian modeling of the stochastic processes of recognizing each of the objects viewpoint, the algorithm copes with a sparse database, blurred images from motion and object spatial symmetry, to recognize objects and estimate its viewpoint. A multi-modal Kalman based tracking was also implemented in order to recognize multiple objects simultaneously. The approach was tested in a mobile platform and the comparison between the single viewed and the proposed recognition gave promising results, reinforcing the proposal of blending odometry and recognition.**

I. INTRODUCTION

The vast majority of the literature focus on single image object visual recognition for helping robots in tasks such as semantic navigation **??**, pose estimation for grasping **??** and environmental search **??**. Typically, a set of features is extracted from a segmented object candidate and, subsequently, compared to a database of priori known objects. Extensive work have been done in order to increase efficiency in each one of the sub-processing steps. Among them : segmentations methods using range cameras, features that describe color and texture **??**, geometry **??**, contours **??**, besides classifiers and matching techniques. Alternatively, a deep neural architecture **??** can perform a direct object visual classification after a delicate training phase. However, the classic recognition pipeline seems to be more natural and simple to be implemented with a straight-forward training, still having reasonable results.

Nevertheless, ambiguous viewpoints easily trick visual descriptors reducing its recognition capability. Observing objects sequentially from distinctive points of view seems to be a natural way to deal with the problem. A solution inspired by human behavior for learning new unseen objects has been proposed by **??**, using key-frames and the rate of matching features with past frames, to overcome ambiguity in face recognition task. More work have been done to model objects different viewpoints perspectives summarized by Roy and al. **??**.

[1]Luigi Franco Tedesco is a Master Student with Faculty of Electrical Engineering, Robotics and Artificial Intelligence, ENSTA ParisTech, 91762 Palaiseau, France tedesco@ensta.fr

[2]Céline Craye is a Doctoral Student with Faculty of Visual Saliency, Computer Vision and Developmental Robotics, ENSTA ParisTech, 91762 Palaiseau, France celine.craye@ensta-paristech.fr

[3]Jean-François Goudou is the R&I Project Manager at the VisionLab, ThereSiS, Thales Service, 91767 Palaiseau, France jean-francois.goudou@thalesgroup.com

[4]David Filliat is with the Computer Science and System Ingeneering Laboratory, ENSTA ParisTech, 91762 Palaiseau, France david.filliat@ensta-paristech.fr

II. PROPOSED APPROACH

A. General archtecture

Our approach was designed for robots equipped with odometric and RGB-D sensors. Informations from these units are sent to a processing module that isolates objects from input images, extracts their characteristics through features, and compare them with a reference database stored in memory. When no matching is found between observation and the current database, a new object can be added to the later, increasing the robot's knowledge about the environment.

The architecture of the system is illustrated in Figure 1 and explains the dependencies between the different processing modules and the information flow through them.

Precisely, the processing units takes as input the point cloud from the RGB-D sensor as well as the odometry measure of the two wheels. The first module consists in a segmentation step that cleans up the point cloud by isolating objects of the scene into clusters of points. Those clusters are then sent to the feature extraction module to convert them to discriminative description histograms. Simultaneously, a cartography module converts the odometry data and the segmented object image position into the localization of scene elements in the world referential. Finally, the recognition module uses both the object feature and the evolution of those features compared to the displacement of the robot to retrieve the most likely object and view.

B. Object Segmentation

The segmentation step aims to differentiate objects from the background of raw images. Stereoscopic and infra-red cameras helped the treatment adding a new dimension to images, allowing segmentation geometrically. In the case of motionless sensors, statical background subtraction approach are typically used for segmentaiton [5]. This is not applicable in our case as the robot is constantly moving in its environment. By making the hypothesis that objects are represented in the scene as cluster of points right above a main plan, approaches such as *Tabletop object detector* [6] and Caron et al. [4] determine the main plane's convex hull and search for objects clusters inside it. The later algorithm also removes plans orthogonal to the ground, considered as walls, and is more suitable for finding objects placed on ground level in indoor environments, therefore, explaining our choice for it. Furthermore, in our robot assemble, the

