Project Acronym:	GRASP
Project Type:	IP
Project Title:	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise
Contract Number:	215821
Starting Date:	01-03-2008
Ending Date:	28-02-2012

Deliverable Number:	D15
Deliverable Title :	Mapping of sensor model to the ideal observer and segmenting new action types
Type (Internal, Restricted, Public):	PU
Authors	Ch. Papazov, O.Ruepp, D. Burschka, J. Bohg, D. Kragic, T. Asfour
Contributing Partners	TUM, KTH, UniKarl

Contractual Date of Delivery to the EC:	28-02-2010
Actual Date of Delivery to the EC:	28-02-2010

Contents

Chapter 1

Executive Summary

This deliverable encapsulates several contributions to the implementation of a cognitive architecture for a robot systems with cognitive grasping capabilities. The deliverable reflects work done in WP5 in context of monitoring of the environment for changes and mapping of the sensor model to the perceptual model of the ideal observer. The deliverable focuses on segmentation of new actions violating the current expectation.

In the second year we tackled the work on following tasks

- **Task [5.1]** - continuous work on implementation of the surprise event hierarchy based on neuroscientific findings. The work focused on control of attention to predict possible events and view planning strategies based on findings about human behaviors in similar situations. Research results from WP1 are incorporated into the model representation of the robotic system. The sensor model of the sensor used in the system is adapted on the requirements of the abstract task definition to find an optimal strategy suited for a given imaging model. The work will define the optimal next-view strategy and data processing modalities depending on the current context of the task.
- **Task [5.2]** - Evaluation of efficient methods to monitor changes in the environment that will be insensitive to sensor inaccuracies and that compensate eigen-motions/actions of the system in the environment. In collaboration with the WP4, an internal representation of the environment is generated that will define the expectations of the system. This representation goes beyond a geometric representation of the world and will define also contextual and dynamic information about the world.
- **[Task 5.3]** - is a new extension to validate action primitives through combination of the sensor perception with generic actions specification derived by WP2. Surprise events that reach the Implausibility Layer in the hierarchy defined in Task 5.1 need to be further analyzed by the system. In collaboration with WP2 new actions will be segmented out of a continuous stream of actions and used to define new representations in the ontology defined in Task 5.2.

The work in this deliverable relates to the following second year milestones:

- **[Milestone 4]** Analysis of action-specific visuo-spatial processing, vocabulary of human actions/interactions for perception of task relations and affordances.

The work within this deliverable focused on different aspects of knowledge representation, parsing of human actions and exploration of the environment. We continued to fill in the knowledge representation presented in the previous deliverable (Fig. 1.1).

- **Human Grasp Strategies** - together with WP1 (neurosciences) grasp strategies in human subjects are evaluated. In these experiments, the subjects are told to use different hand configurations (between 2 and all 5 fingers) to grasp objects in the scene. The manipulation capabilities of the hand are deteriorated by using finger covers to reduce applicable forces on object surfaces and, therefore, requiring the subject to use strategies for grasping. This work generates a modified set

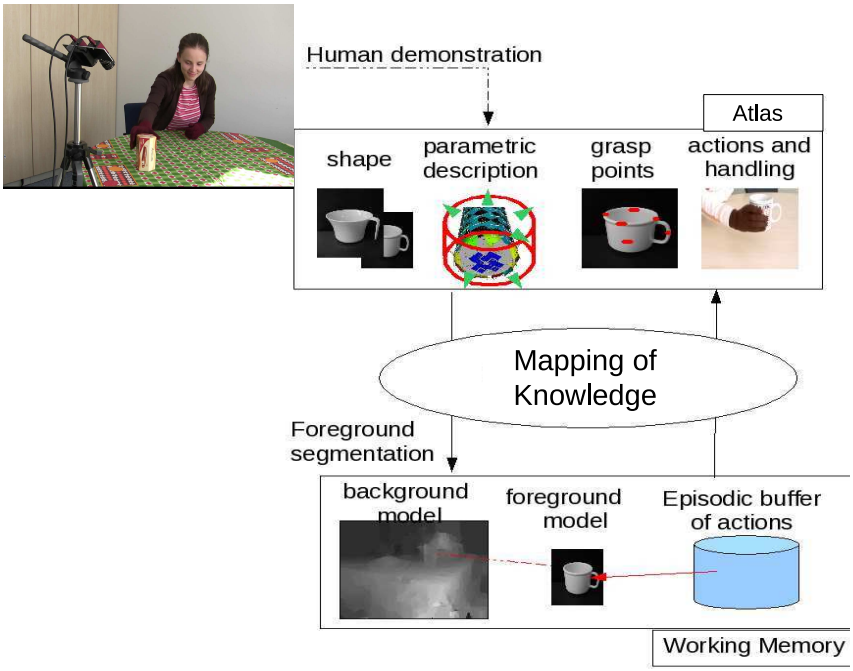


Figure 1.1: Knowledge in cognitive systems is represented a-priori in *Atlas* and it is mapped on the current scene in the *Working Memory*.

of "heat maps" presented already in Year 1 of the project. These heat maps are stored as grasp point candidates in the Atlas in Fig. 1.1. This work will not only provide important input for the haptic exploration step while acquiring new information in the system but will also help in learning about mapping between different manipulator kinematics at a later stage of the project.

- **Scene Representation** - a new representation of the scene geometry is developed where symbolic object representation and a geometric point description are linked together to allow a better maintenance of modifications in the scene due to partial visibility and the resulting partial updates of the geometric pose (Fig. 1.2). The symbolic layer contains also the *Working Memory* description suggested in Year 1 of the project. This work was presented in [RBed, RBon].

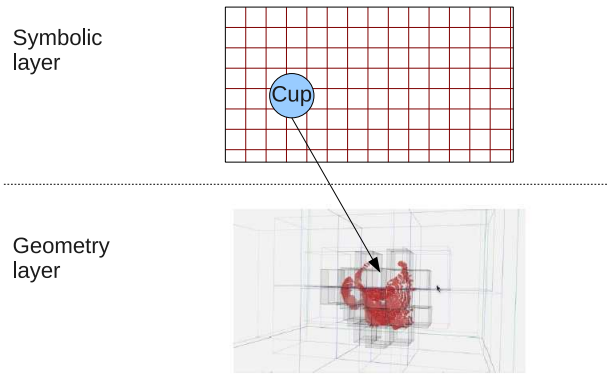


Figure 1.2: Geometric world representation is coupled to a symbolic layer, where 3D points are associated with an abstract object description.

- **Spatio-Temporal Action Analysis** - the *episodic buffer of actions* in Fig. 1.1 needs to be filled with current observations. Each object detected in the scene is described by: its geometrical 3D shape, the actions that were used to manipulate the object, and the locations in space where the object was observed so far. An important task of WP5 is the detection of the surprise events (unexpected occurrences in the world). Such surprise events are new locations for an object in the scene and new type of actions that were applied to an object during manipulation. The *Atlas* contains a collection of all possible actions known to be applicable to an object so far by the system,

but a specific instance of the object in the current scene may allow only a subset of these actions. A good example here is a cup that can be handled in arbitrary way if empty or moved with specific orientation and motion constraints if it contains liquid. An observation of human actions allows to map the appropriate action set from the *Atlas* to the *Working Memory* of the system. This work is presented in [PBar]. We use an object-centric representation in our system, where the object "filters" the space of possible actions and restricts them, e.g., in the rotational space but also the velocities of the handling are limited.

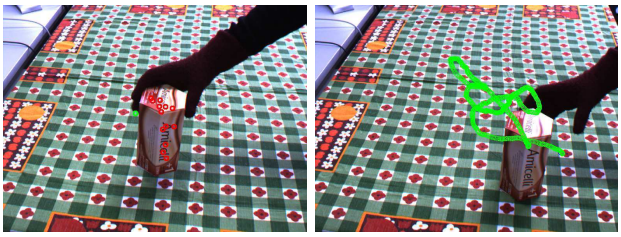


Figure 1.3: Tracking of the object (6DoF from monocular). *Left:* Example of feature set used for tracking. The tracked features are shown in red, the predicted position of the lost features is drawn in green. *Right:* the object trajectory is shown in green.

The contact of the human hand with a geometric structure in the environment is detected and the resulting motion is tracked in 6DoF (Fig. 1.3). In the next step, accurate hand-gesture recognition from WP1 will be added here for a better estimation of the grasp points of the object. This will allow to narrow down the set of possible grasp points stored in the *Atlas* to those used on the current object.

- **Fusion of Visual and Haptic information** - for a better understanding of the physical structure of the foreground objects the shape of the objects does not provide the complete information. We follow two approaches to complete the information about the physical attributes of the objects to be manipulated. Since the observation of the object itself does not allow to estimate parameters as stiffness, friction, mass, and center of gravity, we try to estimate them first from the observation of human actions. This is done in collaboration with WP1 where we try to understand the human strategies in dependence of variations in these physical parameters. A human grasps objects differently depending on the mission goal but also depending on their stiffness and center of gravity. In a second step, the robot tries to repeat the action and here a multi-modal exploration of the object becomes an important tool to complete the information in the *working memory* of the robot. A multi-modal scene exploration allows a validation of initial object hypotheses generated from visual input through haptic exploration with the robot system. A detect-update-predict loop is followed in which the current belief about the state of the map is updated with a Gaussian Process [BJRBKed].

- **Object Registration and Localization in Learning Loop**

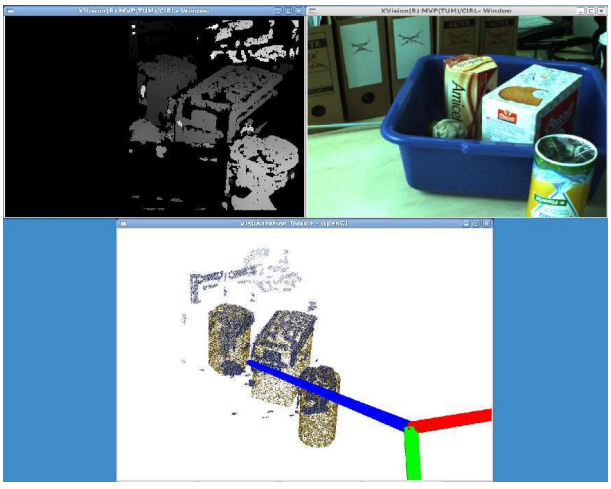


Figure 1.4: Object localization with significant occlusions without any supporting plane pre-selection.

- in continuation of the work from Year 1, we extended the object detection and registration to a system that is able to find objects in arbitrary cluttered scenarios with occlusions by other objects and only limited partial visibility. This approach does not rely on any supporting planes of a table or workbench to cluster points in the scene but it registers objects in arbitrary cluttered scenarios as depicted in Fig. 1.4. In the learning loop, the appearance of the object is unknown and the object is categorized based on its shape or shape components.

  The approach developed here allows also to generalize object categories as planned for the next step in the project. The novel registration process allows to parametrize shape relations of the reference models that are used for registration. This way, different dimensions of, e.g., cups can be registered to one a-priori information in the *Atlas*. This allows to use generic a-priori information not only for known objects but it allows also to generate grasp hypothesis for unknown objects based on their similarities to the objects in the database. At the current stage, we are also able to find similarities between parts of the objects since only a partial view is required to complete the hypothesis of the object. In Fig. 1.4 the gray points are visible parts of the objects that get completed to the yellow model hypothesis.

- **Object Localization in Mission Loop** - once an object is mapped from the *Atlas* to the *Working Memory*, its appearance is known and can be used to localize it in the scene. We can distinguish here between an initial identification of a known object, where the appearance was mapped to the shape categorized in the *Object Registration and Localization in Learning Loop* step. Here, approaches like SIFT can be used to identify an instance of the object in the current working memory [AAD09].

  A real-time localization algorithm to track poses in the scene was developed in [MSSB09] and validated in [SMB+09] for object reconstruction and localization tasks in manipulation area. This system is also used in [PBar] for online analysis of object trajectories to learn about actions in the environment.

- **Object Completion** A novel active stereo system was developed in WP4 with a specific aim on camera-in-hand application, where a second camera was replaced by an active DLP (projector) that projects a calibrated pattern onto the scene that is sensed by the camera mounted in a pre-calibrated location relative to the projector (Fig. 1.5). The projector is light-weight and can be supplied directly from the USB port of the computer processing the stereo information. This is one of the module contributing to the tool-chain that is evaluated by the project to match the goal of an ideal observer for a manipulation task.

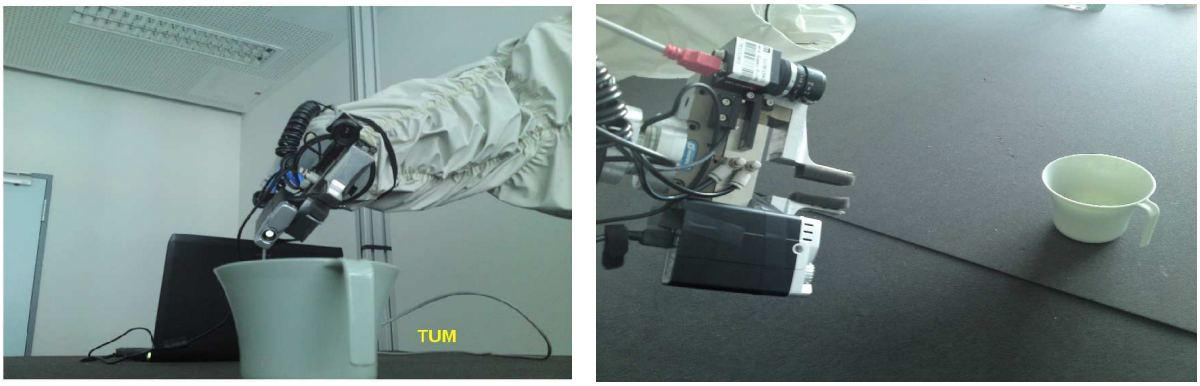


Figure 1.5: One of the cameras of a stereo setup is replaced by an active DLP (digital light processor) to project calibrated texture onto the scene.

Chapter 2

Conclusion

The goal of the deliverable was to identify the requirements of an ideal observer and to provide a set of tools that would allow to satisfy these requirements for manipulation purposes. We identified the needs to satisfy the requirements in different phases of the manipulation task. We evaluated ways how to represent the knowledge about the environment in a way that allows a consistent maintenance of information acquired with actual physical sensors with their limited field of view and accuracy. Since the manipulation task requires not only geometric shape information but also the knowledge about the physical properties of objects in the scene which cannot be observed by a vision-based system alone, we provided two solutions to this problem that can be integrated into a cognitive manipulation system. The first solution is to derive the missing information from observation of the human who interacts with the robot system. Here, neuroscientific findings are integrated into a technical system to extend the knowledge that can be gained about the environment from pure observation. This observation provides many clues about the physical state of the object to be manipulated without the necessity to directly interact with it. The second method involves the robot directly exploring the missing information in an active multi-modal visual and haptic exploration task.

The sensor system of the robot has two functions that were evaluated in this deliverable:

- **Mapping of Knowledge** from the a-priori *Atlas* representation into the *Working Memory* of the robot. Here, sensing modalities needed to be developed for the knowledge representation and registration techniques. The a-priori knowledge is necessary to enrich the visual sensor data by hypotheses derived from previous experience stored in the *Atlas*.
- **Validation of Predictions for Surprise Detection** - we trigger any modifications in the knowledge of our system by unexpected events. The system needs to implement here a surprise detection framework that allows to monitor for events that cannot be explained with the current knowledge stored in the system. In Year 1, we implemented a pure mismatch-based surprise detection monitoring changes in the geometry. This simple approach was extended by a temporal context representing actions in the environment. In the current implementation, the system knows where the mission relevant *foreground* objects are typically localized (where are the places that they usually occur) and it knows the typical handling of the objects. A new action on an object that was not previously observed triggers an update process in the system which modifies the content of the *episodic action buffer* (Fig. 1.1).

