BLORT - The Blocks World Robotic Vision Toolbox

T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich and M. Vincze

***Abstract*— The vision and robotics communities have developed a large number of increasingly successful methods for tracking, recognising and on-line learning of objects, all of which have their particular strengths and weaknesses. A researcher aiming to provide a robot with the ability to handle objects will typically have to pick amongst these and engineer a system that works for her particular setting. The work presented in this paper aims to provide a toolbox to simplify this task and to allow handling of diverse scenarios, though of course we have our own particular limitations: The toolbox is aimed at robotics research and as such we have in mind objects typically of interest for robotic manipulation scenarios, e.g. mugs, boxes and packaging of various sorts. We are not aiming to cover articulated objects (such as walking humans), highly irregular objects (such as potted plants) or deformable objects (such as cables). The system does not require specialised hardware and simply uses a single camera allowing usage on about any robot. The toolbox integrates state-of-the art methods for detection and learning of novel objects, and recognition and tracking of learned models. Integration is currently done via our own modular robotics framework, but of course the libraries making up the modules can also be separately integrated into own projects.**

I. Introduction

Even with the large pool of powerful methods for learning, recognition and tracking of objects available today, putting together a system that "simply works" can be a time-consuming task. Objects need to be recognised and tracked, and learning of models should not be too cumbersome, if possible on-line and require no extra hardware besides the robot itself. Simple as well as complex scenes should be covered, with no need to either place markers on objects or paint them in bright colours. A sufficiently large number of objects should be detectable and trackable in real-time, with no constraints on object type or shape. And if the task requires manipulation full 6D object pose and shape are required.

Recent increasingly successful approaches in object recognition and tracking are typically based on some variant of interest points and a combination of offline training and online recognition. Training often requires either hand-labelling of images or presentation of objects on specialised equipment like turn tables, which is sometimes not desirable. Approaches for learning by showing use e.g. optical flow for segmenting the object of interest from the background and then train an interest points-based model, accumulating new interest points as the object is rotated. These approaches however typically create a "sparse" 3D model consisting of several interest points rather than a model of the actual physical surface. Calculating points of contact for grasping or touching an object however requires a dense surface model such as a triangle mesh. Furthermore not all methods are amenable to real-time tracking, which is needed as soon as the robot or a human handles objects and the scene becomes dynamic.

T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich and M. Vincze are with the Vienna University of Technology, Vienna, Austria, email: [moerwald, prankl, arichtsfeld, zillich, vincze]@acin.tuwien.ac.at

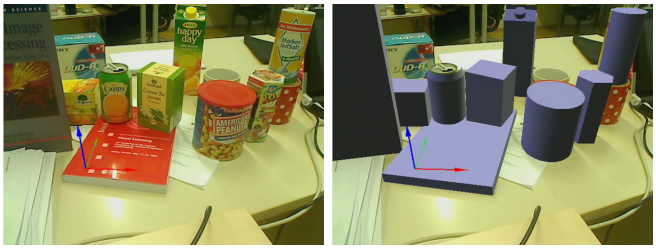

Fig. 1. Scene and recognised/tracked objects

With BLORT, the blocks world robotic vision toolbox, we combine state-of-the-art methods for learning, recognising and tracking objects modelled as dense triangle meshes. Fig. 1 shows an example of a scene containing several (partly occluded) objects and their overlaid 3D models after recognition. Specifically we employ a combination of edge-based detection of basic geometric shapes, fast edge-based monte carlo particle filter tracking and SIFT-based recognition. The rationale behind this particular choice of methods is twofold. First we want to enable human tutor driven learning-by-showing as well as completely automatic on-line model acquisition by the robot. For the latter we have to make simplifying assumptions to allow segmentation of unknown objects and it is here where blocks world enters: detection of novel objects is limited to simple geometric shapes, namely cuboids and cylinders. While this is admittedly a considerable limitation, it allows experiments where a robot interacts with a (simple) scene fully autonomously. More complex objects can be acquired in a learning-by-showing manner but require known 3D triangle mesh models to start with. Second, using edge-based methods covers both textured and untextured objects. Tracking will be improved by available surface texture but works satisfactorily without. SIFT-based recognition obviously requires texture but can be substituted with the edge-based detection.