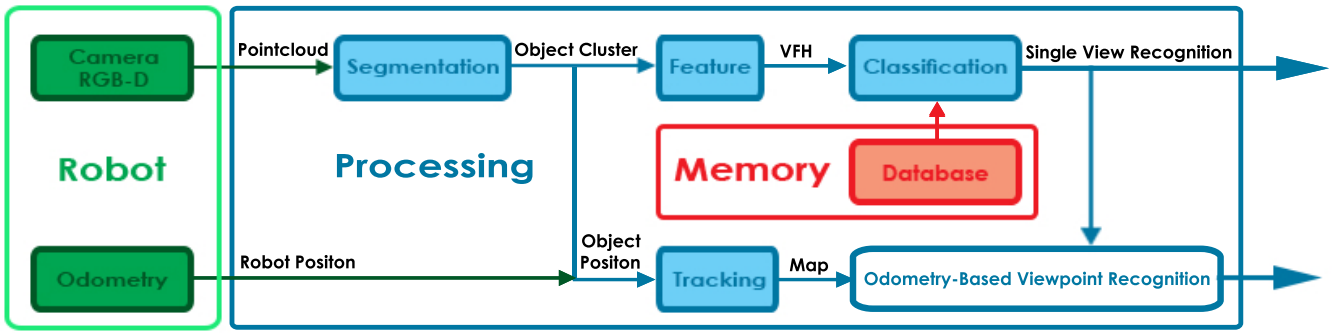

Fig. 1. General achitecture of the system

camera is always keeping the same orientation towards the floor, in this manner, its plan equation can be estimated once during a calibration step and used during the whole experiment. Based on this algorithm, we obtain the position of each object in the camera frame, as well as the associated point cloud and normals.

C. Object Representation

The segmentation module provides us a point cloud for each isolated object in the observed frame. This raw information is however not enough to efficiently discriminate the observed object and retrieve the associated view in the database. From the point cloud, we then extract two types of information. The first dimension of information relates to its position in the environment. This cartographic data allows the robot to track objects while moving around the scene, consequently, associating multiple views of it. A second dimension that can be translated in terms of visual and geometrical features is related to intrinsic characteristic of the objects capable of differentiating them from one another.

1) Features: Visual features used for the task of object recognition are either RGB-based or geometrical. Among RGB features, bag of words or SIFTs or color histograms are commonly used []. Deep features [12] have also received a lot of attention recently, but their high dimensionality requires high computation performance to be processed in real time. Geometrical features based on point clouds are either local descriptor, such as SHOT [11], semi-global such as CVFH [2], [3] descriptors or global [9], [10].

In our system, a global descriptor for each view is the easiest and best suited approach, as we need a way to efficiently compare a large number of view between a database and the current observation. We also restrict our descriptor to geometrical features that we found more discriminative than visual ones. Among global geometrical features, we selected the Viewpoint Feature Histogram **??** - VFH. This feature captures in a single histogram the geometry of the object by estimating the angular transformation between the normal of each of the its points and the standpoint from where it has been viewed. It is therefore discriminative of the object, but also the viewpoint at which it was observed.

2) Multi-view object representation: We define a multi-view representation based on polar aspect-graphs in order to store objects in the memory of the robot. This representation fuses both the intrinsic characteristics of the views of objects as well as the necessary movement to change between them.

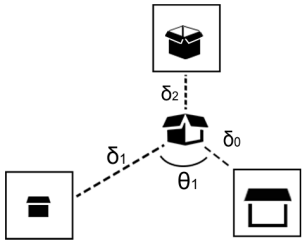

Fig. 2. Multi-view object reprensentation based on a polar aspect graph

For this representation, we used a polar referential with an origin centered at each object, illustrated in 2. Its nodes represent the object aspect from a certain point of view, or in other words, the VFH histogram computed for this view, and transitions from nodes are angular displacements obtained by the robot odometry. The use of VFH instead of viewpoint invariant features is then justified by this type of reprensentation. Indeed, viewpoint information can bias the classification to search for eligible candidates in the database.

D. Odometry-Based Multi-view Recognition

Moving the robot around the environment produce a series of object observations. The idea behind our system is to find consistency between this sequence of possibly misrecognized objects given a known displacement. In other words, tracking the positions of objects and the robot provides us an estimation of the physical transition between two successive viewpoints which can reinforce candidates as they are consistent with the movements prediction. To this end, we use a probabilistic model based on hidden Markov models.