Chapter 3

Appendix A: Attached Papers

[AAD09] P. Azad, T. Asfour, and R. Dillmann. Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition. In *Proc. of IROS 2009*, pages 4275–4280, St. Louis, USA, October 2009.

[MSSB09] E. Mair, K. Strobl, M. Suppa, and D. Burschka. Efficient Camera-Based Pose Estimation for Real-Time Applications. *Proc. of IROS 2009*, pages 2696–2703, 2009.

[RBon] J.C. Ramirez and D. Burschka. Data Association for Object-Model and Scene Map Building. In *Proc. of IROS 2010*, Taiwan, in submission.

[RBed] J.C. Ramirez and D. Burschka. Data Abstraction Framework for Object-Consistent 3D Data Fusion. In *Proc. of ICPR 2010*, Istanbul, Turkey, submitted.

[SMB+09] Klaus H. Strobl, Elmar Mair, Tim Bodenmüller, Simon Kielhöfer, Wolfgang Sepp, Michael Suppa, Darius Burschka, and Gerd Hirzinger. The self-referenced dlr 3d-modeler. In *Proc. of IROS 2009, Best Conference Paper Finalist*, pages 21–28, 2009.

[PBar] S. Petsch and D. Burschka. Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans. In *Proc. of IEEE ICRA 2010*, Anchorage, USA, May 2010 (to appear).

[BJRBKed] J. Bohg, M. Johnson-Roberson, M. Björkmann, and D. Kragic. Strategies for Multi-Modal Scene Exploration. In *Proc. of IROS 2010*, Taiwan, submitted.

Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition

Pedram Azad, Tamim Asfour, Rüdiger Dillmann

Institute for Anthropomatics, University of Karlsruhe, Germany

azad@ira.uka.de, *asfour@ira.uka.de*, *dillmann@ira.uka.de*

***Abstract*— In the recent past, the recognition and localization of objects based on local point features has become a widely accepted and utilized method. Among the most popular features are currently the SIFT features, the more recent SURF features, and region-based features such as the MSER. For time-critical application of object recognition and localization systems operating on such features, the SIFT features are too slow (500–600 ms for images of size 640×480 on a 3 GHz CPU). The faster SURF achieve a computation time of 150–240 ms, which is still too slow for active tracking of objects or visual servoing applications. In this paper, we present a combination of the Harris corner detector and the SIFT descriptor, which computes features with a high repeatability and very good matching properties within approx. 20 ms. While just computing the SIFT descriptors for computed Harris interest points would lead to an approach that is not scale-invariant, we will show how scale-invariance can be achieved without a time-consuming scale space analysis. Furthermore, we will present results of successful application of the proposed features within our system for recognition and localization of textured objects. An extensive experimental evaluation proves the practical applicability of our approach.**

I. INTRODUCTION

In the recent past, the recognition and localization of objects based on local point features has become a widely accepted and utilized method. Among the most popular features are currently the SIFT features (Scale Invariant Feature Transform) [1], [2], the more recent SURF features (Speeded Up Robust Features) [3], and region-based features such as the MSER (Maximally Stable Extremal Regions)[4]. The most popular interest point operators are the Harris corner detector [5] and the *Good Features to Track* [6], also referred to as Shi-Tomasi features.

The main task of matching features that are defined by interest points is to achieve invariance to the mentioned changes. In this context, the term feature *descriptor* is often used, denoting the data structure that is compared in order to calculate the similarity between two feature points. Various methods have been proposed for this purpose. In [7], an approach is presented using a rotationally symmetric Gaussian window function to calculate a moment descriptor. In [8], *local jets* according to [9] are used to compute multiscaled differential grayvalue invariants. In [10], two types of affinely invariant regions are proposed: one based on the combination of interest points and edges, and the other based on image intensities. In [3], a speeded up approach named SURF is presented, using a fast Hessian detector and gradient-based descriptor.

In [11], the performance of five types of local descriptors is evaluated: SIFT, steerable filters [12], differential invariants [9], complex filters [13], and moment invariants [14]. In all tests, except for light changes, the SIFT descriptor outperforms the other descriptors.

In [15], an object recognition system with a database of 50 objects is presented, which uses the Gabor wavelet transformation around Shi-Tomasi interest points in order to calculate a feature descriptor. k-means clustering is used to reduce the number of features stored in the database. Murphy-Chutorian and Triesch show empirically that for their test database, 4,000 shared features are the optimal tradeoff between computation time (27 s) and detection rate (79%). Without feature sharing, the storage and comparison of 160,000 independent features would be required.

A completely different approach for point matching is presented in [16]. Instead of calculating a descriptor analytically to achieve invariance, robustness to scaling, rotation, and skew is achieved in a brute-force manner. Each image patch around a point feature is represented by a set of synthetically generated different views of the same patch, intended to cover all possible views. In order to speedup matching, PCA is applied to all view sets. Point matching is performed by calculating the nearest neighbor in the eigenspace for a given image patch. The complete process takes about 200 ms for a single frame on a 2 GHz CPU.

This type of feature representation was used in our previous work [17]. It was shown that through combination with the idea of applying k-means clustering from [15] an object can be recognized within 350 ms, using a database consisting of 20 objects. However, the learning procedure is very time consuming (approx. 20 hours for 20 objects) due to the computation of the covariance matrix for PCA computation and the subsequent k-means clustering. More importantly, such an approach does not allow incremental updates of the database, since the PCA must be computed for all features, as well es k-means clustering.

In this paper, we will present our novel types of features, which combine the Harris corner detector with the SIFT descriptor[1]. In order to achieve scale-invariance in spite of omitting the scale space analysis step of the SIFT features,

[1]Note: The unpublished term *Harris-SIFT* that can be found on the internet has nothing to do with the proposed features and describes a completely different approach.

the features are computed at several predefined spatial scales explicitly. A thorough analysis of the scale coverage of the SIFT descriptor and the proposed extension justifies the choice of the involved parameters. Furthermore, we will present our 2D object recognition system that uses the proposed features. Experimental results show that the proposed features are computed within approx. 20 ms on images of resolution 640×480 and allow robust real-time recognition and localization of a single object at frame rates of 30 Hz using conventional hardware.

The work presented in this paper is part from [18]. In parallel, Wagner et al. have developed a similar approach based on the same idea, using a combination of the SIFT descriptor and Ferns descriptor [19] together with the FAST detector [20], as presented in [21]. In this paper, the original SIFT descriptor is combined with the Harris corner detector, and all parameters are derived from a thorough analysis of the scale coverage of the SIFT descriptor.

II. Feature Calculation

In this section, the developed feature calculation method is presented. As already stated, our experiments proved that the SIFT descriptor is a very robust and reliable representation for the local neighborhood of an image point. However, the scale-space analysis required for the calculation of the SIFT feature point positions is too slow for visual servoing applications. As stated in [3], the computation of the SIFT features for an image of size 800×640 takes approx. 1 s (using a Pentium IV, 3 GHz). This scales to about 0.6 s for the resolution of 640×480. The SURF features require approx. 0.15–0.24 s (depending on the SURF variant) on the same image size. The goal was to find a method that allows feature calculation in approx. 20 ms for an image of size 640×480.

One of the main strengths of the SIFT features are their scale-invariance. This is achieved by analyzing and processing the images at different scales. For this, a combination of Gaussian smoothing and a resize operation is used. Between two so-called octaves, the image size is halved, i.e. resized to half width and half height. The different scales within an octave are produced by applying a Gaussian smoothing operator, and the variance of the Gaussian kernel is chosen in a way that the last scale of one octave and the first scale of the next octave correspond to each other.

Since the scale space analysis performed by the SIFT features for calculating the feature point positions is the by far most time-consuming part, the idea is to replace this step by a faster method, namely an appropriate corner detector. As shown in [22], the Harris corner detector is a suitable starting point for the computation of positions of scale and affine invariant features. In [22], the Harris-Laplace detector, which is based on the Harris corner detector, is extended to the so-called Harris-Affine detector, which achieves affine invariance.

However, the computational effort for the calculation of the Harris-Laplace or even more the Harris-Affine features is again too high for visual servoing applications. Therefore, the goal was to investigate if it is possible to combine the conventional Harris corner detector with the SIFT descriptor, while keeping the property of scale-invariance.

Fig. 1. Image used for evaluation of the scale coverage of the SIFT descriptor. For this image, 284 feature points were calculated by the Harris corner detector, using a quality threshold of 0.01. The computed feature points are marked by the green dots.

As a first step, the scale coverage of the SIFT descriptor computed with a fixed window size of 16×16 was evaluated. For this, the Harris corner points were calculated for the image from Fig. 1 and stored as a set $\{\boldsymbol{x}_i\}$ with $i \in \{1, \ldots, n\}$ and $\boldsymbol{x}_i \in \mathbb{R}^2$. The image was then resized with bilinear interpolation to different scales $s \in [0.5, 2]$. At each scale s, the stored corner point locations were scaled, i.e. $\boldsymbol{x}_i^{(s)} = s\,\boldsymbol{x}_i$, so that ground truth for the correspondences is given by $\boldsymbol{x}_i^{(s)} \sim \boldsymbol{x}_i$. For each feature in the scaled image, the best matching feature in the set $\{\boldsymbol{x}_i\}$ was determined. In Fig. 2, the resulting percentages of correct matches at the different scales are plotted. In order to see the symmetry of the scale coverage, a $\frac{1}{s}$ scale was used for the part of the s-axis left of 1.0.

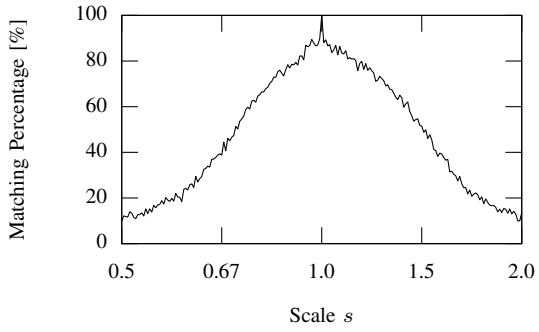


Fig. 2. Plot of the scale coverage of the SIFT descriptor. The evaluation was performed on image scales computed by resizing with bilinear interpolation.

As can be seen in Fig. 2, the matching robustness of the SIFT descriptor is very high (>80%) within a range of approx. 10–15 %. Therefore, it must be possible to close the gap between two scales by exploiting the scale coverage of the SIFT descriptor only, if the scales are close enough to each other. In other words: The idea is that a time-consuming scale space analysis based on a scale space representation using Gaussian filtering can be omitted, and instead a suitable scale factor is used for computing predefined scales using a resize operation with bilinear interpolation. For the

conventional SIFT features, the scale factor between two consecutive octaves is 0.5. The question is now, what is a suitable scale factor Δs with $0.5 < \Delta s < 1$ when omitting the scale-space analysis and closing the gap between adjacent spatial scales by exploiting the scale coverage of the SIFT descriptor only?

In Fig. 3, the matching percentages for the same experiment as before are plotted, this time computing SIFT descriptors at multiple predefined scales. Three scales were used for producing the SIFT descriptors, i.e. $(\Delta s)^0$, $(\Delta s)^1$, and $(\Delta s)^2$. As before, the Harris corner points were only calculated once for the original image, and the image locations were scaled for calculating the SIFT descriptor at the lower scales. Note that this is only done for comparison purposes; for normal application, the interest points are re-calculated at the lower scales to avoid the computation of dispensable features. The peaks at 100 % occur when $(\Delta s)^i = s$, i.e. the features to be matched are computed on the exact same image.

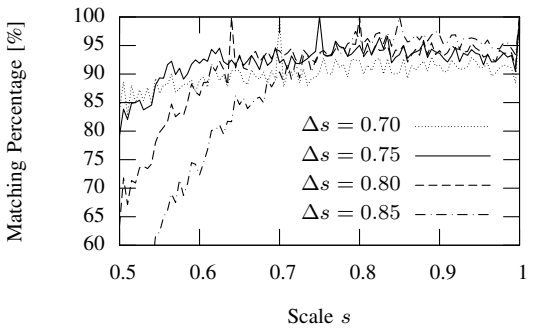


Fig. 3. Plot of the scale coverage when using SIFT descriptors at multiple scales. The evaluation was performed on image scales computed by resizing with bilinear interpolation. Three levels were used; the parameter Δs denotes the scale factor between two consecutive levels.

As can be seen, the scale factors $\Delta s = 0.75$ and $\Delta s = 0.8$ essentially achieve the same performance within the interval $[0.6, 1]$. For the scales smaller than 0.6, $\Delta s = 0.75$ is superior, as expected. Within the interval $[0.8, 1]$, $\Delta s = 0.85$ achieves the best results. However, the performance decreases rapidly for scales smaller than 0.7, since only three levels are used. Within the interval $[0.7, 1]$, $\Delta s = 0.7$ achieves the worst results. The strengths become visible at the smaller scales. However, this can be also achieved by using a larger Δs and an additional fourth level if necessary, while the inferior performance of $\Delta s = 0.7$ for the crucial higher scales cannot be improved. Judging from theses results, $\Delta s = 0.75$ is a good tradeoff between a high matching performance and a high coverage.

Finally, the extended Harris-SIFT features must prove to perform as well when applied in practice, i.e. the training view and the current view are acquired using different setups. The two images used for the following experiment are shown in Fig. 4. The training view on the very right is the same as shown in Fig. 1; it is included again only for illustrating the scale differences. The features were tested on the image

Fig. 4. Images used for testing the performance of the extended Harris-SIFT features. The computed feature points are marked by the white dots. Left: view corresponding to a scale of 0.32 relative to the training view, with 438 computed feature points. Middle: view corresponding to a scale of 0.64 relative to the training view, with 500 computed feature points. Right: training view, with 284 computed feature points.

shown in the middle of Fig. 4, which contains the object at a scale of approx. 0.64. For the tests, this image was resized to scales from $[0.5, 1]$, i.e. the smallest effective scale of the object was $0.5 \cdot 0.64 = 0.32$ (see left image from Fig. 4), compared to the training view.

In Fig. 5, the total number of successfully matched interest points at each scale for this experiment is plotted. Note that according to [1], for each point, several SIFT descriptors are computed, if the calculated orientation tends to be ambiguous. In order to not falsify the results by counting several matches for a single interest point, for each interest point at most one match was counted. By doing this, the resulting plot shows what counts for recognition and pose estimation: the number of successfully matched image locations. The plot shows the results for $\Delta s = 0.75$, using 3, 4, and 5 levels, respectively. The maximum number of interest points was restricted to 500. For the computation of the SIFT descriptor, a fixed window size of 16×16 was used.

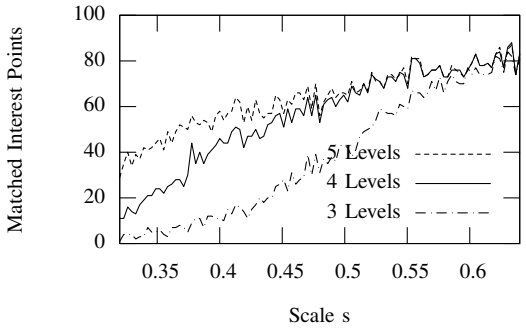


Fig. 5. Illustration of the performance of the extended Harris-SIFT features for the views shown in Fig. 4. As scale factor, $\Delta s = 0.75$ was used. The plot shows the total number of successfully matched interest points at each scale s, where s is understood in relation to the object's size in the training image.