As a motivating example let us consider a scenario where a human tutor shows a new object to the robot within a learning setup that is intended to make things easy initially, i.e. the tutor basically puts the object on a table and says something like "This is a tea box." The system detects this new object

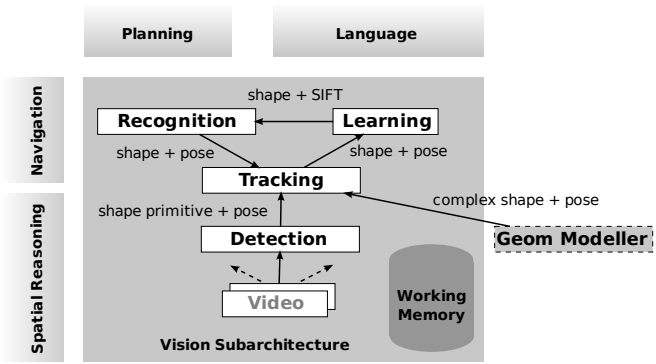

Fig. 2. System overview: interplay between detection, tracking and recognition

and starts tracking it. The tutor then picks up the object and shows it from different sides while the system learns the different object views. Alternatively the robot could go round the object itself, e.g. using an arm-mounted camera. The system is then able to recognise the learned object and re-initialise the tracker in more general scenes, with all the background clutter and varying lighting that are typical of robotic scenarios.

Fig. 2 shows the interplay between detection, tracking and recognition within our robotics framework, where other parts of the system concerned with localisation, planning, language understanding etc. are only hinted at. The framework is based on a software architecture toolkit [1] which handles issues like threading, lower level drivers and communication via shared working memories. Tracking starts when a new object is detected automatically or a model is supplied explicitly. While the object is tracked, initially only based on its wireframe model, surface texture and SIFT features are collected and mapped onto the object surface. This builds up a SIFT-based model that is used later to re-initialise the tracker.

The remainder of this paper is organised as follows. After reviewing related work in Section II we describe detection of novel objects in Section III, followed by tracking in Section IV and recognition in Section V. Experimental evaluation is presented in Section VI followed by conclusion and outlook in Section VII. Note that size constraints prevent us from going into the details of the involved methods and the reader is referred to referenced previous work. The focus of this paper lies on showing the interplay between methods.

II. Related Work

There are many approaches for obtaining 3D object models. The simplest option would be to just download a model from the rich selection on Google 3D Warehouse, if the particular object happens to be modelled (which is quite likely for things like coke cans or bottles). Professional solutions for building 3D models from multiple images such as 2D3® exist but tend to be costly. Systems like the low-cost laser scanning system by Winkelbach et al. [2] or the projected-light system by Rusinkiewicz et al [3] allow quick and easy capture of full 3D models but require special hardware setups.

A very convenient way to construct models is by simply showing an object and adding model information as it is rotated. The system by Brown et al. [4] builds 3D wireframe models of simple geometric objects. The user however has to initialise a 2D wireframe model by selecting image lines and model acquisition takes around 5 minutes. Vacchetti et al. [5] track a 3D CAD model (which has to be specified in advance) and augment it with distinctive features, first from user supplied keyframes in a training phase and also online while tracking. Only a very rough 3D model (e.g. just an ellipsoid) is needed for the system by Özuysal et al. [6], which learns object models for tracking-by-detection by "harvesting" features. The user aligns the model with the first image where keypoints based on randomised tree classifiers are learned and mapped to the 3D model. New features are successively added while the object is tracked using the available features. In [7] the same authors use marking of the outline of a planar object in a single training image in a similar tracking-by-detection approach. Tracking-by-detection has the advantage that lost tracks are recovered immediately. The models are however somewhat costly in terms of memory (in the order of hundred MB) which quickly becomes an issue even on modern hardware. Grabner et al. [8] use a similar approach of accumulating keypoint classifiers while tracking. Only a region of interest near the object center is needed for initialisation. The approach requires no explicit 3D model but also assumes planar objects. Roth et al. [9] use background subtraction to initialise an MSER-based tracker and incrementally learn a PCA model of the object while it is tracked. These models however do not represent 3D shape as would be needed by typical robotics tasks. Riemenschneider et al. [10] take a similar approach of tracking based on MSER but learn a model based on a SIFT codebook. Again however the models do not represent 3D shape. Pan et al.'s ProFORMA system [11] allows to interactively build high quality dense triangle meshes by reconstructing and tracking a model while it is rotated by the viewer. No constraints are placed on the type of object to be modelled other than it be textured.

Gordon and Lowe [12] build a 3D model composed of SIFT descriptors in an offline training phase by performing structure and motion estimation. The online phase then uses RANSAC to estimate 6D pose from 2D-3D correspondences. The system though is geared at augmented reality applications and the scene is not segmented into objects. Collet et al. [13] extend the above for application in the robotics domain, specifically by augmenting RANSAC with a Mean-Shift clustering step to allow recognition of multiple instances of the same object. The system does require manual segmentation of the object in each training image though. Furthermore the obtained sparse 3D points model has to be manually aligned with a CAD model of the object, so the whole procedure requires considerable user intervention.

Tracking of 3D object models has a long history (see Lepetit and Fua [14] for an overview) and we are here

mostly concerned with approaches that use surface texture or combine it with model geometry. Masson et al. [15] use point features from surface texture in conjunction with edges to increase robustness especially with respect to occlusion. Rosten and Drummond [16] present a similar approach fusing FAST features with edges. Both of the above treat edges and texture point features independently and fuse them explicitly. Klein and Murray [17] take advantage of the large processing power of recent GPUs for tracking a wire-frame model using a particle filter. Murphy and Trivedi [18] follow a similar approach but use surface texture by computing the cross-correlation of pixel patches.