1) Viewpoint recognition based on a single observation: Regardless odometry and cartographic data, a single observation provides a VFH histogram for each segmented object. Determining the viewpoint of the object consists on finding the closest histogram in the database. To this end,

the work from Aldoma et al. [1] suggest the use of a K-Nearest Neighbors, based on a chi-squared distance between histograms. The K-NN classifier has the great advantage of having both classification and training stages extremely fast for the intended database size. Thus, the closest viewpoint V from the observation O is retrieved based on the following equation:

$$\begin{cases} V = \underset{v_1 \dots v_k}{\operatorname{argmin}}\ d(O, s_k) \\ d(O, s_k) = \sum_i \frac{\left\| H_O(i) - H_{s_k}(i) \right\|^2}{H_O(i)} \end{cases} \tag{1}$$

with $H_p(i)$ the ith bin of the VFH histogram of a point cloud p and $s_1, \dots s_k$ are the different views available in the database. Moreover, the distance $d(O, s_k)$ between an observation O and a view V in the database can easily be turned into a probability $P(s_k|O)$

$$\mathbf{P}(O|s_k) = \frac{d(O, s_k)}{d_{max}} \tag{2}$$

where d_{max} is the maximum distance between two views of the database.

2) Object Tracking and Angular Displacement Estimation: The segmentation algorithm provides us at each new frame a set of point clouds (one for each object) and associated centroid position in the camera sensor reference frame. On the other hand, we receive a signal of the speed of the wheels of the robot that we convert into a displacement estimation in the absolute reference frame. Therefore, we can convert both the position of the robot and point cloud centroids to the absolute reference frame. Nevertheless, imprecision from communication delays between processing modules, segmentation and object centroid calculation call for a filtering method to refine the absolute position of the observed objects. Moreover, because of the segmentation restrictions and constant displacement of the robots, objects are alternatively present and absent from the input image.

For that reason, we rely on a multi-tracker based on Kalman filtering that both refines the positions of each observed centroid and keeps an history of all observations and angular displacement for each tracked object from the beginning of the experiment.

The multi-object tracker is composed with a set of Kalman filters, each of them tracking a single object and being updated as soon as a new observation of the object is found. Briefly, the update rule is done as follows: For each new frame, the centroid of all segmented objects are converted to the absolute reference frame. Then, their positions are compared with the position of the last observation of each available tracker (initially, no tracker is available). If the position of a centroid is close enough to the last observation of a tracker, the tracker is updated using Kalman update equations, and the pointcloud is added to the tracker history. If a centroid does not match with any Kalman position, the centroid is considered as a new object and a dedicated tracker is initialized.

This multi tracker relies on the two asumption that objects of the scene are static (although a dynamic model could be added) and that objects are separated of a minimum distance to avoid confusions between trackers.

Between two consecutive views $k-1$ and k of the same object, an angular displacement is estimated to store the tracked object history with a polar aspect-graph representation (See section II-C.2). The angular displacement is obtained by the following equation

$$\delta_{k-1,k} = atan(p_{k-1} - p_{obj}) - atan(p_k - p_{obj}) \tag{3}$$

with p_{k-1} and p_k the absolute position of the robot for views $k-1$ and k, and p_{obj} the absolute position of the centroid of the tracked object.

3) Odometry-reinforced viewpoint recognition: Based on the sequence of observations and estimated angular displacement obtained with the odometry, we use Hidden Markov Models to retrieve the closest sequence of views in the database. This sequence not only provides us a reinforced recognition of the object, but also an estimate of the successive views at which the object was seen. The definition of our approach is inspired from [7].

Given a track T associated with k observations $O_1, \dots, O_k$ and $k-1$ angular transitions $\delta_{1,2}, \dots, \delta_{k-1,k}$ between each of them, we model the viewpoint of the object from the robot sensor tracked by T at time k by a random variable V_k that follows a Markovian process.

Based on the notations used in [8], we define our HMM as $\lambda = \{N, M, \Pi, A, B\}$, with N the number of states, M the number of observations, Π the initial state vector, A the transition probability matrix, and B the observation probability.

In our case, a hidden state is associated with each viewpoints s_i, $0 < i \leq N$ in the database. N is therefore the number of views in the database. As no prior assumption is made about the initial state of the system, we set Π as a vector of length N with equal probabilities.

The state transition probability matrix $A = \{a_{i,j}\}$ is defined as

$$a_{i,j} = \mathbf{P}(V_k = s_i | V_{k-1} = s_j) \tag{4}$$

or in other words, the probability of seeing view s_i at observation k given view s_j at observation $k-1$. This probability depends both on the angular displacement $\delta_{k-1,k}$ between the two observations, and the angular distances $\theta_{i,j}$ between views s_i and s_j in the database. Therefore, the transition probability is modified as each observation k and defined as

$$a^k_{i,j} = \begin{cases} 1, & \text{if } \theta_{i,j} < 2\delta_{k-1,k} \\ \varepsilon_1, & \text{if } Obj(i) = Obj(j) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

In practice, we consider that if s_i and s_j belong to a different object, their angular distance is infinite. A small probability ε_1 is attributed to unlikely transitions between

same object views to reinforce objects recognition rather than pose consistency. Transitions between views of different object are not allowed with zero probability of happening.