As can be seen, using four or five levels leads to the same results within the interval $[0.47, 1]$. When using three levels, the performance starts to decrease noticeably at approx. 0.57. For the performed experiments, three levels were used for the training views, which proved to be fully sufficient when using images of size 640×480. Note that, in practice, often the limiting factor is the effective resolution of the object in the image, and not the theoretical scale invariance of the features.

III. Recognition and 2D Localization

In this section, our recognition and 2D localization system, in which the proposed features are applied, is summarized briefly. The approach is a variant of Lowe's framework [1]; the main differences are the voting formula for the Hough transform and the final optimization step using a full homography. Details are given in [18].

The feature information used in the following is the position (u, v), the rotation angle φ and the feature vector $\{\boldsymbol{f}_j\}$ consisting of 128 floating point values in the case of the SIFT descriptor. These feature vectors are matched with those of the features stored in the database using a nearest neighbor approach. For recognizing objects on the basis of point feature correspondences, an approach consisting of three steps is used:

A) Hough transform
B) RANSAC
C) Least squares homography estimation

A. Hough Transform

In the first step, a two-dimensional Hough space with the parameters u, v is used; the rotational information φ and the scale s are used within the voting formula. In contrast to [1], the scale s is not taken from the features but votes are cast at several scales [23], since the scale is not computed by the Harris-SIFT features.

Given a feature in the current scene with u, v, φ and a matched feature from the database with u', v', φ', the following bins of the Hough space are incremented:

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = r \left[\begin{pmatrix} u \\ v \end{pmatrix} - s_k \begin{pmatrix} \cos\Delta\varphi & -\sin\Delta\varphi \\ \sin\Delta\varphi & \cos\Delta\varphi \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} \right] \tag{1}$$

where $\Delta\varphi := \varphi - \varphi'$ and s_k denotes a fixed number of discrete scales. According to the results of the extended Harris-SIFT features for $\Delta s = 0.75$ and using three levels (see Fig. 5), $s_k := 0.5 + k \cdot 0.1$ with $k \in \{0, \ldots, 5\}$ was used for the performed experiments. The parameter r is a constant factor denoting the resolution of the Hough space.

After the voting procedure, potential instances of an object in the scene are represented by maxima in the Hough space. The set of correspondences is then filtered by only considering those correspondences that have voted for a maximum or cluster of interest.

B. RANSAC

In the second step, a RANSAC approach is applied using the filtered set of correspondences from the previous step. The RANSAC algorithm allows to filter outliers, which could potentially lead to a wrong local minimum throughout the least squares approach for accurate homography estimation in the third step. For the error tolerance, 5 pixels are used and a fixed number of 200 iterations.

C. Least Squares Homography Estimation

For the filtered set of feature correspondences resulting from the RANSAC algorithm, now a full homography is estimated with a least squares approach. First, in an iterative procedure, an affine transformation is computed, filtering outliers in each iteration. In the final step a full homography is estimated to allow for maximum accuracy.

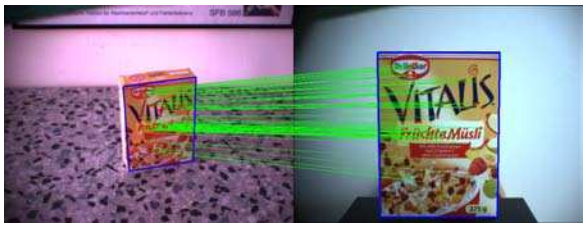


Fig. 6. Filtered feature correspondences after iterative computation of the affine transformation.

If after the complete process of homography estimation, a certain number of feature correspondences is remaining and the mean error is smaller than a predefined threshold, an instance of the object is declared as recognized. The final, filtered set of feature correspondences for an example scene is illustrated in Fig. 6. The 2D localization is given by the transformation of the contour in the training view to the current view.

IV. Run-time Considerations

As described in Section II, throughout the experiments three levels were used with a scale factor of $\Delta s = 0.75$. However, when assuming that the object never appears larger than the largest training view, then multiple levels are not needed for feature computation on the current view. It is sufficient to use multiple levels for the training view, so that the object can be recognized at smaller scales. This strategy significantly reduces the number of feature comparisons and therefore the run-time of the matching procedure.

The computation of the nearest neighbor for the purpose of feature matching is the most time-consuming part of the complete recognition and localization algorithm. To speedup the nearest neighbor computation, a kd-tree is used to partition the search space; one kd-tree is built for each object. In order to perform the search efficiently, the Best Bin First (BBF) strategy [24] is used. This algorithm performs a heuristic search and only visits a fixed number of n_l leaves. The result is either the actual nearest neighbor, or a data point close to it. The parameter n_l depends on the number of data points i.e. SIFT descriptors: The more SIFT descriptors the kd-tree contains, the greater n_l must be to achieve the same reliability. Since each kd-tree only contains the features of one object, n_l can be chosen to be relatively small. Throughout the experiments, $n_l = 75$ was used for feature sets consisting of not more than 1,000 features.

V. Experimental Results

In this section, results of experiments for the evaluation of repeatability, accuracy, and speed are presented. The repeatability of the proposed features equals the repeatability of the Harris corner points within a scale interval of approx. $[0.87,\ 1]$, when using a scale factor of $\Delta s = 0.75$ ($\sqrt{0.75} \approx 0.87$). For measuring the repeatability, the

right image from Fig. 6 was rotated and scaled, and the Harris corner points were computed both on the original and the result image. The repeatability measure was computed with the formula given in [22]. When applying the Harris corner detector, three parameters are important: the quality threshold, the minimal distance between two feature points, and the maximal number of feature points to be calculated. Throughout all experiments, we used a minimal distance of 5 pixels. The quality threshold was set to 0.001 in order to produce many features. Fig. 7 shows the results for 500 and 1200 feature points, where 1200 was the maximum number of features that could be calculated with the chosen parameters. As can be seen, the repeatability at the scale 0.87 amounts to 73% for 500 points and 84% for 1200 points. Note that it is impossible to provide one representative value of the repeatability for a specific scale, since the repeatability always depends on the provided parameters and the image data.

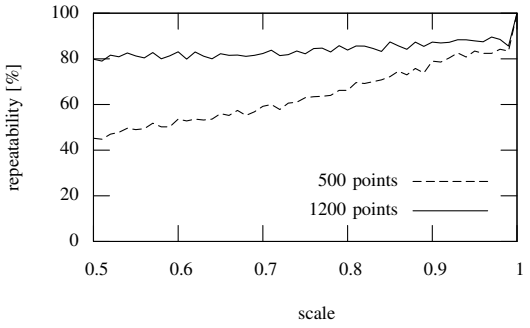


Fig. 7. Results of repeatability experiments.

The performance of the proposed features within our object recognition system and localization system is shown in Fig. 8. A difficult scene with skew and a low effective resolution of the object of interest was chosen. The 2D error was computed as the mean projection error into the current image. As can be seen, a low quality threshold for the Harris corner detector should be used.

The processing times given in Table I were computed using a trained object representation containing 700 SIFT descriptors and 230 SIFT descriptors were extracted from the current view. The processing times for matching and for homography estimation scales linearly with the number of trained objects. Furthermore, the matching time scales linearly with the number of features extracted from the current view. The system was implemented using the Integrating Vision Toolkit (IVT)[2], which, among many other features, offers a fast Harris corner detector (compared to OpenCV 1.0: 10 ms vs. 17 ms) and an efficient kd-tree implementation. The company *keyetech*[3] offers highly optimized implementations (e.g. Harris corner detection within less than 5 ms or nearest neighbor computation).

[2]http://ivt.sourceforge.net
[3]http://www.keyetech.de

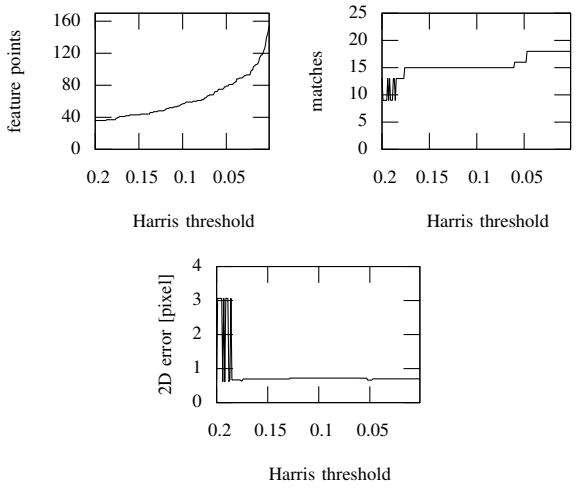


Fig. 8. Effect of the Harris quality threshold for an example with a low resolution of the object. One learned view was used containing 700 feature descriptors. The computation time of the Harris corner points took 13 ms in all cases.

Finally, exemplary recognition results on real image data aquired by the humanoid robot ARMAR-III [25] operating in a kitchen environment are shown in the Fig. 9 and 10. The video attachment shows the results of processing an image sequence with a moving object.

	Time [ms]
Harris corner detection	10
SIFT descriptor computation	6
Matching	12
Iterative homography estimation	3
Total	**31**

TABLE I

PROCESSING TIMES FOR THE PROPOSED OBJECT RECOGNITION AND LOCALIZATION SYSTEM. THE OBJECT REPRESENTATION CONSISTED OF 700 DESCRIPTORS AND THE CURRENT VIEW CONTAINED 230 DESCRIPTORS. THE TESTS WERE PERFORMED ON A 3 GHZ CORE 2 DUO.

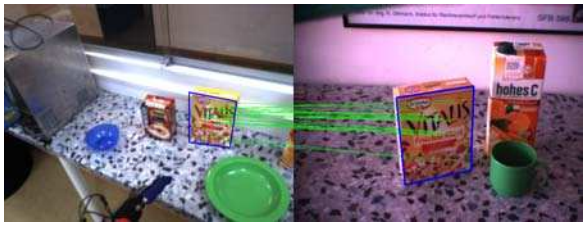


Fig. 9. Computed feature correspondences and recognition result for a difficult scene, featuring out-of-plane rotation and a low effective resolution of the object of interest.

VI. DISCUSSION

We have presented a novel type of point feature, which combines the Harris corner detector with the SIFT descriptor. It was shown how scale-invariance can be achieved efficiently and effectively without a time-consuming scale

Fig. 10. Exemplary results with the proposed object recognition and localization system.

space analysis. Furthermore, the integration of the proposed features in our object recognition and localization system has been presented.

Results from experiments on simulated image data as well as on real image data from the humanoid robot ARMAR-III operating in a kitchen environment proved the practical applicability and performance of the proposed features. The features are computed within approx. 20 ms for an image of resolution 640×480; with the proposed system a single object can be tracked in real-time at frame rates of 30 Hz.

ACKNOWLEDGMENT

The work described in this paper was partially conducted within the EU Cognitive Systems projects PACO-PLUS (IST-FP6-IP-027657) and GRASP (IST-FP7-IP-215821) funded by the European Commission, and the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

REFERENCES

[1] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Greece, 1999, pp. 1150–1517.

[2] ——, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision (ECCV)*, Graz, Austria, 2006, pp. 404–417.

[4] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," in *British Machine Vision Conference (BMVC)*, vol. 1, London, UK, 2002, pp. 384–393.

[5] C. G. Harris and M. J. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151.

[6] J. Shi and C. Tomasi, "Good Features to Track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, USA, 1994, pp. 593–600.

[7] A. Baumberg, "Reliable Feature Matching Across Widely Separated Views," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Hilton Head, USA, 2000, pp. 1774–1781.

[8] C. Schmid and R. Mohr, "Local Grayvalue Invariants for Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 19, no. 5, pp. 530–535, 1997.

[9] J. J. Koenderink and A. J. van Doorn, "Representation of Local Geometry in the Visual System," *Biological Cybernetics*, vol. 55, pp. 367–375, 1987.

[10] T. Tuytelaars and L. V. Gool, "Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions," in *British Machine Vision Conference (BMVC)*, Bristol, UK, 2000.

[11] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Madison, USA, 2003, pp. 257–263.

[12] W. Freeman and E. Adelson, "The Design and Use of Steerable Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 13, no. 9, pp. 891–906, 1991.

[13] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets," in *European Conference on Computer Vision (ECCV)*, Copenhagen, Denmark, 2002, pp. 414–431.

[14] L. V. Gool, T. Moons, and D. Ungureanu, "Affine / Photometric Invariants for Planar Intensity Patterns," in *European Conference on Computer Vision (ECCV)*, vol. 1, Cambridge, UK, 1996, pp. 642–651.

[15] E. Murphy-Chutorian and J. Triesch, "Shared features for Scalable Appearance-based Object Recognition," in *Workshop on Applications of Computer Vision (WACV)*, Breckenridge, USA, 2005, pp. 16–21.

[16] V. Lepetit, J. Pilet, and P. Fua, "Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Washington, DC, USA, 2004, pp. 244–250.

[17] K. Welke, P. Azad, and R. Dillmann, "Fast and Robust Feature-based Recognition of Multiple Objects," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Genova, Italy, 2006, pp. 264–269.

[18] P. Azad, "Visual Perception for Manipulation and Imitation in Humanoid Robots," Ph.D. dissertation, Universität Karlsruhe (TH), Karlsruhe, Germany, 2008. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000011294

[19] M. Özuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," in *European Conference on Computer Vision (ECCV)*, Graz, Austria, 2006, pp. 430–443.

[20] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *European Conference on Computer Vision (ECCV)*, Graz, Austria, 2006, pp. 430–443.

[21] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, Cambridge, UK, 2008, pp. 125–134.

[22] K. Mikolajczyk and C. Schmid, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 1, pp. 63–86, 2004.

[23] P. Azad, T. Asfour, and R. Dillmann, "Stereo-based 6D Object Localization for Grasping with Humanoid Robot Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, USA, 2007, pp. 919–924.

[24] J. S. Beis and D. G. Lowe, "Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Juan, Puerto Rico, 1997, pp. 1000–1006.

[25] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Genova, Italy, 2006, pp. 169–175.

Efficient Camera-Based Pose Estimation for Real-Time Applications

Elmar Mair, Klaus H. Strobl, Michael Suppa and Darius Burschka

***Abstract*— Accurate online localization is crucial for mobile robotics. In this paper, we describe a real-time image-based localization technique, which is based on a single calibrated camera. This can be supported by a second camera to improve accuracy and to provide the proper translational scale. The system aims for a robust and unbiased pose estimation on highly dynamic and resource-limited systems, requiring the following steps: The robustness of the applied pose estimation technique has been significantly improved, a novel approach for stereo subpixel accurate landmark initialization is used and the conventional tracking routines have been sped up to achieve online capability. Although, the algorithm is designed for accurate, online short-range egomotion estimation for hand-held 3D scanning, it can be used for any mobile robot application. Various tests and experimental results with a mobile platform and a hand-held 3D modeler are presented and discussed.**

I. MOTIVATION

Visual localization has become an engaging field in the last years. Especially in mobile robotics, the advantages of optical sensors are evident. The compact, accurate, noninvasive and low-current cameras replace more and more complex laser or sonar sensors and conventional error-prone odometry. Several other advantages arise from using cameras: not only geometric, but also textured maps can be built and the knowledge about the human or the animal vision sense can be applied.

However, due to photometric effects and the loss of one dimension by the optical projection, it is not trivial to extract 3D information from images. The complex 3D reconstruction is time-consuming and crucial on resource-limited computers like in embedded systems. Especially, if the grabbed scene is close to the camera, e.g. as in hand-held scanning, preprocessing becomes more complicated. Tracked landmarks leave rapidly the field of view and stereo correspondences must be found within a large image region.