Detection of geometric shapes based on perceptual grouping of edges is a well known topic in computer vision with an abundance of literature since the eighties. Approaches such as [19], [20] and [21] use groups of edge fragments to detect learned classes of shapes and show impressive results on databases. Our models differ in that they are only topological models of generic views of basic shapes such as an arrangement of lines and ellipses forming a cylinder, where the limited number of these shapes allows us to ignore learning. Dickinson and Metaxas [22] combine qualitative and quantitative object models to detect and track ten object primitives (box, cylinder, tapered cylinder etc). That system is still brittle in the presence of texture and clutter. To this end Sala and Dickinson [23] describe objects beyond simple shapes but still of limited complexity (cups, hats, desks) with qualitative, parts-based shape abstraction base on a vocabulary of 2D part models corresponding essentially to closed contours of various shapes. Their system can extract such representations from images containing textured objects as well as complex backgrounds.

III. Detection of Novel Objects

Detection of unknown objects, with segmentation from a possibly cluttered background is a notoriously difficult problem and we thus have to make several simplifying assumptions. We use a robot mounted camera which allows us to at least assume the ground (or table) plane and that objects for training initially rest on this ground plane. Furthermore we restrict our search to objects belonging to simple shape classes (cuboids and cylinders) which are detected in generic views from edge images. Detection is based on an incremental perceptual grouping approach and outputs 2D projections of shape primitives. We then use the ground plane constraint to generate 3D wireframe models.

A. Incremental Indexing and Anytimeness

Perceptual grouping is based on previous work of Zillich and Vincze [24] which provides an anytime solution to finding junctions between edge segments and subsequently closed contours avoiding the need for arbitrary distance thresholds and Richtsfeld and Vincze [25] which adds higher level primitives such as cuboids, cylinders and cones. Indexing is used to efficiently identify candidates for junctions, where the indexing space is the image itself. Each edge endpoint defines a set of search lines consisting of tangential and normal search lines. These search lines are drawn into the index image using Bresenham line drawing, essentially voting for junctions. Whenever two lines index into the same bin, i.e. their search lines intersect, we create a new junction. Depending on the types of search lines intersecting we form an L-junction, a collinearity or a T-junction between the respective originating lines. If more than two lines intersect, the according number of pairwise junctions are created. Shortest path search in the resulting graph consisting of edges and junctions then finds closed contours.

In order to avoid the definition of certain length thresholds for search lines they are drawn incrementally, continuously checking for junctions. So the longer we search, the more junctions and eventually closed contours will be found, where “easy” cases typically pop out fast and “difficult” ones (broken edges, partial occlusions, more clutter) follow later. This allows us to stop processing any time, e. g. after a certain frame time has elapsed or, if we happen to know that we expect precisely three cylinders in the scene, after having found three cylinders.

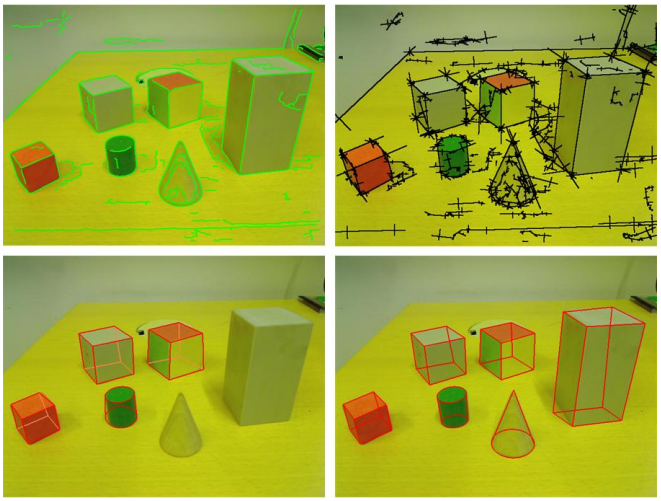

Fig. 3. Edge image, voting image with growing search lines and object shapes after 300 ms and 450 ms processing time.

B. Perceptual Grouping

We then define a hierarchy of grouping principles to enable efficient abstraction of image edges into basic geometric primitives. Triggered by the incrementally growing search lines referred to in the above section, lower level primitives such as closures or ellipses are formed and in turn trigger formation of higher level primitives such as cuboids and cylinders as seen in Fig. 3. Concretely cuboids are defined as three rectangles pairwise connected along an edge. Cylinders are defined as two ellipses and two parallel straight lines. Note that as we move up the abstraction hierarchy the corresponding primitives get more and more distinctive. So while we will generally find lots of closures, rectangles and ellipses are already somewhat less frequent. Finally cuboids comprised of three rectangles or cylinders being composed

of a specific topological arrangement of lines and ellipses already rarely appear accidentally.