Last the observation probability matrix $B = \{b_j\}$ is also constructed from each new observation. The distance from the K-NN is transformed in a probability distribution of the K neighbors.

$$b_j^k = \begin{cases} \mathbf{P}(O_k|V_k = S_j), & \text{if } s_j \in D_K \\ \varepsilon_2, & \text{otherwise} \end{cases} \quad (6)$$

where D_K is the set of K nearest neighbors and $\mathbf{P}(O_k|V_k = S_j)$ is the converted histogram distance into probability, defined by equation 2. If the state view S_j is not among the K nearest neighbors, a small probability ε_2 is associated instead.

Last, the Viterbi algorithm is used to find the most likely sequence of views $\mathbf{V}$ given the HMM λ, and the sequence of observations $\mathbf{O} = \{O_1, ...O_M\}$.

$$\mathbf{V} = \underset{\mathbf{S}}{\text{argmax}}\ \mathbf{P}(\mathbf{S}|\mathbf{O}, \lambda) \quad (7)$$

The Viterbi path of the track with observations $\mathbf{O}$ then coincides with the sequence $\mathbf{V}$ of recognized view from $\mathbf{O}$. 8. The optimization is solved using the following equations

$$\begin{aligned} V_{1,i} &= b_k^1 \cdot \pi_i \\ V_{k,i} &= \max_{j \in \{1,...N\}} \left(b_k^i \cdot a_{j,i}^k \cdot V_{k-1,j} \right) \end{aligned} \quad (8)$$

where π_i is the probability of initially being in ith state (from Π vector), and $V_{k,i}$ is the probability that the most probable sequence based on the k first observations ends up in state s_i. The most probable sequence is then retrieved by taking the sequence of V of successive states used to compute $V_{M,i}$ so that

$$s_i = \underset{s_j, j \in \{1,...N\}}{\text{Argmax}}\ (V_{M,j}) \quad (9)$$

III. Experimental results

The proposed recognition system was deployed in a differential mobile robot, the Wifibot V2, embedded with a RGB-D camera, Asus Xtion Pro Live. The algorithm architecture were implemented over ROS using PCL and OpenNi2 libraries.

In the interest of validating the approach, the robot was initially taught aspects graphs from chosen objects and two sets of experiments were proposed to analyze the efficiency of the algorithm in real scenarios.

A. Object Database

First, twenty objects varying in size and form (monitor, boxes, chairs, trashcans, people, ...) were selected to compose the knowledge database of the robot. The aspect graphs of each object weas composed by VFH features from *eight* equally distant viewpoints acquired from positioning the robot around the to be learn object 1.5 meters away.

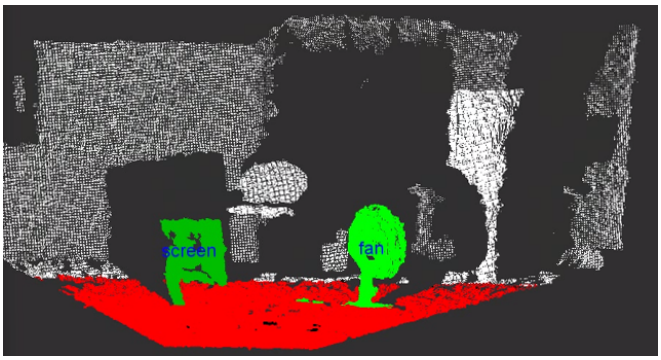

Fig. 3. **Single-image recognition** - Object identification for two segmented objects - a computer monitor and a small electric fan. The red points represent the floor plan and the ignored white ones exceed the three meters threshold. One could also notice the infra-red shadows created by objects that possible can occlude others