For highly dynamic mobile systems, it is essential to obtain an accurate pose in real-time. Processing only the video stream of one camera keeps the computational effort small enough to allow online processing. There are two possibilities to retrieve 3D information from a monocular image stream: artificial markers or structure from motion (SFM) approach. Often it is impracticable to modify the environment and, on the other hand, SFM lacks a true scaling factor. Using a second camera would provide precise 3D landmark estimation by stereo triangulation. However, stereo matching is time-consuming and hence not online capable on resource-limited systems. A combination of monocular image processing where the tracked features are initialized by a second camera yields the solution for both requirements: a precise pose estimation in real-time.

In this paper, we present an accurate real-time localization system based on a single, calibrated camera. By combining this monocular localization with a novel stereo-initialization step, the speed of single camera processing and the accuracy of subpixel stereo triangulation can be achieved. No external referencing system is necessary, nor artificial markers are used. Further, the robustness of the pose estimation by the tracked features, based on visual GPS (VGPS), has been improved [1], [2]. An intelligent feature management robustifies the pose estimation additionally. The algorithm is designed for close range applications like hand-held 3D scanning, but allows arbitrary, highly dynamic mobile robots to estimate their motion online without any knowledge about their environment.

Fig. 1 shows two application scenarios where the presented system is used and which are explained in more detail in section IV-D and IV-E. In the left picture, the 3D modeling process is illustrated: the precise poses of the sensors have to be determined in order to allow a robust fusion of the acquired data. The right image shows a Pioneer 3-DX moving around a table. The acquired camera frames are used to build an image-based environment model, requiring an accurate pose estimation to merge the images properly.

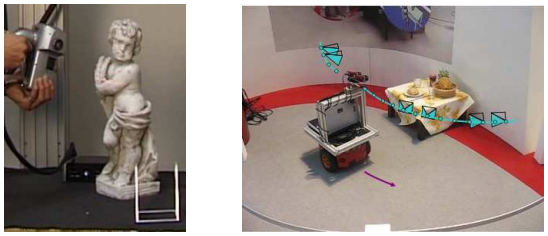


Fig. 1. *Left:* A hand-held 3D modeling system has to be localized globally in order to fuse the acquired data. *Right:* A Pioneer 3-DX watching a scene from different views to build a vision-based environment model.

The remainder of this paper is structured as follows. In the next section, we are setting our work in context to the related work. The algorithm is divided into several modules, which are described in Section III. In IV we show some experimental results for the modules and for the whole algorithm. Furthermore, the two mentioned application scenarios

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems (CoTeSys)* and within the German Aerospace Center (DLR).

E. Mair and D. Burschka are with the Department of Informatics, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany {elmar.mair,burschka}@cs.tum.edu

K. H. Strobl and M. Suppa are with the Institute for Robotics and Mechatronics, German Aerospace Center (DLR), Münchner Str. 20, 82234 Wessling, Germany {klaus.strobl,michael.suppa}@dlr.de

are discussed in detail.

II. RELATED WORK

In order to extract some exact 3D information from image-sequences, the camera positions have to be known. Several different approaches in this field already exist, which are based on the optical flow between two images. The most popular ones to estimate the sensor position by a calibrated camera are probably the 8- ([3]) and 5-point ([4]) algorithms. Nevertheless, several iterative methods as the 3-point algorithms [4] or vision-based GPS (VGPS) [1], [2] are well known, too.

The visual SLAM (V-SLAM) algorithms as localization technique are also well known in literature. The solution provided by Davison consists of building a probabilistic 3D map with a sparse set of good landmarks to track [5], [6]. The points are used in an Extended Kalman Filter (EKF) for a repeatable localization with limited drift. However, due to the map, this is only real-time capable in restricted environments - e.g. the amount of landmarks in [6] has been limited to 100. A well known problem of monocular localization is the correlation of localization error and feature initialization error. If no loop-closure can be accomplished (e.g. due to a straight trajectory), the estimated error is accumulated and the accuracy decreases with increasing distance from the starting point. An inexact localization leads to a false landmark initialization, which again results in an inaccurate pose estimation.

By efficiently separating the tracking and the mapping routines, Klein was able to achieve even more accurate results as the EKF based approach of Davison [7]. Further, the method scales better and even if it is designed for small workspaces, it is also applicable for larger environments. The drawbacks are the high processing power and the memory usage for the keyframes and the large number of features. Hence, up to now this method is not usable on resource restricted systems.

Thanh et al. describe a stereo SLAM method with two EKFs [8]. They use a combination of mono- and binocular feature estimation. At the beginning the stereo-EKF is initialized by the simple odometry and then the stereo-matching is done with help from SLAM. Thus, the features are initialized and used for monocular pose estimation. This rather complex stereo algorithm improves the accuracy and robustness of conventional MonoSLAM clearly, especially for long range motion. However, stereo processing is time-consuming and hence the maximum possible framerate is about only 9 Hz.

Mourikis and Roumeliotis present a dual-layer localization architecture [9]. A combination of a Multi-State-Constraint Kalman Filter (MSC-KF) and a Bundle Adjustment yields highly accurate long-term visual pose estimation. In their work they achieved a processing time of the MSC-KF of about 100 ms. However, the combination with an inertial measurement unit (IMU) leads to a robust long-range localization.

Several other V-SLAM approaches in literature are combined with probabilistic techniques, like e.g. Monte-Carlo localization [10] or particle filters [11]. The application area of such methods is a map based environment, where the map is known in advance. This does not adhere to some of the constraints our algorithm is based on, namely those for 3D modeling without any modifications in the environment.

III. REAL-TIME VISION-BASED LOCALIZATION

In order to provide input for a navigation control-loop of a mobile robot a high data rate is favored. The higher the data rate, the more dynamic can be the controlled system. The localization algorithm should, therefore, not limit the camera framerate. In the following, we call an image-based algorithm real-time or online-capable if its processing time permits the standard camera framerate of 25 Hz. Regarding 3D modeling systems, not only the *speed* but especially the *accuracy* of the estimated pose is crucial. Only so, the acquired data can be reliably merged to a global model. Thus, it appears that the most challenging part is to bring together these two conflicting objectives.

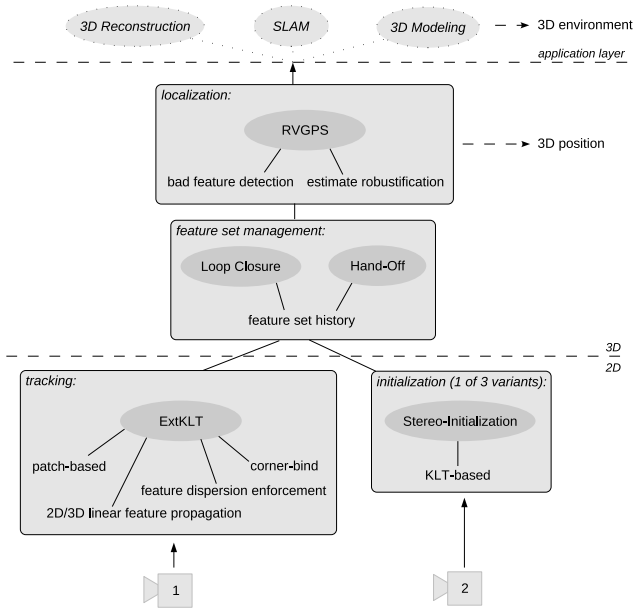


Fig. 2. Several modules are necessary for localization from images. The resulting positions can be used for an arbitrary application as 3D Modeling or SLAM. The 3D structure can be initialized in different ways - in this graphic only the stereo approach is listed.

The complexity of visual localization becomes first apparent in the cumbersome image preprocessing. Landmarks have to be extracted, which can then be tracked reliably over time. These image features have to be initialized and managed for an intelligent hand-off and loop closure. Based on the feature locations, the camera pose is estimated. Figure 2 illustrates the modules of the framework.

Dealing with high dynamic systems, like a human's wrist or a flying robot, makes any time-based filtering (e.g. a Kalman filter) dangerous to wrongly damp the measurements. Renouncing such time-based filters prevents any restrictions in dynamics but demands at the same time

measurements with only a small jitter. We do not provide probabilistic methods to smooth the motion estimation over time, but try to make the algorithm as reliable and robust as possible. Depending on the application, the results can nevertheless be used as input for any probabilistic framework, as e.g. Kalman or particle filters.

A. Monocular, extended KLT feature tracking

In order to estimate the motion from an image stream, the changes between the images have to be detected. Without any artificial markers and with no knowledge about the environment, this task is known to be not trivial [3]. First, good features to track must be selected and then these features have to be tracked from image to image.

Due to its speed and its robustness, the Kanade-Lucas-Tomasi (KLT) tracker ([12], [13]) fits our requirements best. Nevertheless, some improvements to the standard implementation of the KLT algorithm were necessary to provide a 25 Hz image processing in resource-limited environments:

- The KLT preprocessing step consists of smoothing the image and calculating its gradients. This means that three convolutions with different kernels need to be done. To speed up tracking, the preprocessing has been restricted to small patches around the tracked points. Hence, not the whole images have to be processed and stored, but just those small image patches. Therefore, also the processing of high resolution images becomes possible.
- A linear motion model allows not only for larger feature displacements between the images, but also for a smaller searching area. The results are fewer tracking iterations and a reduced size of the image-patches. The motion is modeled based on the 2D feature displacements in the image and provides so a strict separation of the tracking and the pose estimation routines which increases the robustness.
- It is often the case that a scene captured by a camera is built by different structured regions. Patterns, which are rich in contrast are preferred by local feature trackers like KLT, because they allow a good discrimination during tracking. Therefore, a well known fact for those trackers is that they often take only features within a small region of the image with a high-contrast pattern. Splitting the image into commensurate subimages for feature selection, leads to a better landmark dispersion. A feature set, which covers a wider cone of view allows a better conditioned pose estimation.

B. Subpixel-accurate structure initialization

To ensure real-time capabilities, this visual localization method is based on a monocular video-stream. Because of the loss of one dimension by the perspective projection with no further knowledge it is only possible to estimate the translation up to scale. However, there are three different ways to initialize the scale from images anyway: by using the dimensions of some known objects (*Structure from Reference*), by moving a camera (*Structure from Motion*) or by using a stereo camera system with stereo triangulation (*Structure from Stereo*) [3]. Any of these three mentioned initialization techniques can be used at this point. The stereo variant allows the most precise feature initialization and, therefore, this method is chosen. Indeed our approach leads to a subpixel-accurate result.

Conventional stereo matching by corresponding patches on the epipolar line is error-prone and results in pixel-accurate stereo matches. In practice the epipolar line even grows to a small band due to calibration inaccuracies. Therefore, often feature matching routines like SURF [14] and SIFT [15] are used. These use the computational expensive Harris-Affine and Difference-of-Gaussian point detectors, to be able to deal also with affine transformations. However, in the case, where the cameras are mounted parallel on the stereo rig and the baseline is short, the affine component of the stereo-transformation can be neglected. This leads to the same assumptions of Shi and Tomasi in [13]. According to that, good features correspond to a matrix $\boldsymbol{Z}$ with large eigenvalues, whereas $\boldsymbol{Z}$ is defined as following integral over the window W

$$\begin{aligned} \boldsymbol{Z} &= \int\int_W \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}^T(\boldsymbol{x})w(\boldsymbol{x})d\boldsymbol{x}\,, \\ where \quad \boldsymbol{g} &= \left[\frac{\partial}{\partial x}\left(\frac{I+J}{2}\right) \quad \frac{\partial}{\partial y}\left(\frac{I+J}{2}\right)\right]^T. \end{aligned} \tag{1}$$

Further, $\boldsymbol{x} = [x, y]^T$ are the feature coordinates, I and J the two images and $w(\boldsymbol{x})$ an optional weightening function, e.g. for smoothing.

Thus, such good features to track are extracted from the main camera image. Next, even a higher number of features is acquired from the second image. This restricts the correspondence search from all pixels within the epipolar band to a few interesting points in it. These are now used as starting points for the KLT tracker, which aligns to subpixel accuracy. If the tracker finds more than one match, we use the one with the smallest pixel difference of the gradient patches. Thus, in case of a match, a subpixel-accurate feature correspondence is found.

Because the stereo matching takes up to one second, depending on the amount of features, it is not real-time capable. Hence, the initialization of a new feature structure is done concurrently while the old feature set is continuously tracked.

Restricting the search range for stereo matches to the displacement corresponding to an object's distance, only features in that space are found. With this restriction, we can localize in respect to an object even if it is moved, because we do not refer to landmarks out of the specified range, like e.g. on the scene background. This allows tracking of a moving object and in the 3D modeling scenario the reconstruction of dynamic subjects.

With SURF or SIFT the accuracy is at best one pixel by the nature of the algorithm and they are in addition computationally more expensive. Some results of our technique are presented in section IV-B.

C. Sequential robustified VGPS - RVGPS

Independent of the tracker, its result can contain bad features due to occlusions, disocclusions, virtual features and reflections. Also the stereo initialization can provide false matches yielding an incorrect depth of the respective feature. Therefore, a localization algorithm has to be used, which is able to detect bad features and reject them from processing and future tracking.

VGPS (vision-based GPS) is an image-based method for the self-estimation of camera poses [1]. The method assumes a reference image S_0 and thereby solves the relative orientation problem of determining the pose of the camera S_t with respect to S_0. Therefore, it requires an internal 3D model (a set of n points ${}^0P_i,\ i \in \{1..n\}$, in the scene) attached to S_0. This model can be constructed in an arbitrary way (e.g. stereo triangulation, see III-B). Thus, the exterior orientation between the current frame and the reference 3D model is computed as follows: an additional tentative 3D model ${}^t\widehat{P}_i$ is generated from the 2D projections in the images by using approximated ranges only. These ranges are estimated from the preceding pose estimation in $t-1$. Thus, the problem of determining the absolute orientation is reduced to finding the relative orientation between these two sets of points ${}^0P_i$ and ${}^t\widehat{P}_i$. This can be solved in closed form using the singular value decomposition (SVD).

In a nutshell: Relative translation and rotation are separately estimated. We first set the origins of the sets of points to their respective centroids without modifying their orientations which yields the sets ${}^0P'_i$ and ${}^t\widehat{P}'_i$. The relative rotation between these sets of points of the same model corresponds to the relative rotation between camera reference frames and can be calculated by maximizing the trace of the inertia matrix of the matched set:

$$ {}^t\boldsymbol{R}^* = \arg\max_{\boldsymbol{R}} \text{trace}({}^t\boldsymbol{R}^T\, {}^t\boldsymbol{M}) \ , \ {}^t\boldsymbol{M} = \sum_{i=1}^{n} {}^t\widehat{P}'_i\ {}^0P_i'^T. \quad (2) $$

Let $(\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V})$ be the SVD of ${}^t\boldsymbol{M}$, that is $\boldsymbol{U}^T\boldsymbol{\sigma}\boldsymbol{V} = {}^t\boldsymbol{M}$, then the solution to Eq. (2) is

$$ {}^t\boldsymbol{R}^* = \boldsymbol{V}\boldsymbol{U}^T \quad (3) $$

and the translation can be also found as follows:

$$ {}^tT^* = \frac{1}{n}\sum_{i=1}^{n} {}^t\widehat{P}_i - {}^t\boldsymbol{R}^* \frac{1}{n}\sum_{i=1}^{n} {}^0P_i \ . \quad (4) $$

Since the tentative 3D model ${}^t\widehat{P}_i$ may differ from the reference one, the final solution is found iteratively by optimizing the quality (the unknown ranges) of the tentative model at the same time. The algorithm terminates whenever sufficient consistency with the original set of points is achieved.