C. From 2D to 3D

The following tracking procedure in Section IV requires a 3D wire-frame model of the detected object shape as well as an initial pose estimate relative to the camera. Note that everything so far is purely 2D, i. e. we detect projections of shapes in generic views onto the image plane. Assuming a camera with known elevation and tilt angle and further assuming that detected objects (cubes, cones, cylinders and spheres) rest on the ground, allows us to convert them to 3D shapes. We intersect view rays with the ground plane and thus obtain 3D position on the plane as well as unambiguous size and fill in the unseen backsides from simple symmetry considerations.

IV. Tracking Known Objects

Tracking is based on edges and a particle filter for 6D pose estimation. The tracker uses geometry edges (surface discontinuities and contour edges) as well as edges resulting from surface texture (if present) both of which are treated the same, i. e. we make no explicit distinction between tracking edges and tracking texture. Texture is mapped onto the surface while tracking as part of the learning procedure. Texture edges are generally more numerous and thus lead to improved robustness. All time-consuming parts of the tracker (such as matching of edge images) are implemented on a GPU allowing tracking at frame rate. More details can be found in Moerwald et al.[26] and Richtsfeld et al. [27].

The algorithm for tracking is illustrated in Fig. 5. It is a modified version of the well known bootstrap filter in Doucet et al. [28] applied to vision based object tracking. With respect to computational costs it can be separated into *image processing* and *particle filtering*.

A. Image Processing

In the following an object is described by the geometry of its surface S (approximated by polygons and vertices $\mathbf{v}$) and its 6 DOF pose $\mathbf{x} = [R \,|\, t]$. Furthermore with the information of the geometry S and the initial pose $\mathbf{x}_d$ of the object as described in Section III the colour texture I_S can be retrieved from the camera image I_C and mapped onto the surface S.

In the image processing stage shown in Fig. 4 first the edge image of the incoming camera image is computed. This is to be compared with the edge images of all projected object hypotheses represented by the particles to finally weight each particle with a matching score. Rendering each hypothesis with its texture is fast using the GPU but edge detection on each rendered image would take too long. On the other hand storing thinned edge images as surface texture instead of normal colour texture leads to bad aliasing effects for smaller projected scales. So we forward-project the colour texture once using a "representative" current pose (the weighed mean of all particles), do edge detection on the rendered image and back-project the edge image as (temporary for this frame) surface texture. Thus rendering of the edge texture happens

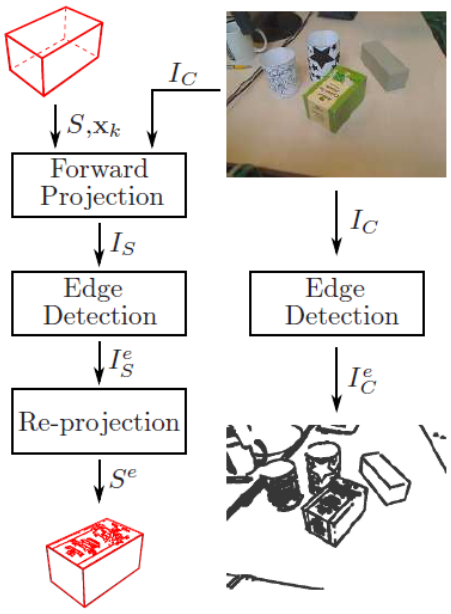

Fig. 4. Block scheme of image processing. The decision whether a face uses texture information or not depends on the viewing angle and confidence level c_k of the tracking step.

at or near the appropriate current scale and aliasing effects are drastically reduced.

B. Particle Filtering

For each tracking step the particle filter executes the methods shown in Fig. 5. First the particles $\mathbf{x}_0^i, i = 1, \ldots, N$, each representing a different pose hypothesis of the object, are generated using Gaussian noise. Then the confidence level c_k^i and importance weight w_k^i of each particle $\mathbf{x}^i$ are evaluated by matching its corresponding edge image against the edge image of the camera I_C^e. According to the importance weights the set of particles is resampled and then perturbed using again Gaussian noise.

The loop formed by the blocks "Importance Evaluation" and "Resampling with Replacement" is executed several times (2-5 times depending on the power of the processor). We refer to this as Iterative Particle Filtering. It increases accuracy and increases robustness especially in cases where the object is moving fast.

First particles $\hat{\mathbf{x}}_k^i$ are resampled from the prior particle distribution $\mathbf{x}_{k-1}^i$ according to the importance weights. Then $\hat{\mathbf{x}}_k^i$ is perturbed by adding Gaussian noise with a standard deviation scaled by the prior confidence level c_{k-1}^i. Each particle is tested against the camera image and its current confidence level is calculated. To this end the correlation between the gradients of the edges of the camera edge image and the particle edge image is evaluated by comparing the direction of the edges at each image point (u, v), producing the edge correlation image Φ_i. The image Φ^i now contains the degree of correlation between the pose suggested by the particle i and the camera image. The angular deviation of the edge angles Φ^i is scaled to the range of 0 to 1.