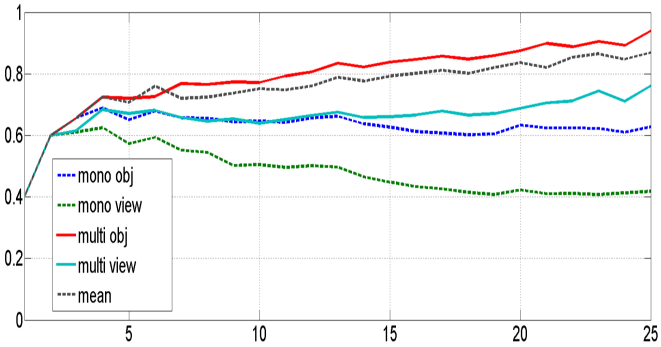

Fig. 4. **Evaluation result** - Dashed curves represent object and view recognition on single image basis. Solid ones are obtained based on multi-view estimation. When the number of observation is low, the mono-view recognition tends to be slightly better than multi-view. The gap between mono and multi-view recognition increases with the number of observations. With 25 or more observations, the multi-view recognition outperforms the mono-view by 33% with 92% of correct recognition for object recognition and 74 % for view estimation.

B. Odometry Contribution

The first experiment consist in a performance comparison between the single image and odometry-base recognition techniques. In other words, this comparison attest whether the proposed architecture is interesting or not, since having the same or worst performance by the cost of adding a complex post-processing and tracking modules is not interesting.

Technically, each object was partially circled by the robot from at least four random initial positions at speed of $0.35 \pm 0.1 m/s$. The robot recorded for each run between 5 and 30 frames at different viewpoints of the object, perhaps inexistent in the database, and try to recognize them using both recognition algorithms.

A typical experiment is illustrated in Figure 5. The first row represents the image sequence segmented by the robot for a given object at each new frame. This is the object to be recognized. The second line, is the viewpoint recognition based on a single image, without using any odometric information. The only measure used to recognize the view is provided by the distance to the nearest neighbor in the database (see section II-D.1). Interesting is to notice that the

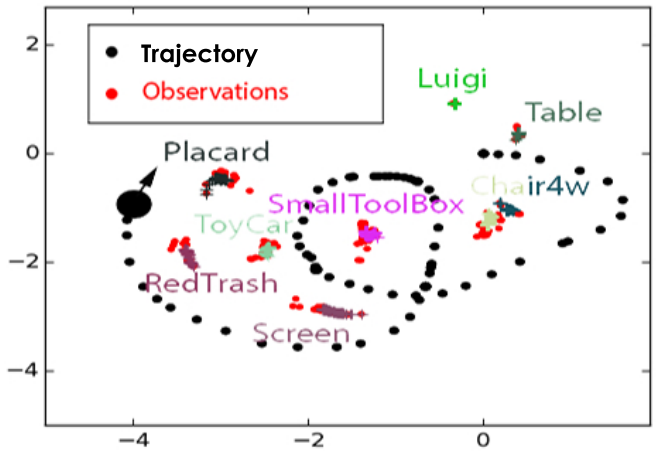

Fig. 6. **Mutli-objects tracking results** - Red dots corresponds to segmented objects centroids. Each color cross stands for the position estimation from a tracker. Each filter is associated with a unique color. Theoretically, a single filter should be associated with a single object. Black dots represent the trajectory of the robot during the experiment.

rotation invariance of the VFH descriptor actually tricks the view estimation by taking the enantiomorph correspondent in the first red square. In addition, the back of the penguin is a large and almost flat surface. It is, therefore, considered as background wall by the segmentation module and almost entirely removed. The resulting point cloud is insufficient to accurately characterize the view, thus leading to wrong view detection in the blue squares. The last row represent the final output of the Viterbi algorithm. As expected, the combination of VFH-based view estimation and odometry is able to retrieve the correct sequence of views.

C. Multi-object recognition

In a second evaluation, concurrently with the recognition efficiency, the multi-tracking capabilities were put into test. Tracking objects correctly is extremely important for the well functioning of the system.

The experiment resides in moving the robot around a room full of objects, ensuring it visualize them from different perspectives. This scenario is a much more complex than the first one as moving in a room with many elements,likely occlude each other, thus leading to incomplete point clouds and biased feature descriptors. Even if tracking was correct in most cases, the result was a drop in recognition performance for both algorithms.