The specifics of our particular VGPS approach are as follows:

- VGPS is here sequentially applied to different reference frames. This is determined by a higher level decision making process (see section III-D).
- The accurate internal model ${}^0P_i$ used by VGPS is in our implementation obtained by stereo-vision at the correspondent reference frame S_0 (see section III-B). Due to the high accuracy of this initialization the acquired model is not updated anymore. Nevertheless, in the next step weights are applied to each feature to rate their quality and to detect outliers.
- The novel solution to the absolute orientation problem within the VGPS algorithm makes use of a redescending M-estimator on the residual Euclidean distances between matched points. This is in order to disregard gross outliers without compromising the estimation convergence because of the naturally noisy nature of the problem. Gross outliers may correspond to either a faulty internal 3D model, false matching correspondences, virtual features (e.g. features from occlusions), or, more infrequently, to wrong tentative ranges and are fatal to unbiased pose estimation. Therefore, in case of outliers, the robustified VGPS not only disregards this data but also sends a signal to the features database in order to remove those features for good. In particular, we use the biweight function of Tukey because of its continuous derivatives and its handy weights. Both, the initial estimation and the chosen scale, are much more influential parameters to global fast convergence than the nature of the employed function [16]. Thus, the modification concerns weighting the contribution of each point to the inertia matrix of the matched set of points with the weight

$$ \begin{array}{lcll} {}^tw_i & \propto & (1 - {}^tS_i \cdot {}^tS_i)^2 & \text{if } |{}^tS_i| < 1 \\ {}^tw_i & = & 0 & \text{if } |{}^tS_i| \geq 1 \end{array} \quad (5) $$

  where ${}^tS_i = ({}^t\boldsymbol{R}\,{}^0P_i - {}^t\widehat{P}_i)/s$ is the estimated normalized matching residual for object point i at instant t before performing the SVD and s is the scale of the inlier noise. In the end:

$$ {}^t\boldsymbol{R}^* = \arg\max_{\boldsymbol{R}} \text{trace}({}^t\boldsymbol{R}^T\, {}^t\boldsymbol{M}^R) \ , \ {}^t\boldsymbol{M}^R = \sum_{i=1}^{n} {}^tw_i\, {}^t\widehat{P}'_i\ {}^0P_i'^T. \quad (6) $$

- Finally, we use an efficient termination policy determined by a threshold on the absolute orientation correction over the course of the iterations.

The advantage by using M-estimaters and not a simple outlier rejection method like RANSAC is that each measurement can be weighted according to its accuracy contribution. Some test results in section IV-C depict the improvements by using M-estimators and compare the RVGPS to the conventional VGPS algorithm.

D. Feature set management

Whenever the camera is moved, the landmarks can leave its field of view. If not enough features are trackable, we concurrently initialize a new feature set and save its offset to the origin. The motion during the feature initialization is estimated based on the old features and is used to propagate the position of the new ones in the current image. Thus, a wrong initialization, e.g. because of false correspondences or occlusions, leads to a false propagation and, therefore, to a

feature's loss - a first bad feature detection is provided. The initialized feature sets are stored in a database.

The accuracy of the pose estimation is not only based on the amount of landmarks, but also where they are located. Like any image-based pose estimation algorithm, RVGPS is also ill-conditioned if all features lie within a small area of the image and especially if they are all at the same image border. To detect such situations, we calculate the centroid of the 3D features and its projection on the image plane. If it leaves a specified inner area of the image a new feature set initialization is triggered, irrespective of the number of features still trackable.

Further, the image-plane projection of all centroids in the database is used to detect the most central feature set and, thus, to trigger a switch-over to the most central one. The accumulated error becomes reduced to the value after initializing that old set. Thus, the feature-set hand-off and short range loop closure policy reads as: A new set is acquired each time the number of trackable features drops below a specific threshold or the projection of the features' centroid is outside the image center. Further, if an old feature set seems to allow a better pose estimation, we attempt to use that one for tracking.

Figure 3 illustrates the different steps and branches within the presented framework.

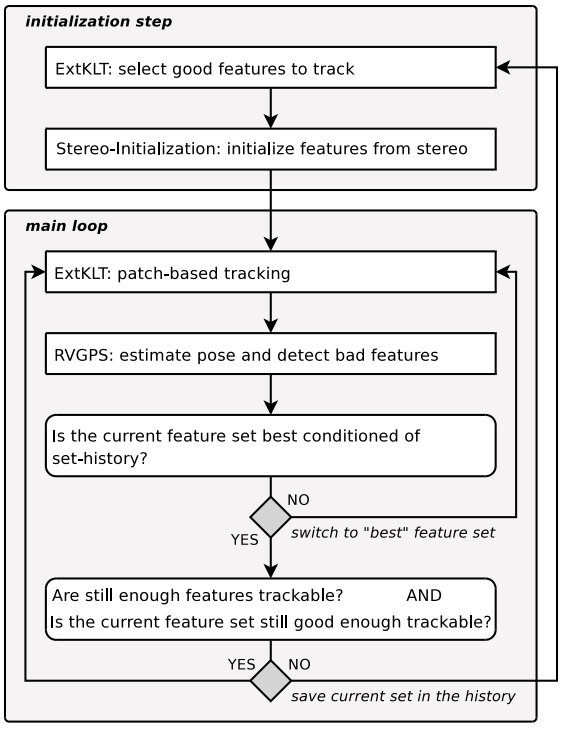


Fig. 3. Simplified flow chart of the image-based localization routine.

IV. EXPERIMENTS

In this section we show some results of experiments regarding the innovations of the presented algorithm. Further, some application results are illustrated.

A. KLT extension

The speed up by the patch based implementation of the KLT tracker carries more weight if less features are tracked within a small search range on a large image. Due to the linear feature propagation this search range can be kept quite small. Fig. 4 shows some experimental results for the standard, the patch-based KLT implementation and a method which uses regions of interest (ROIs) to reduce processing. The latter approach is always less efficient than the whole-image- or patch-based variant and thus it can be neglected. If there are too many features to track, the overlapping areas of the patches produce a larger amount of pixels to be processed and the patch based variant becomes less efficient. The variant preferred depends, therefore, on the feature search range in each direction and the number of features. In our applications we use between 6 and 25 features for tracking, which has proven to be sufficient for reliable and robust pose estimation. Thus, in our case the patch based variant allows a significant speed up.

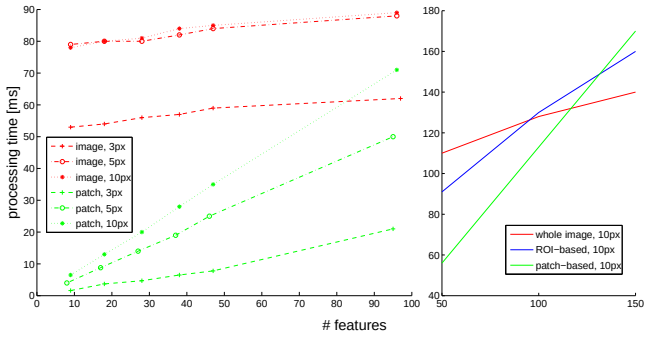
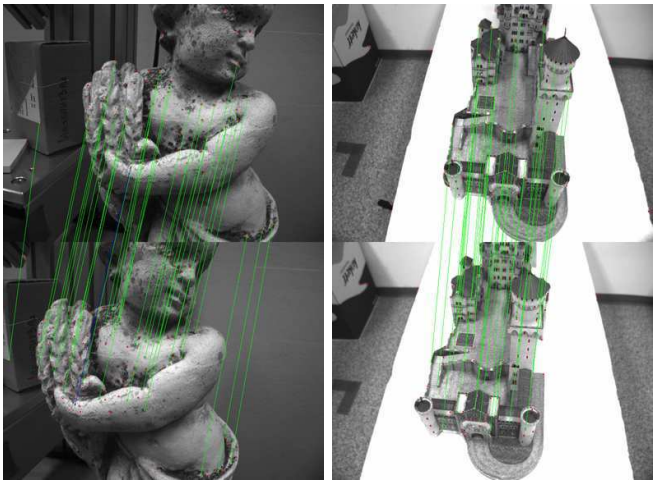


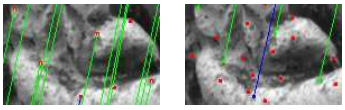
Fig. 4. These figures compare the processing times of the KLT-variants for different number of features to track. The red values are representing the whole-image-based implementation, the blue values the ROI-based version and green are the times for the patch-based variant. On the left, different sizes for the search window are chosen. While the processing time for the whole image almost only depends on the number of pyramid-layers (3px search range results in one, 5px and 10px in two pyramid layers), the patch-based approach is also related to the search range and the number of features. The right image shows the intersection of the lines for a 10px search range.

B. Stereo initialization

The results of the fast and subpixel-precise stereo initialization method (see section III-B) are illustrated in Fig. 5. The lower small images are regions of the left scene and show in detail two corresponding parts of the large pictures. Thus, the reader can see, that the right correspondences are found. The red dots in both shots are the extracted points of interest which are used as matching candidates. The green lines link up corresponding features of the left camera (upper image) and the right one (lower image). The green dots at the end of the lines in the lower image are the sub-pixel accurate correspondences. Blue lines are matches, where the KLT tracker could not find a minimum due to the limited iteration number. Nevertheless, experiments have shown, that these correspondences are mostly also acceptable accurate (as can be also recognized in the small figures).



(a) The KLT based stereo initialization.



(b) Details of the putto-scene above.

Fig. 5. The KLT based stereo initialization allows fast subpixel-accurate stereo matching. The lower small images are parts of the putto-scene and show some details of the upper image (left camera) and the lower image (right camera).

C. Accuracy

Next, we apply the method presented in this work to a stereo camera[1] which is mounted on the wrist of a robotic manipulator *KUKA KR16*. The results of our positioning system are then compared to the accurate output of the robot's kinematics. The experiment consists in a motion round the object to be modeled. The images and the poses of the manipulator are acquired synchronously [18]. There is a total amount of 710 images (see Fig. 6), the round-trip has a total length of $125\,cm$, and the orientation changes arbitrarily up to $55\,°$.[2] We run the experiment twice, whereas the only difference was that the VGPS robustification has been enabled resp. disabled.

Concerning the accuracy analysis: Figs. 8 and 9 show the residual errors in rotation and translation with respect to the data from the *KUKA* manipulator, which relative positioning accuracy is $0.1\,mm$ and in orientation less than $0.1\,°$. The blue dots show the results using RVGPS, while the pink dots represent the output with conventional VGPS.

On the RVGPS performance: The orientation error increases in the course of the experiment up to $0.5\,°$ at the turning point, $65\,cm$ away from the origin. This is because until then the method performs dead-reckoning and accumulates orientation errors after each initialization moment. The accuracy suffices for several modeling applications.

[1]The intrinsic stereo camera calibration and extrinsic positioning with respect to the manipulator was performed using the camera calibration toolbox *DLR CalDe and DLR CalLab* [17].

[2]The video is available at http://www6.in.tum.de/~maire/videos/visualLocalization_castle.mp4

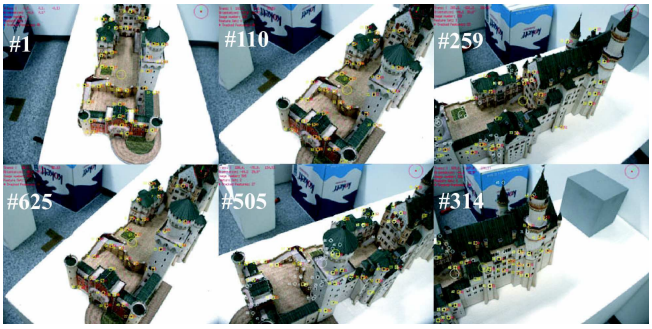


Fig. 6. The motion is from the front (image $\#1$) to the back (image $\#350$) of the castle, and then back to the front in image $\#710$.

After that, on its way back, the field of view heads back to the former structures, being able to find them again and in this way to get rid of the accumulated orientation error. The positioning error is represented in Fig. 9. In the comparison the inaccuracies of our algorithm, due to drifting features and wrong correspondences, add up together with the inaccuracies of the manipulator. At the turning point the mean accuracy surpasses $4\,mm$, but the errors lower after that. In this example the positioning bias is not completely canceled either because of mechanical hysteresis or residual tracking drifts. No Kalman- or particle-filtering is used to prevent any restrictions to the system dynamics.

Comparing both runs: The rotation estimation without using the robustification as described in section III-C is almost the same as with M-estimators. The variation is rather random and would not bear an improvement. However, the difference is visible watching the translation estimations. Here, the pink dots accumulate an error up to more than $20\,mm$. This result can be explained by the characteristics of such an iterative estimation method, like it is used in VGPS: Irrespective of the kind of error, whether it occurred at the feature initialization step or during tracking, the sum of all errors can always be lessened by zooming out the camera. Thus, it is obvious that a gradient descent method follows the gradient on the optical axis of the camera to diminish the error, while disregarding the orientation estimation. Of course, special error combinations could also lead to a wrong rotation estimation. Although, the translational error in the VGPS-run is quite large, the old features are refound, so that also there the error becomes reduced.

D. Application to a hand-held 3D modeler

The *DLR 3D-Modeler* in Fig. IV-D is a multi-sensory compact device for 3D modeling [19]. A strong requirement for modeling is the necessity for accurate pose measurement of the sensor in a fixed common reference frame. Currently, this has been achieved by either (see Fig. 10)

- an external tracking system, e.g. an infrared light-emitting stereo camera rig which tracks the reflecting markers on the modeler platform, or by
- a robotic manipulator, either active or passive.

By using our image-based navigation algorithm, the operation space of the modeler is neither restricted by the robot

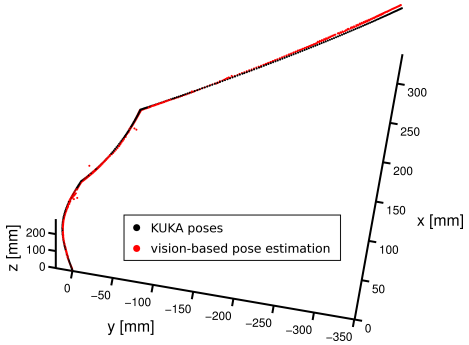


Fig. 7. The two trajectories are compared to each other: black the result of the inverse kinematics of the KUKA robot, red the poses estimated by the presented vision-based algorithm.

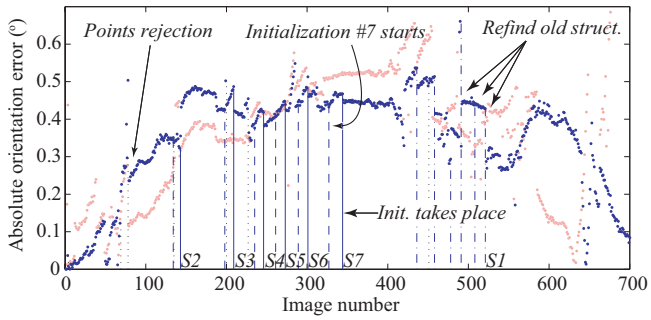


Fig. 8. Residual orientation error with respect to the *KUKA* manipulator. Blue: using RVGPS. Pink: using VGPS.

workspace nor by an external tracking device (see Fig. 11). A further advantage is that the error of the visual navigation algorithm and the laser-stripe profiler correlate, because both methods are based on the same image data.