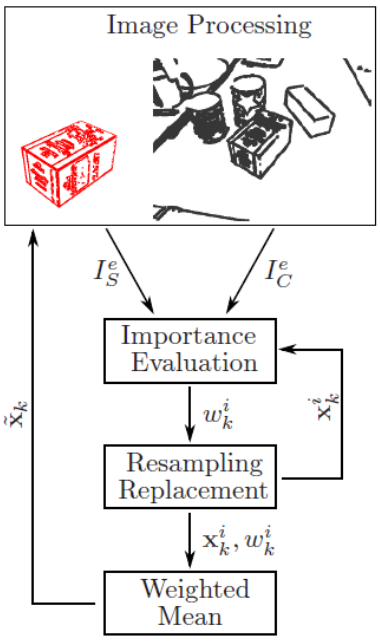

Fig. 5. Block scheme of particle filtering

The confidence level c^i and importance weight w^i are evaluated as follows:

$$\begin{aligned} c_k^i &= \frac{1}{2}\left(\frac{m^i}{n^i} + \frac{m^i}{n_{max}}\right) \\ w_k^i &= (c_k^i)^p \end{aligned} \tag{1}$$

with

$$\begin{aligned} m^i &= \sum_{u,v} \Phi^i(u,v) \\ n^i &= \sum_{u,v} |I^e_{S^i}(u,v)| \\ n_{max} &\propto \max(n^i | i = 1, \ldots, N) \end{aligned}$$

The first term within the brackets of Eq. (1) is the percentage of matching edge pixels m^i with respect to the total visible edge pixels n^i. Calculating the confidence level only with this term would cause the tracker to lock when only one side of the 3D object is visible. If in this degenerate view the object is rotated slightly, another side of the object becomes visible. The particle representing this rotation would typically get less weight than the prior front facing particle as the number of matching pixels m^i grows slower than the number of non-matching pixels n^i when rotating the object out of the front side view. This effect is amplified by the fact that edge detection for strongly tilted faces is less accurate. The second term allocates more weight to the total number of matching pixels m^i which is intrinsically higher for the rotated particle. n_{max} is the maximum number of visible edge pixels in the actual area and scales the pixels to the proper range.

The weights of the particles are calculated by raising c_k^i to the power of p, which controls the speed of convergence of the particles. With a higher power p, w_k^i increases, which leads to more particles assigned to $\mathbf{x}_k^i$ when resampling the whole particle distribution and therefore to a faster convergence. As explained in Section IV-A for projection and re-projection of the model, a single representative pose $\tilde{\mathbf{x}}_k$ is required, where we use the weighted mean of the particle distribution.

V. Learning and Recognising Objects

While edges are well suited for fast tracking we use highly discriminating SIFT features for recognition (where again we use a GPU implementation [29]). For recognition we follow a standard approach similar to Gordon and Lowe [12] and Collet et al. [13] but our training phase differs in that we do not build a sparse 3D SIFT point model via bundle adjustment but use the 3D pose and object geometry already provided by the tracker and simply map SIFT features onto that.

During the learning phase SIFT features are detected in keyframes and mapped to the surface model using the known 3D pose from the tracker. SIFT features falling outside the object boundary are discarded. Keyframes are indicated by the user via a button press. According to Lowe's findings [30], that SIFT can be reliably detected up to a view point change of about 30°, about 6 keyframes taken from around the object plus 2 for top and bottom are typically sufficient. A larger number of keyframes will result in improved recognition rates for tough cases like large scale occlusions or scale changes, but comes at the cost of increased recognition time per object.

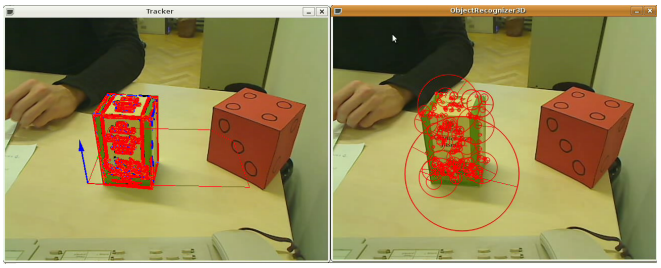

Fig. 6. Learning phase: Edge-based tracking (left) and learned SIFT features (right) during learning phase

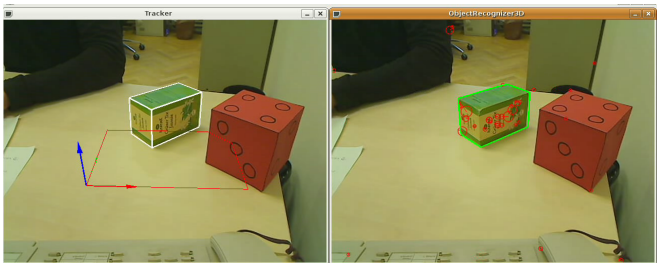

Fig. 7. Recognition phase: Re-initialised tracker (left) after SIFT based recognition (right)

To speed up recognition SIFT features are represented using a codebook (one per object). SIFT descriptors are

clustered using an incremental mean-shift procedure and each 3D location on the object surface is assigned to the according codebook entry.