IV. Conclusion

The proposed method focus on integrating odometry information to boost recognition efficiency, based on the intuition that different perspectives of the object and analysis of transitions between them provides unexplored informations. The results from practical experiments shows that the odometry-based algorithm outperforms the classic one in both object and viewpoint recognition, strengthening the interest in a more sophisticated pipeline. Moreover, the approach has the advantage of not being platform specific, which means it could be implemented in any mobile system equipped with a

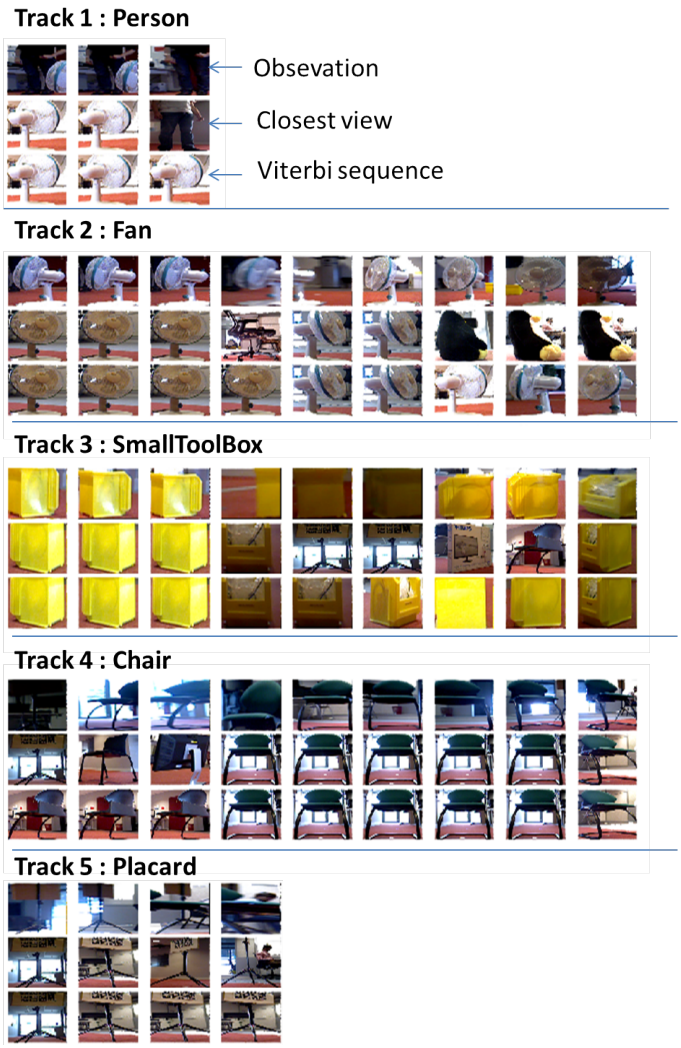

Fig. 7. **Reconnaissance Multi-cible** - Quatre des cinq objets prsents dans la scne ont t correctement reconnus avec une estimation d'orientation raisonnable. Le premire objet (une personne) tait trop prs du ventilateur, les deux ont donc t confondus dans le multi-tracking.

RGB-D sensor and capable of measuring its position. However, the algorithm is quite sensitive to bad segmentation, specially if a object is divided into two disjoint clusters. Errors due to occlusion could be reduced using semi-global descriptors and a more exigent segmentation. Future work consist in recognizing moving objects, where the recognition feeds the tracking module with the moving dynamics of the target, and including object transformation by allowing state transitions to views of other objects [1]. A more immediate challenge is to create the object database on the fly once a new observation is not encountered on it, according to a distance threshold.

References

[1] A. Aldoma, Zoltan-Csaba Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point

[1] i.e. human moving towards a chair and sitting down, creating a new object human+chair

Fig. 5. **Typical experiment** - The HMM refinement makes self correction of irrelevant views estimation, even when objects were badly segmented during the database creation.

cloud library: Three-dimensional object recognition and 6 dof pose estimation. *Robotics Automation Magazine, IEEE*, 19(3):80–91, Sept 2012.

[2] Aitor Aldoma, Federico Tombari, Radu Bogdan Rusu, and Markus Vincze. *OUR-CVFH–Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*. Springer, 2012.

[3] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 585–592. IEEE, 2011.

[4] Louis-Charles Caron, David Filliat, and Alexander Gepperth. Neural network fusion of color, depth and location for object instance recognition on a mobile robot. In *Computer Vision-ECCV 2014 Workshops*, pages 791–805. Springer, 2014.

[5] Ke-xue DAI, Guo-hui LI, Dan Tu, and Jian YUAN. Prospects and current studies on background subtraction techniques for moving objects detection from surveillance video. *Journal of Image and Graphics*, 11(7):919–927, 2006.

[6] Ros documentation. Tabletop object detector.

[7] Cédric Le Barz, Nicolas Thome, Matthieu Cord, Stéphane Herbin, and Martial Sanfourche. Global robot ego-localization combining image retrieval and hmm-based filtering. In *6th Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, pages 6–p, 2014.

[8] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[9] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[10] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010.

[11] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer, 2010.

[12] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.