Fig. 12(a) and 12(b) show modeling results acquired by a two-turn sweep of the DLR 3D-Modeler. Because the device is manually guided, the points are not uniformly distributed on the objects surface.[3] More details to the functionality and usability of this system are depicted in [20]

[3]An example of the operation is available at http://www6.in.tum.de/~maire/ videos/3dMo_putto.mp4

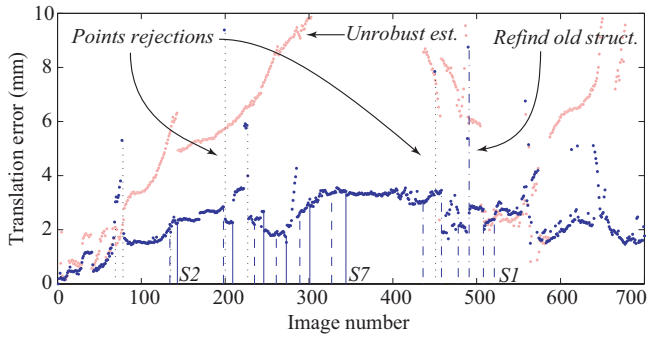


Fig. 9. Residual translation error with respect to the *KUKA* manipulator. Blue: using RVGPS. Pink: using VGPS - the error rises up to more than 20 mm, which is not visible to preserve an adequate scale.

Fig. 10. The upper left picture shows the DLR 3D-Modeler and the PC running the software (an Intel® core duo T2050 with 1.6GHz and 2GB RAM). In the past two external reference systems have been used to estimate the pose of the scanner: a robot arm (lower left), where the joint positions are used to calculate the modeler pose (robot kinematics), and an external tracking system (right), where six spheres attached to the modeler are tracked by stereo cameras.

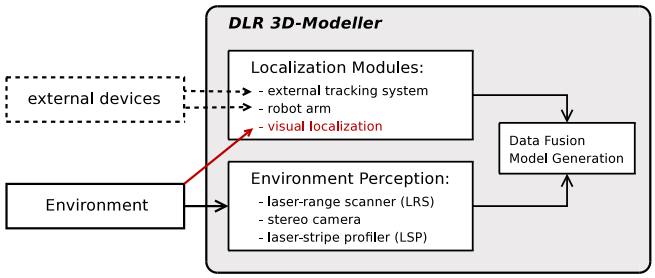


Fig. 11. This figure illustrates the modules provided within the DLR 3D-Modeler. Due to the visual localization module, there is no external device necessary anymore.

E. Application on a Pioneer 3-DX

Like on most mobile platforms the odometry of the Pioneer 3-DX is erroneous. It is usually not sufficient to build a model of the environment from acquired data. We use the raw results of the described localization system without any Kalman- or particle-filter to register the camera images in real-time. The achieved accuracy allows to render a virtual camera image from the exactly registered images of the database. For that application it was not possible to determine the ground truth to compare our results with. Nevertheless, the fact, that we are able to merge the images

(a) 3D point-cloud: statue in front of a box. (b) 3D point-cloud of a putto.

Fig. 12. Two 3D point clouds acquired by the DLR 3D-Modeler using the presented visual navigation algorithm.

properly, even without any bundle adjustment, is a proof for accuracy. Fig. 13 illustrates the localization results on an image sequence acquired while performing a quarter circle with 0.9 m diameter (see Fig. 1). For further details about this work please refer to [21].

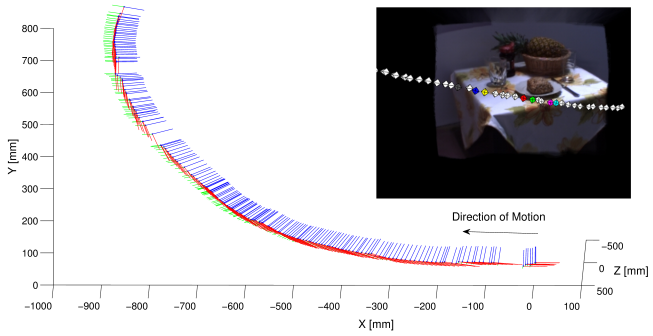


Fig. 13. Localization results of a Pioneer 3-DX performing a quarter circle as illustrated in the small subfigure. The blue lines show the looking direction of the capturing camera. The white gaps in the trajectory are due to swapping on the hard disk. The cameras were slightly tilted to the floor and so the trajectory is not only within the X-Z plane. The small picture in the upper right corner shows the virtually rendered model of the scene (see also Fig. 1).

V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an accurate and online-capable visual navigation technique for resource limited systems. The algorithm is based on KLT features tracked in a monocular image sequence. The KLT routines are sped up to allow a 25 Hz camera framerate also with limited computing resources. For stereo-initialization of the landmarks a novel fast and subpixel-accurate approach for stereo matching has been presented. The pose estimation algorithm is based on VGPS, but it has been robustified in order to detect bad features and exclude them from further processing. An intelligent hand-off and short range loop closure provides only a small error accumulation over longer terms. We have shown, that the robustification of VGPS keeps the error drift small enough to allow short range loop closure without Kalman or particle filter. By avoiding such filters the system's dynamics is not restricted which makes the algorithm applicable for high dynamic applications. Experiments provide proof of the framework's properties and illustrate the enhancements to the used algorithms. Also some results with a hand-held 3D modeler are presented. The algorithm allows unrestricted image-based online 3D-modeling without external referencing systems.

A problem of any hand-held modeler are the short, large rotational movements of the humans wrist. Such movements lead to large feature displacements between two consecutive images which prevents proper tracking. To overcome that problem, we currently test a combination with an IMU. Summing up the IMU data should ensure a better feature propagation. First results have proven that tracking can be done also during a wiggly sensor guidance.

VI. ACKNOWLEDGMENTS

We want to acknowledge the great support by the members of the 3D-Modeler project at the German Aerospace Center (DLR): Tim Bodenmüller, Wolfgang Sepp, Simon Kielhöfer and especially Gerd Hirzinger, head of the Institute for Robotics and Mechatronics at the DLR.

Special thanks also to Werner Maier for the virtual rendering results of the Pioneer experiment.

REFERENCES

[1] D. Burschka and G. D. Hager. V-GPS - image-based control for 3d guidance systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1789–1795, 2003.

[2] D. Burschka and G. D. Hager. V-GPS(SLAM): Vision-based inertial system for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 409–415, 2004.

[3] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2nd edition, 2003.

[4] D. Nister. A minimal solution to the generalised 3-point pose problem. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[5] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision*, volume 2, pages 1403–1412, 2003.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29:1052–1067, 2007.

[7] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, 2007.

[8] T. N. Thanh, Y. Sakaguchi, H. Nagahara, and M. Yachida. Stereo slam using two estimators. In *IEEE International Conference on Robotics and Biomimetics*, pages 19–24, 2006.

[9] A. I. Mourikis and S. I. Roumeliotis. A dual-layer estimator architecture for long-term localization. In *Workshop on Visual Localization for Mobile Platforms, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[10] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image retrieval system with monte carlo localization. *IEEE Transactions on Robotics*, 21:208–216, 2005.

[11] R. Sim, P. Elinas, M. Griffin, and J. J. Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 9–16, Edinburgh, Scotland, 2005.

[12] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[13] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[14] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[15] D. G. Lowe. Object recognition from local scale-invariant features. pages 1150–1157, 1999.

[16] P. J. Huber. *Robust Statistical Procedures*. SIAM, 2nd edition, 1996.

[17] K. H. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. DLR CalLab and DLR CalDe - http://www.robotic.dlr.de/callab/.

[18] T. Bodenmüller, W. Sepp, M. Suppa, and G. Hirzinger. Tackling multisensory 3d data acquisition and fusion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2180–2185, 2007.

[19] M. Suppa, S. Kielhöfer, J. Langwald, F. Hacker, K. Strobl, and G. Hirzinger. The 3d-modeller: A multi-purpose vision platform. In *IEEE International Conference on Robotics and Automation*, 2007.

[20] K. H. Strobl, E. Mair, T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger. The self-referenced DLR 3D-modeler. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[21] W. Meier, E. Mair, D. Burschka, and E. Eckehard. Visual homing and surprise detection in cognitive mobile robots using image-based environment representations. In *IEEE International Conference on Robotics and Automation*, 2009.

Data Abstraction Framework for Object-Consistent 3D Data Fusion

Juan Carlos Ramirez and Darius Burschka
Department of Computer Science
Technische Universität München, Germany
{*ramirezd*|*burschka*}*@cs.tum.edu*

Abstract

We present a mapping framework for consistent data fusion of 3D data in mobile exploration and manipulation systems. Our framework combines a pure geometric approach handling 3D data of the reconstructed objects and surfaces with a more symbolic representation of object candidates and supporting planes in the sensor view. Since we use the data not only for navigation but also for manipulation, an abstraction of the representation to clusters or even objects is necessary. Only this dual layer representation in our map allows a correct data fusion of partially observed objects with a sensor system with a limited field of view, like typical stereo camera system. The layered approach allows to cluster the geometry data into object candidates which are updated as a connected component also from partial updates. We present the layered structure of our approach and present the way how we associate data from real 3D sensor-based reconstructions in our map.

1. Introduction

Mission and path planning for mobile systems require a knowledge about the geometric structure of the world. This information can be provided a-priori from CAD models or explored with the sensors of the robot. Most existing exploration approaches were developed for mobile exploration. There exist systems like KARTO from SRI [1], which store the world as 2D maps to be able to navigate in them without any additional knowledge. On the other hand, manipulation tasks usually deal with single objects with known geometries. An exploration of the environment for manipulation is a challenging tasks that we aim to support with our approach. Our framework has the goal to support manipulation tasks in additional to traditional mobile exploration, therefore, a 3D representation of the world is required. An active exploration allows to update the map representation to the most current state. In a typical configuration, we assume that three-dimensional data together with the estimated position of the sensor in the world frame is passed to the map.

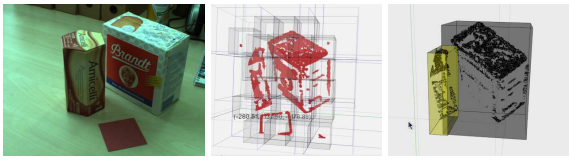


Figure 1. A camera image (left), its representation in the geometric octree layer (middle), and cluster layer with the two clusters representing the boxes (right).

One requirement on our map is a selective query to the information stored in the map. We want to predict only geometry visible in the current sensor cone. The underlying Octree structure provides here a good selectivity without the necessity of re-organizing the structure if new data is added (tree balancing). Additionally, we want to be able to update the geometry data in our map in a consistent way, so that clusters of points that are good candidates for objects will be updated correctly even if only part of the object can be perceived in the current view.

1.1. Related Work

Data association is considered as the discrete problem of SLAM and has been adressed considerably in SLAM and tracking literature [2]. Probabililstic methods have been applied succesfully relying basically on Bayesian filter approaches like the Kalman Filter (KF) family: EKF (Extende KF) [7], UKF (Unscented KF) and particle filter-based algorithms [6]. Others approachs have tried to improve data association by the

[1] `http://kartorobotics.com`

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

use of target attributes for a faster identification of targets [3] and reducing the computational cost of the process.

2. Approach

In this work, we have implemented a double-layered representation scheme to fuse the captured camera readings to meet the requirements stated in Section 1 (Fig. 1). The sensor provides basically 3D points which need to be stored at their geometric position together with their current accuracy. Our system needs to be capable of decision about the current quality of the points stored in the map. The 3D raw points are stored inside a local geometric map and treated as independent 3D points, they represent the spaces that are occupied by possible objects in the 3D scene.

Once data is stored in the map, any information about neighborhood relations between 3D points is lost. A cup standing on a desk results in a continuous set of points representing both. Since we need to assume that the sensor readings are not perfect and the pose estimation for the sensor views may be erroneous as well, we need to be able to correct the geometry with each sensor reading. Therefore, we cluster points to clusters based for example on the ”supporting plane” subtraction which is easily done in the disparity images [4]. In case, that in a consecutive view just part of the object is visible, we need to combine the corresponding points in an upper layer in our approach.

In this section we introduce the map structure and the fusion that constitute the relevant procedures in this work.

2.1. Map Structure

By a map we are referring to the computational data structure DS used principally to store the 3D points produced by the stereo reconstruction. Among the large variety of DS we chose the so called octree. This DS is widely used in computer graphics since it allows keeping 3D data in a hierarchical and recursive fashion. As its name indicates, this map gives us a basic raw representation of a real environment. The two layers of representation of the 3D data correspond to two different maps: the input map represented by the octree is a static one, that is, it does not change in time since it is created and available in advance by the mobile mechanism carrying the cameras and corresponds to the lower layer in this scheme (Fig. 2).

The abstract layer of our map is represented by a *blobtree, i.e.* an *octree* of 3D blobs, and constitutes a higher level of abstraction that not only simplifies the

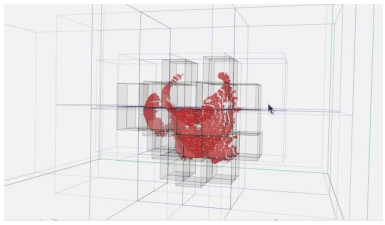


Figure 2. A cup in the geometric layer

map management and processing of 3D points but also allows us to treat and consider a cluster of points that share a common space or feature as an entity that conduces to the object recognition and classification. The Fig. 1 shows the different clusters of points that belong to the two different boxes on the table depicted on the left side.

Our mapping system *Scouter* queries the octree representation to return for each query only the visible elements contained in the sensor cone (Fig. 3).

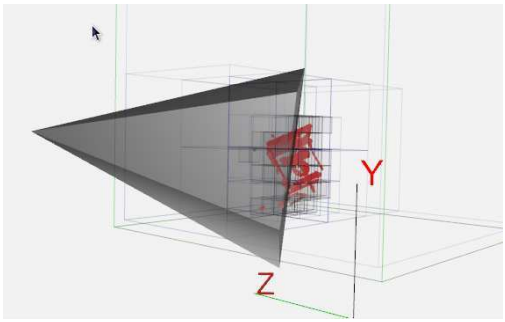


Figure 3. The data prediction with a sensor cone access to the octree structure.

The *Scouter* has basically the capacity of navigating freely throughout the input map and to take “pictures from any position and orientation inside it. This prediction of the view can be used by the sensor to disambiguate matches in the stereo processing, because in many cases the system needs merely to confirm the correctness of the prediction instead of the complicated and error-prone search for new correspondences between stereo images. Although the camera has an unlimited range, due to the shape of the disparity function only a limited reconstruction region is reasonable. It is represented as a sensor cone in Fig. 3.

2.2. Data Fusion

The data fusion is performed pair-wise between two blobs once two conditions are detected: *i*) they share a

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

common space; *ii*) they have points in common. The first condition is detected readily inside the *blobtree* by checking for overlappings among all the blobs, for the second point we resort to techniques related to data association issues. The abstraction of the geometry to the abstract blob representation simplifies the test, because just the rectangular hulls of the point clusters need to be intersected instead of searching through the entire geometric database for possible matches. In this way, many impossible possibilities can be pruned without any further investigation.

Data Association *DA* or *correspondence problem*, is the main problem in multiple target tracking that in turn arises in many fields such as Computer Vision, sensor networks and error correcting codes. In this mapping context *DA* is defined as that of associating or determining the proper correspondence between a set of current observations or measurements y_i with an already existing set of data h_i, involving in the first set the presence of distorted measurements due to noisy sensors and the presence of an inherent object dynamics in the other set. Although our reconstruction process in the geometry map is distorted by noise we know the uncertainty of each reconstructed point and determined the possible correspondence of any pair of sets for the fusion of their corresponding blobs. Figure 4 shows a pair of different point sets that belong to the same scene taken from different angles and positions.

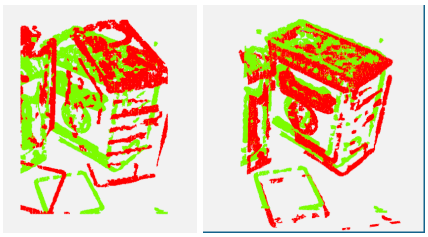


Figure 4. Point sets: (left) directly in the sensor frame at two different positions, (right) after alignment with localization result of the robot.