In the recognition phase SIFT features are detected in the current image and matched with the codebook. According to the codebook entry each matched feature has several corresponding 3D model locations. To robustly estimate the 6D object pose we use the *OpenCV* pose estimation procedure in a RANSAC [31] scheme with a probabilistic termination criterion, where the number of iterations necessary to achieve a desired detection probability is derived from an estimate of the inlier ratio, which is taken to be the inlier ratio of the best hypothesis so far. So the number of RANSAC iterations is adapted to the difficulty of the current situation and accordingly easy cases quickly converge.

To distinguish between hallucinating false detections and correct object locations we define the object confidence

$$p(o \mid m, f_t) = \frac{n_{inlier}}{n_{detected}}$$

of an object o for a given image frame f_t and an object model m as the ratio between the matched interest points n_{inlier} and the number of detected interest points $n_{detected}$ located within the boundary projected to the current image. This provides a good estimate independent of the total number of features in the model and independent of current scale.

Fig. 6 shows an example of the learning phase. On the left we see the tracked objects with overlaid texture edges. The right image shows the detected SIFT features of that view. Fig. 7 shows a recognised object on the right (again with overlaid SIFT features) and the re-initialised tracker on the left.

VI. Experimental Results

We present preliminary results. To evaluate our toolbox we learned 10 object models and recognised them in different lighting conditions and scales. Some objects were simple shapes (box- and cylinder-shaped packaging) that were acquired using the basic shape detector as explained in Section III. Others were more complex (though still of the packaging sort) and for these we downloaded models from Google 3D Warehouse. Furthermore we qualitatively evaluated tracking performance in cases of severe occlusion and scale change.

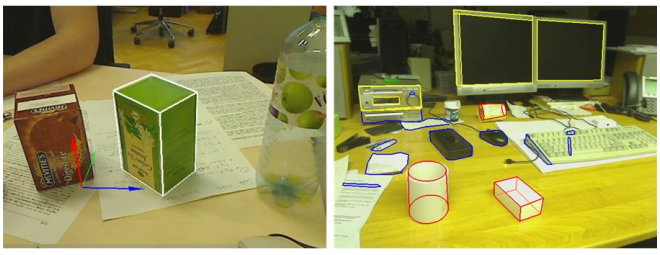

Fig. 8. Detecting a box (left) and the limits of shape detection (right) where red indicates detected shapes and other colours lower level primitives.

Fig. 8 (left) shows a typical case of learning a box-shaped object. Note that shape detection is able to handle moderate amounts of background clutter, no clean white background is required but a typical office table suffices. Also surface texture (with the spurious edges it creates) can be handled as long as the texture does not cross object sides, thus rendering geometry edges all but invisible. Fig. 8 (right) shows the limits of shape detection. The toilet roll lying sideways in the background is detected but results in a bad model and the black box-shaped object is lacking internal geometry edges (only its contour is detected) due to low contrast and thus is not detected as a cuboid.

Fig. 9. Good (left) and bad (right) examples of recognition at 0.5 m and 1.0 m respectively, illustrating how recognition performance for some objects degrades with object distance

Table I shows object recognition rates for two lighting situations (sunny noon and artificial light) and two distances of the camera from the scene (0.5 and 1.0 m) and Fig. 9 shows some typical examples. Recognition rates for each scene were taken over 10 images of that scene. As can be seen some objects like the green "Jasmin" tea box (in the center) with its good texture are very stable, while others like the "JodSalz" (yellow cylinder on the right) suffer badly from scale change. Lighting variations did not matter much, as is to be expected from SIFTs invariance to lighting.

TABLE I

Recognition rates in percent and avg. recognition times per object for different lighting situations and distances, over 10 runs each.

light	noon		artificial	
distance [m]	0.5	1.0	0.5	1.0
Cappy	90	30	100	0
GuteLaune	100	0	100	10
HappyDay	100	70	100	90
Jasmin	100	100	100	100
JodSalz	100	0	100	40
Koala	100	100	100	100
peanuts	100	100	100	80
Digital	100	100	100	100
Visual	90	20	100	100
avg. time [s]	1.023	2.772	0.829	2.403

Note that even in cases where the recogniser's object pose is slightly mis-aligned the tracker, once initialised, typically "snaps" on to the correct object pose quickly.