It is worth to mention that these point sets have been previously rotated and translated in order to be aligned and a subtraction of the supporting plane [4] was applied in order to filter all the points that do not belong to possible objects. As we can observe in the Figure 4 such alignment is not enough to determine with accuracy the correct pose of an object. Once we have computed the possible correspondences, we obtained a new set of corrected points (figure 5) representing the possible real positions out of the matching points.

We need to decide for each new point cluster if it can be part of an existing cluster in the map. The first simple test is an intersection between the hull of the new cluster in the sensor reading and the hull of a blob in the map. If an intersecting region is detected then a more detailed test for corresponding points is performed to check if there is some overlapping geometry between these two objects. Note that only parts of both clusters may belong to an overlapping set.

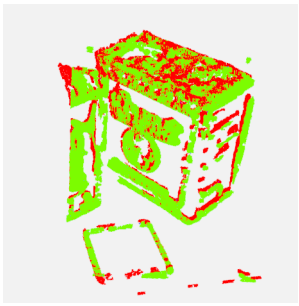


Figure 5. Corrected point sets.

In practice, computing the possible correspondences between two large sets of blobs could be a very expensive procedure. In our case we form a subset of one of the sets by randomly picking a fixed number of points and with this reducing the computational cost. This fixed number can be a small percentage of the total, in this trial we selected 1000 points out of approximately 52000. Figure 5 shows all the matching points of a run with all the 52839 points in one set and 46486 in the other.

In the next step we apply RANSAC [5] to determine the homogeneous transformations that relate with the smallest error each of the noisy sets with the corrected one. This is achieved by the Arun's Algorithm [1]. We assume two corresponding point clouds $\{P_i\}$ and $\{P'_i\}$, which are only rotated and whose rotation we want to estimate. First the origin of the coordinate frame has to be moved to the center of the point cloud:

$$\bar{P} = \frac{1}{n}\sum_{i=1}^{n} P_i, \quad \bar{P'} = \frac{1}{n}\sum_{i=1}^{n} P'_i \tag{1}$$

$$P_i^* = P_i - \bar{P}, \quad P_i'^* = P'_i - \bar{P'} \tag{2}$$

Now, the non scaled sample cross-covariance matrix for these point clouds is calculated to

$$\tilde{M} = \sum_{i=1}^{n} P_i'^* \, P_i^{*T} \tag{3}$$

Therefore, $\frac{1}{n}\tilde{M}$ is the sample cross-covariance matrix between $\{P_i\}$ and $\{P'_i\}$. It can be shown that the rotation matrix which minimizes the objective function also fulfills

$$\tilde{R} = argmax_{\tilde{R}}\, tr(R\tilde{M}) \tag{4}$$

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

Is $(\tilde{U}, \tilde{\Sigma}, \tilde{V})$ the SVD of $\tilde{M}$

$$\tilde{M} = (\tilde{U}, \tilde{\Sigma}, \tilde{V}) \tag{5}$$

then $\tilde{R}$ can be calculated to

$$\tilde{R} = \tilde{V}\,\tilde{U}^T \tag{6}$$

$\tilde{R}$ is orthonormal, symmetric and positive definite. However, it can happen that all features lie in a plane. In that case not the rotation matrix, put a mirroring matrix is calculated. Such a result can be recognized by the determinant of R, if $det(\tilde{R}) = -1$ instead of $+1$ and compensated by inversing the last vector from V. The translation between the point clouds can be calculated by compensating the rotation and subtracting the two middle points from each other.

Figure 5 shows the final result used for fusion in the map.

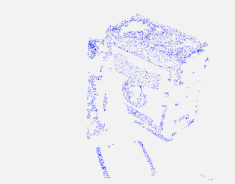


Figure 6. Reduced set of points used for calculation of the pose correction between the new update and the current map content.

Each point in the geometric representation is stored with its uncertainty σ_i that initially describes the 3D uncertainty due to detection accuracy in the sensor image Δd and calibration errors. The σ_{mi} from the corresponding point in the map and σ_i from the current sensor reading are used to define position of the resulting point in the map after fusion.

The resulting point in the map is estimated based on the corresponding σ values in the map and from the current sensor reading. We get finally two rotations: one of the sensor reading to the fused position and one of the cluster content in the map to the calculated new position. These rotations need to be applied on the entire cluster in the map and also on the entire cluster detected in the current sensor reading to keep the object representation consistent. Without this step, partial updates of the object would result in a split of the corresponding point cloud and a deformation of the corresponding object shape.

2.3. Experimental Setup

Our framework *Scouter* was implemented on LinuxOS. We used a binocular setup with two Applied Vision Marlin cameras with a focal length of 8mm. A typical update of the map with approx. 52.000 points takes in the current implementation approx. 5s. We have not optimized the matching process yet. We plan to use z-buffering to further reduce the time to come close to real-time operation with the stereo system, which is currently running with 10Hz on a Pentium-M 1.2GHz Dual Core system.

3. Conclusions and Future Work

We presented a data fusion framework, which aims to support not only mobile exploration but also manipulation. Manipulation relies strongly on consistent data representation of objects, therefore, we needed to implement a true 3D map for the shape representation and, additionally, we needed to ensure that the clusters in the scene get corrected as an entity in each update step independent of the current visibility. The future work will focus on speeding up the matching process by using computer graphics techniques to accelerate the search for corresponding points. We plan to use the data ordering in the sensor reconstruction to reduce the number of possible correspondences.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, September 1987.

[2] Y. BarShalom and T. E. Fortmann. Tracking and data association. 1988.

[3] S. Blackman and R. Popoli. Design and analysis of modern tracking systems. 1999.

[4] D. Burschka and G. Hager. Stereo-Based Obstacle Avoidance in Indoor Environments with Active Sensor Re-Calibration. In *International Conference on Robotics and Automation*, pages 2066–2072, 2002.

[5] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[6] D. Hähnel, W. B. D. Fox, and S. Thrun. A highly efficient fastslam algorithm for generating cyclic maps of largescale environments from raw laser range measurements. *In Proceedings of IROS03.*, 2003.

[7] J. V. Miro, G. Dissanayake, and W. Zhou. Vision-based slam using natural features in indoor environments. *In Proceedings of the 2005 IEEE International Conference on Intelligent Networks, Sensor Networks and Information Processing.*, pages 151–156, 2005.

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans

Susanne Petsch and Darius Burschka

***Abstract*— We propose a system for vision-based estimation of manipulation-relevant properties of objects in natural scenes based on observation of human actions. The system consists of an a-priori *(Atlas)* knowledge about known generic objects in the scene and classifies the scene into mission relevant objects and background geometry that is important only for collision avoidance. We present the object-centric structure of our system consisting of an *Atlas* representation and a *Working Memory* storing the current knowledge about the scene, the manipulated objects and actions applied to them in the local environment.**

We present experimental results how the system maintains the information in the database and we show the quality of the results that can be obtained with our system.

I. Motivation

Cognitive systems need to be capable of identifying the mission relevance and of learning the model description of objects by themselves during a joint action with a human operator. Most generally, a model of context specifies the entities to observe, the properties to measure and the relations to detect according to [17]. Dey [6] proposed an operational model for context aware perception. In this model, a situation is defined as a configuration of entities and relations relative to a task. The task serves to determine which entities and relations are of interest and should be observed. We transfer these findings into our environment representation, which allows to decouple complex object recognition loops from the low level 3D reconstruction.

Sensation and perception are key components of cognitive systems. Cognition can be defined as "generation of knowledge on the basis of perception, reasoning, learning and prior-models". Perception is the main source of information for reasoning and learning capabilities. Scene classification is an important task in cognitive systems. It helps in sensor-based 3D model generation to discriminate between objects interesting for missions (*foreground*) and *background* objects relevant merely for localization and obstacle avoidance.

A cognitive system is one that is capable of interacting with humans and other systems in an environment and that is capable to respond to an unexpected event that we will refer to as a *surprise* in the following text. Our system uses the *surprise* to control the learning about the scene and to trigger its own actions as responses to the external stimuli in the environment. We aim to develop a knowledge representation that allows to define manipulation actions based on the current action context and the estimated object properties from observation of human actions and previous interactions with the given object. The distinction between *foreground* and *background* elements allows the system to deal with a possible high complexity of the scene. It focuses the processing only on the structures relevant for manipulation. Our system observes a human operator who identifies the mission relevant objects through a direct interaction with them (manipulation). This way, our system does not need to identify and to learn about all objects in the scene but only about the objects that were used by the human. These objects define the *foreground* layer of our representation while the geometrical model of the entire scene remains as a global three-dimensional structure in a *background* layer. Only a contact of a human hand with an object followed by a change of its position renders the action as something that the system should know about (Fig. 1).

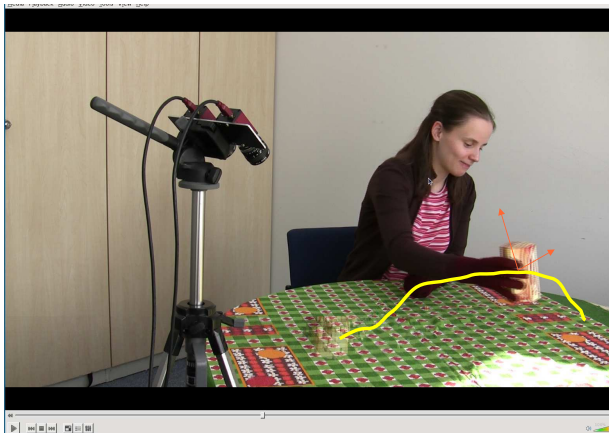


Fig. 1: System observes human actions and completes the internal knowledge representation for objects relevant for a manipulation task.

The target selection task is a challenging part of the system and can be implemented as a manual or automatic process. Examples in 2D image space are described in [14], [13] in more detail. Interesting targets like single standing objects in the scene need to be separated from the supporting planes of the table and floor that are merely relevant for collision avoidance.

Single standing objects are categorized as *foreground* iff a human operator interacted with them or iff they are known to be mission-relevant from previous actions. They need to

This work was supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project)

Susanne Petsch and Darius Burschka are with the Machine Vision and Perception Group, Department of Informatics, Technische Universität München. 85748 Garching, Germany
{petsch|burschka}@in.tum.de

be separated from the environment structure (*background*) first. In an additional step, the remaining *foreground* objects are classified according to their shape, appearance, and their observed allowed motion relative to the scene. The *background* structures are used in a subsequent classification process to estimate the scene context for the current mission.

We consider in general visual and haptic perception as the stimuli generating the input for our cognitive processing. This multi-modal sensor input allows to extract the initial information about *foreground objects* in the scene, to classify them, and to match them to already known representations in the *Atlas (long-term memory)* (Fig. 2). In this paper we focus on visual observation as a first step to acquire an initial guess about the object properties from its appearance.

The paper is structured as follows: in the next section we present details of our approach. We present the way how the a-priori and working knowledge about the actual environment is represented and how the processing of the robot is implemented. In Section III, we present our experimental results showing the different steps of the processing chain. We conclude in Section IV with an evaluation of the current system and present our future work in this area.

A. Related Work

Modayil and Kuipers [10], [11] developed a method where a learning agent can autonomously learn about object models, by detecting, tracking, and characterizing clusters of foreground pixels in the sensory stream. Their agent is a mobile robot that receives a stream of sensory information from a laser range-finder. Grauman and Darrel [7] learned feature masks for object categories by embedding sets of unordered image features into a space where they cluster according to their partial-match correspondences. Weber et al. [16] focused on learning object models that are represented as flexible constellations of rigid parts. Savarese and Fei-Fei [12] proposed a model to represent and learn generic 3D object categories by linking together diagnostic parts of the objects from different viewing points. All these methods learn models for particular objects or object categories from a database of static images under different viewpoints and different backgrounds. Our approach works in 3D space providing a more robust segmentation and registration performance.

In the field of object tracking, Comaniciu et al. [5] proposed a kernel-based tracking algorithm where an object is represented by an ellipsoidal region in the image and the mean-shift tracker maximizes the appearance similarity iteratively. Isard and Blake [8] presented a particle filter based tracking algorithm where object shape is represented by B-splines. Yilmaz et al. [19] proposed a contour-based tracking method using the color and texture models in a band around the objects boundary. Tran and Davis presented a robust object tracking method using regional affine invariant features [15]. In our approach, we use our previously presented 6DoF system VGPS that tracks the structure in monocular images and provides in real-time all six motion parameters.

II. Approach

The robot needs to know about the geometric and physical properties of the object to perform a successful manipulation. Hypotheses about the possible grasp points for the robotic manipulator need to be generated based on the shape and the physical properties like mass and friction of an object. The properties that we currently consider as important for a successful manipulation are: mass, center of gravity, shape to find appropriate surfaces for a successful grasp with a given manipulator, and allowed actions that can be applied to an object. Not all of these properties are observable with a camera and, therefore, we use an additional information database *Atlas* in our system (Fig. 2) to represent the "experience" (*a-priori information*) of the system.

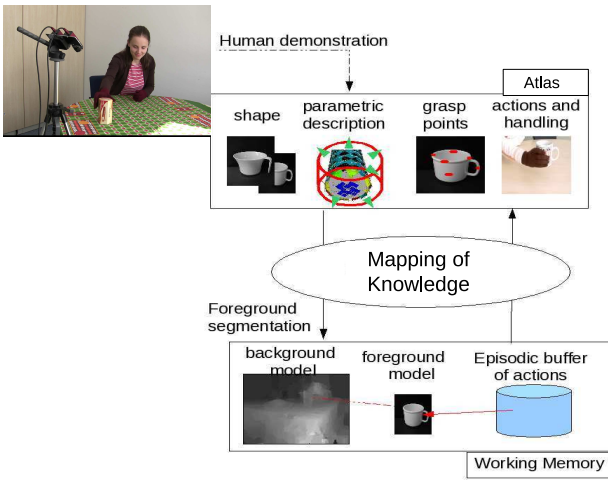


Fig. 2: The system moves the knowledge from a-priori database (Atlas) and instantiates it in the Working Memory representing the actual setup of the manipulation task.

We use for the knowledge representation in the Atlas an analogy to the cognitive capabilities of the human brain and its different strategies, how to store and process the information in the most efficient way. The brain does not store memories in one unified structure. Instead, different types of memory are stored in different regions of the brain. *Long-term memory* in the brain is memory that can last as little as a few days or as long as decades. It differs structurally and functionally from *working memory* or short-term memory, which stores items for only a short time. Working memory (also referred to as short-term memory, depending on the specific theory) is a theoretical construct within cognitive psychology that refers to the structures and processes used for temporarily storing and manipulating information. There are numerous theories as to both the theoretical structure of working memory as well as to the specific parts of the brain responsible for working memory. Baddeley and Hitch (1974) introduced and made popular the multi-component model of working memory [1].

We follow the structure suggested by Baddeley with the long-term memory and the short-term memory maintained

by the central executive (Mapping of the Knowledge in Fig. 2). Our system consists of two databases storing a-priori knowledge about the world (*the Atlas*) corresponding to the long-term memory and a *Working Memory* representing the current visual an spatial representation of the world (visuospatial sketchpad). In this layer, the episodic buffer is implemented as a system storing the typical actions applied to a mission relevant object.

The two layers (Fig. 2) have the following representation:

- **Atlas Representation (Experience of the System)** - this information represents a-priori knowledge given to the system from an expert or representations of the environment collected in previous operations in the same or similar environment. An important difference of the proposed system to many other systems suggested before is that it is supposed to interact with its environment in a cognitive way. This means that the system does not operate based on a set of pre-defined rules but it tries to learn from its own actions and actions of other agents in the environment (human or other robots). The information stored in the Atlas represents a generic knowledge about a class of object.
- **Working Memory**- Working memory is a theoretical construct within cognitive psychology that refers to the structures and processes used for temporarily storing and manipulating information. In our system, the *experience* needs to be grounded to a given environment. We expect to operate in highly complex environments, where the system must not try to analyze all elements of the scene as it is often the case in other current manipulation systems but it needs to focus its *attention* on mission relevant objects whose properties need to be explored for a successful interaction with the world.