Fig. 10 shows qualitative results of tracking robustness. In Fig. 10 (top) an object is occluded by the humans hand while being moved and a stable track can be maintained. In Fig. 10 (center) one object is moved in front of the other leading to

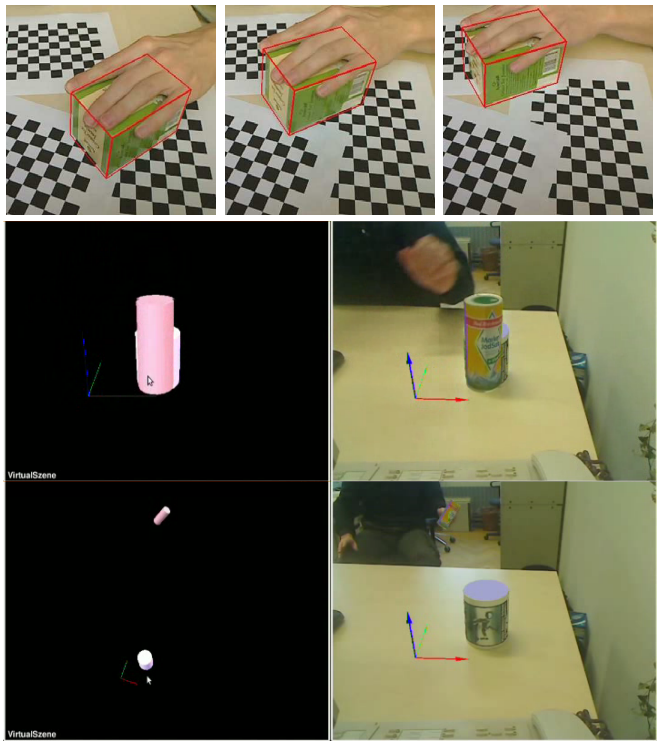

Fig. 10. Robustness of tracking against large scale occlusion (top and center) and large scale variation as well as background clutter (bottom).

even more severe occlusion of the latter. The tracker can still hold on to the occluded object, though in this case it is not moving. Generally accuracy and the maximum trackable speed drop with increasing occlusion. Fig. 10 (bottom) shows robustness with respect to large scale change.

TABLE II

FRAME RATE WITH RESPECT TO NUMBER OF POLYGONS OF THE GEOMETRICAL MODEL WITH DIFFERENT NUMBER OF RECURSIONS AND PARTICLES, COMPUTED ON A GEFORCE 285 GTX

Example Objects	Faces	Frames per Second		
		2x50	3x100	4x300
Box	6	240	100	33
Cylinder (low)	24	220	95	30
Cylinder (mid)	96	210	90	28
Cylinder (high)	384	190	80	25
Complex Scene	1536	160	60	18

Table II shows frame rates of tracking for different models and different number of particles per model. Tracking rate does not depend too strongly on the number of faces in the triangle mesh of the model and is directly related to the number of particles. Note that tracking accuracy and maximum trackable speed increase with number of particles. The number of particles can be changed dynamically for each object while tracking to e.g. maintain a given frame rate or desired accuracy.

Fig. 1 shows a scene with 9 tracked objects, which are all static. So a few particles per object suffice. As soon as an object moves, a higher number of particles will be required to maintain a stable and accurate track. Also other attentional cues (such as a human hand approaching an object that is likely to be moving soon) could be used to intelligently schedule the pool of particles amongst the objects. This is left for future research.

VII. CONCLUSION AND FUTURE WORK

We have presented a toolbox for learning, recognising and tracking objects aimed at robotics research, concentrating on classes of objects that are typical of grasping and fetch-and-carry tasks, namely containers and packaging of various sorts. The toolbox can handle textured as well as non-textured objects and provides shape and full 6D pose in real-time. The simplifying "blocks world" assumption was made for the case of simple shapes (cuboids and cylinders) which can be detected and tracked fully automatically. More complex shapes are handled given that a 3D wireframe model is initially available. While we do not claim to have a silver bullet to cover all problems and scenarios we believe that our simplifying assumptions pose no major limitations for a large class of interesting robotics tasks, such as fetch-and-carry, tutor-driven learning of objects or interacting with a scene composed of unknown simple objects.

Results presented are preliminary and a more detailed evaluation regarding model accuracy and tracking accuracy is needed. Also a more thorough and structured evaluation of recognition performance with regard to scene and model complexity. Several improvements are planned for the individual components. The ground plane assumption required for creating 3D wireframe models from 2D edge images can be removed by simply employing line-based stereo (in cases where the robot provides stereo cameras). The inlier ratio for the RANSAC step of the SIFT based recogniser can be improved taking into account visibility and discarding SIFTs with a surface normal pointing away from the camera. Tracking already provides a fully dynamic allocation of particles for each object and an intelligent scheduling strategy for making best use of the pool of available particles, possibly including additional attentional mechanisms is a promising topic. Regarding learning, we will replace user intervention by key press with an automatic procedure to determine good keyframes for learning SIFTs automatically.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Communitys Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

REFERENCES

[1] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with cast," *Advanced Engineering Infomatics*, vol. 24, no. 1, pp. 27–39, 2010.

[2] S. Winkelbach, S. Molkenstruck, and F. M. Wahl, "Low-Cost Laser Range Scanner and Fast Surface Registration Approach," in *Proceedings of the DAGM*, ser. LNCS, vol. 4174, 2006, pp. 718–728.

[3] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D Model Acquisition," in *Proc. SIGGRAPH*, 2002.

[4] M. Brown, T. Drummond, and R. Cipolla, "3D Model Acquisition by Tracking 2D Wireframes," in *Proc. British Machine Vision Conference (BMVC)*, 2000.

[5] L. Vacchetti, V. Lepetit, and P. Fua, "Stable Real-Time 3D Tracking using Online and Offline Information," *PAMI*, 2004.

[6] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua, "Feature Harvesting for Tracking-by-Detection," in *European Conference on Computer Vision*, vol. 3953, 2006, pp. 592–605.

[7] M. Özuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[8] M. Grabner, H. Grabner, and H. Bischof, "Learning Features for Tracking / Tracking via Discriminative Online Learning of Local Features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[9] P. M. Roth, M. Donoser, and H. Bischof, "On-line Learning of Unknown Hand Held Objects via Tracking," in *Proc. 2nd International Cognitive Vision Workshop (ICVS)*, Graz, Austria, 2006.

[10] H. Riemenschneider, M. Donoser, and H. Bischof, "Robust Online Object Learning and Recognition by MSER Tracking," in *Proc. 13th Computer Vision Winter Workshop (CVWW)*, 2007.

[11] P. Qi, G. Reitmayr, and T. Drummond, "ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition," in *Proc. British Machine Vision Conference (BMVC)*, 2009.

[12] I. Gordon and D. G. Lowe, "What and where: 3D object recognition with accurate pose," in *Toward Category-Level Object Recognition*, J. Ponce, M. Hebert, Schmid. C., and A. Zisserman, Eds. Springer, 2006, ch. What and where: 3D object recognition with accurate pose, pp. 67–82.

[13] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3534–3541.

[14] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundationas and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005.

[15] L. Masson, M. Dhome, and F. Jurie, "Robust Real Time Tracking of 3D Objects," in *Proc. of the 17th International Conference on Pattern Recognition (ICPR)*, vol. 4, 2004.

[16] E. Rosten and T. Drummond, "Fusing Points and Lines for High Performance Tracking." in *IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1508–1511.

[17] G. Klein and D. Murray, "Full-3D Edge Tacking with a Particle Filter," in *Proc. British Machine Vision Conference (BMVC)*, vol. 3, Sept. 2006, pp. 1119–1128.

[18] E. Murphy-Chutorian and M. Trivedi, "Particle Filtering with Rendered Models: A Two Pass Approach to Multi-Object 3D Tracking with the GPU," in *Computer Vision on GPU Workshop (CVGPU)*, 2008, pp. 1–8.

[19] R. C. Nelson and A. Selinger, "A cubist approach to object recognition," Dept. of Computer Science, Univ. of Rochester, Tech. Rep. TR689, 1998. [Online]. Available: citeseer.nj.nec.com/article/nelson98cubist.html

[20] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 36–51, 2008.

[21] B. Ommer and J. Malik, "Multi-Scale Object Detection by Clustering Lines," in *International Conference on Computer Vision*, 2009.

[22] S. J. Dickinson and D. Metaxas, "Integrating Qualitative and Quantitative Object Representations in the Recovery and Tracking of 3-D Shape," in *Computational and Psychophysical Mechanisms of Visual Coding*, L. Harris and M. Jenkin, Eds. Cambridge University Press, New York, 1997, pp. 221–248.

[23] P. Sala and S. Dickinson, "Model-Based Perceptual Grouping and Shape Abstraction," in *The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV 2008)*, 2008.

[24] M. Zillich and M. Vincze, "Anytimeness Avoids Parameters in Detecting Closed Convex Polygons," in *The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV 2008)*, 2008.

[25] A. Richtsfeld and M. Vincze, "Basic Object Shape Detection and Tracking using Perceptual Organization," in *International Conference on Advanced Robotics (ICAR)*, June 2009, pp. 1–6.

[26] T. Mörwald, M. Zillich, and M. Vincze, "Edge Tracking of Textured Objects with a Recursive Particle Filter," in *19th International Conference on Computer Graphics and Vision (Graphicon), Moscow*, 2009, pp. 96–103.

[27] A. Richtsfeld, T. Mörwald, M. Zillich, and M. Vincze, "Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context," in *Computer Vision Winder Workshop (CVWW)*, 2010, pp. 91–98.

[28] A. Doucet, N. De Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[29] C. Wu, "http://www.cs.unc.edu/ ccwu/siftgpu/."

[30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[31] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. of the ACM*, vol. 24, pp. 381–395, 1981.