An important novelty in the presented system is that the objects are represented not only with their spatial and physical properties (shape, mass, friction) but include also temporal handling information which is essential for the system to handle the object with the same constraints regarding its orientation relative to the gravity vector and accelerations in the translational and rotational motions as presented by the human. The following processing chain allows us to extract this information from the visual system of the robot.

Our system (depicted in Fig. 3) contains the entire processing chain for the visual interpretation of a human action. It starts with the detection of candidates for mission-relevant objects in the world using in our first implementation a simple Supporting Plane Removal algorithm presented already in [2], [4]. In the next step, we use our Vision Interaction Cues (VICs) approach [18] to speed up the processing of human actions. In analogy to VICs, each object defines its own actions and defines a monitoring space around itself. In our system, the human triggers any new knowledge acquisition by presenting new actions to the system. Each cluster segmented in the initial segmentation step defines an *interaction space* where gestures are actually analyzed. It is only necessary to do it if the hand is in the vicinity of a given

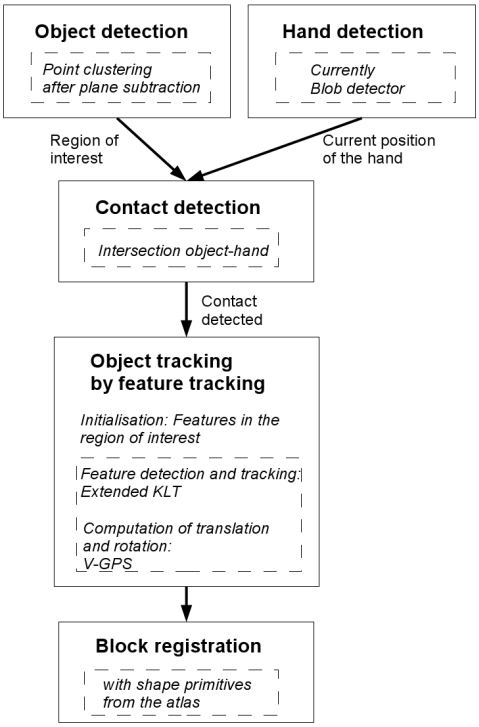


Fig. 3: Overview of the approach. The boxes represent the single modules of the system. Each box contains a dashed box, where the implementations can be found, which are used for the realization of the modules here.

object. Once a grasp gesture at a given cluster is detected the system starts tracking the 6DoF pose of the object to understand the action performed by the user. It stores the corresponding trajectory for later analysis until the object is released. In a final step, a registration step is performed to match the given cluster to the known geometries in the *Atlas* using the shape representations stored in there. The system has the choice to use direct shape registration for known objects or parametric shape analysis to categorize the shape to a specific generic class representation in the *Atlas*.

A. Knowledge Representation

We can tell from Fig. 2 that the *Atlas* contains several distinctive object representations that provide information which is important for the recognition of an object (geometric shape for direct 3D shape registration, and parametric shape description for generalized object class representation) and additional information which is important to initialize parameters which are not observable by the system. These additional parameters are mass, center of gravity and friction leading to specific grasp point representation, and actions that are known to be associated with a given object (e.g., motion constraints on cups or glasses that my contain water).

This a-priori information (*experience*) from the *Atlas* needs to be mapped on the current environment representation surrounding the robot, which is stored in the *Working Memory*. The *Working Memory* contains the geometric shape description as well which is now complete in opposite to the

current sensor reconstruction that usually provides only a partial view due to occlusions in the scene. The registration step to the *Atlas* information allows a completion here. Additionally, now the system is able to store also the texture information representing the appearance of an actual instance of an object in the scene. Now we know not only that there is e.g., a cup, but we also know that this is a cup with a specific texture or logo on it. We move the initial hypothesis about the grasping points and actions from the *Atlas* to the *Working Memory*. Finally, we get also hypotheses about the mass range, center of gravity position, friction and stiffness of the object as an initial guess for the first interaction of the robot with the object. This information is provided as a container for other processing steps and not considered in this paper.

B. Action Representation

An important novelty in our object description is the representation of the temporal changes to the object. We decided to use an object-centric representation of actions. We consider the robot and the human as agents that can imply changes to the state of an object. We are interested in this context only in three phases of the change depicted in Fig. 4

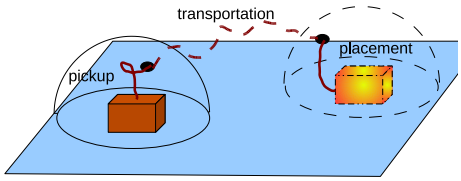


Fig. 4: Three phases defining an action: the type of pickup, the way the transportation is done, and the placement of an object.

The pickup and the placement is mostly concerned with the grasp type performed by the human operator and not part of this paper. This is an information, which is important for an emulation of the grasp by the robot and requires a hand gesture recognition which is out of the scope of this paper. In this paper, we are interested in the analysis of the transportation phase of the action. It is important for us, how free the motion of the object can be (which enforces constraints of coupling between the joints of the robot to ensure a specific orientation relative to, e.g., the gravitational vector) is and where the object is usually placed in the scene. We found it not necessary to save any actual trajectories presented by the human since our focus is on a detailed description of object properties here and the repeatability of the trajectory is relatively low in most cases. For an object it is not important which way it took through the environment but only how it was handled (speeds, orientations) and where it was picked up and placed. This is the information that we need to extract from the vision system.

C. Scene Clustering

In order to detect relevant objects, a plane-subtraction is applied first, which is described in [2], [4]. The approach uses the fact that there is a homography between the (u, v, D) coordinates of the disparity image ($[u, v]$-image coordinates and disparity D) and the corresponding Cartesian coordinates from the 3D scene. According to [2], the planar surface P_r can be represented as

$$P_r : a_r x + b_r y + c_r z = d_r. \tag{1}$$

It is shown in [4], that the equivalent disparity plane is given by

$$D(u,v) = \begin{pmatrix}\rho_1\\ \rho_2\\ \rho_3\end{pmatrix} \cdot \begin{pmatrix}u\\ v\\ 1\end{pmatrix} = n_r^* \cdot \begin{pmatrix}u\\ v\\ 1\end{pmatrix} \tag{2}$$

with the disparity $D(u, v)$ at image coordinates (u, v), $\rho_1 = \frac{a_r}{k}$, $\rho_2 = \frac{b_r}{k}$, $\rho_3 = \frac{c_r}{k}$, $k = \frac{d_r}{B}$ and baseline B.

The next step is the search for the planar candidates. These candidates depend on the gradient of the disparity-map: A high gradient or low gradients with different directions refer to the border of a planar plane, whereas pixel with low gradients of the same direction form a plane. The biggest area with low gradients of the same direction is assumed to be a part of the plane. This area is used for the estimation of the normal vector n_r^* of the plane. The vector n_r^* is estimated according to (6) in [4]:

$$\begin{pmatrix}\sum u_i \cdot D_i\\ \sum v_i \cdot D_i\\ \sum D_i\end{pmatrix} = \begin{pmatrix}\sum u_i^2 & \sum u_i v_i & \sum u_i\\ \sum u_i v_i & \sum v_i^2 & \sum v_i\\ \sum u_i & \sum v_i^2 & \sum 1\end{pmatrix} \cdot n_r^* \tag{3}$$

The direction vector n_r^* enables a comparison between the observed disparity of a pixel and the expected disparity of the plane at the position of the pixel according to the direction vector n_r^* of the plane. If the difference between both is higher than a certain threshold, the pixel is assumed to not belong to the plane. All pixel, which belong to the plane, are deleted in the disparity-map. Consequently the objects, which are placed on the plane, remain in the disparity-map. An example of the plane subtraction is given in Fig. 5.

Fig. 5: Results of plane subtraction. *Left column:* Original color image. *Middle column:* Disparity image of the color image on its left. *Right column:* Remaining object in the disparity image after the plane subtraction.

For the further processing the outer bounding box of the object as the biggest connected component is taken as region of interest. Any other representation could be applied as well.

D. Parsing of Human Action

The manipulation of the objects is going to be parsed as follows. The first step is the detection of the contact of the object and the hand, which will take the object. The position of the object is detected as described before. Since

the position of the object is not changing until its contact with the hand, the computation of the position of the object has not to be computed again. The position of the hand can be determined in different ways, a blob-detector is used here. Therefore the color-image is split in HSV-planes and appropriate thresholds are applied. Just the pixel with the color of the hand remain in the image. If the hand touches the region of interest, a contact is detected. Otherwise the procedure for the contact detection is repeated until a contact is detected.

After the detection of the contact between the hand and the region of interest (the object), the tracking of the features of the object is initialized. Features are selected in the region of the object. If an outer bounding box is used as region of interest, there will be features, which are not on the object and cannot be used for the tracking of the object. The positions of these features do not contain disparity after the plane-subtraction and the features can be deleted. The valid features on the object are used for the tracking of the manipulated object. The contact between the hand and the object is assumed to be lost, when the object and its features are not moving (first trigger) any more and a separation from the object was detected (second trigger). Therefore the tracking of the objects features is finished when all features stop moving. The extended KLT [9] is used here for feature detection and tracking. An example of the contact detection and the tracked features is given in Fig. 6.

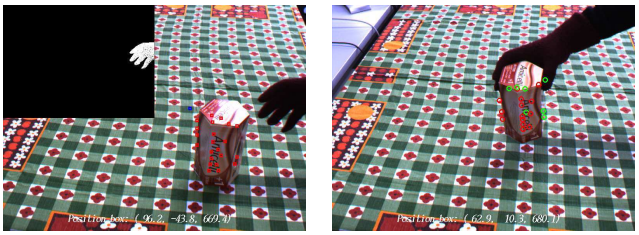


Fig. 6: Contact detection and object tracking. *Left:* The tracking is initialized after the contact detection between the hand and the region of interest. The small red boxes are the valid features, whereas the blue ones are the deleted features, which are not on the box. The top left corner of the image shows the position of the hand, when the contact occurs. *Right:* Example of features during the tracking. The tracked features are shown in red, the assumed position of the lost features are drawn in green.

The recorded trace of the tracked features of the manipulated object enables the computation of its rotation and translation. V-GPS is used for the computation of the rotation and translation [3]. The computed angles and the translation during the movement determine the possible movements of the objects. Additionally the trace of lost features can be reconstructed.

III. Results

In this section the results of the experiments are presented. The used sequences (seq.) have different motion properties, shown in Table I. The movement was either a straight line or an arbitrary motion, the object was either tilted or not. All movements were tested with two different boxes. The scene was recorded with a Firewire Marlin FO46C camera. The following settings were used: image size = 780x582 pixel (width x height). OpenCV, XVision, extended KLT [9] and V-GPS [3] were used in the algorithm, which was running on a Linux system.

TABLE I: Properties of the used sequences

Seq.:	Box 1	Box 2	Movement	Rotation	# Images
1	x		line		705
2	x		line	x	740
3	x		arbitrary		1055
4	x		arbitrary	x	1035
5		x	line		725
6		x	line	x	870
7		x	arbitrary		860
8		x	arbitrary	x	1600

A. Clustering of Object Candidates on a Table

The first part of the experiment is the plane subtraction, as described in II-C, in order to get the position of the object as the region of interest. A sliding-average window was used to get a fill holes in the disparity-map, although that results in smoother transitions between an object and the plane. Just one sequence (seq. 6) required the modification of two additional parameters (a larger kernel for the expected size of the ROI and a higher number of neighbors considered for the comparison of the direction of the gradient) because of a too smooth transition between the object and the table, which included the box as a candidate region for background subtraction.

Fig. 7 shows the result of seq. 6. The result of seq. 3 has already been presented in Fig. 5. The ROI is successfully detected for all sequences, the size of the all ROI is given in table II. The boxes used for the experiments have different sizes (box 2 is larger than box 1), therefore, the algorithm computes correctly a larger ROI for box 2.

Fig. 7: Results of plane subtraction (Seq. 6). *Left column:* Original color image. *Middle column:* Disparity image of the color image on its left. *Right column:* Remaining object in the disparity image after the plane subtraction.

B. Tracking of Human-Induced Motions on the Objects

After the detection of the ROI, the manipulated object is tracked as described in II-D. The tracking is initialized when a contact between the hand and the object is detected. The feature tracking is implemented with extended KLT tracker [9]. The rotation and translation are computed

with V-GPS approach [3]. The rotation and translation is also used for the reinitialization of lost features, since the assumed position of the lost feature can be computed from the estimated rotation and translation of the object. The initialization of the tracker and features during the tracking have already been shown in Fig. 6. An example for the used feature set and the object trajectory is shown in Fig. 8. The trajectory is computed from the position of the tracked features using V-GPS. All sequences show that the trace of a tracked arbitrary feature in the sequence is similar to its projected 3D trajectory.

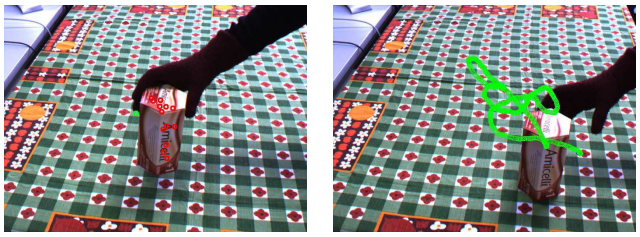


Fig. 8: Tracking of the object (6DoF from monocular) (Seq. 3). *Left:* Example of feature set used for tracking. The tracked features are shown in red, the predicted position of the lost features is drawn in green. *Right:* the object trajectory is shown in green.

Table II contains also the number of features at the beginning and at the end of the sequence. Additionally, the number of features, which were tracked during the whole sequence without a reinitialization, can be seen. The number of features which are tracked during the whole sequence without any reinitialization decreases with the length of the sequence (seq. 3,4,8) and the influence of rotation (seq. 2,4,6,8). The table shows also that the reinitialization of lost features is successful, especially seq. 8.

TABLE II: ROI and number of tracked features

Seq.:	Size ROI (pixel)	# features:	start	end	whole seq.
1	27.945		20	15	7
2	25.488		22	13	4
3	27.800		14	11	3
4	20.088		11	8	2
5	39.026		10	9	8
6	48.830		16	9	5
7	52.398		41	37	30
8	52.832		34	21	7

C. Analysis of the Trajectories

The calculated information about the rotation and translation of the manipulated object during tracking enables the computation of several properties of the manipulation. As already described in III-B, the computation of the object trajectory is possible. Furthermore, the rotation of the object can be computed at each step as well as the speed of the object along the trajectory. Fig. 9 shows the rotation, translation and speed of the object in seq. 7. The orientation of the vertical axis of the object is drawn every 50 steps. We assume, that at the beginning of the trajectory the object orientation is aligned with the calculated normal vector of the table since we do not use any model information for the object. Moreover, Fig. 9 depicts the shape of the speed curve during manipulation. The speed at each position is the average of the past 50 steps in Cartesian coordinates in the 3D Scene. The speed increases during the task.

Fig. 9: Rotation, translation and speed of the object (Seq. 7). *Left:* The development of the rotated and translated normal vector of the table is shown in green. *Right:* The development of the speed along the trajectory is shown in different colors: green = no movement, yellow = slow movement, red = movement, blue = fast movement.

Fig. 10 shows the rotation, the translation and the speed of the manipulation in other trials. The rotation of the object is visible for seq. 4 and 8 while seq. 5 does not contain any rotation of the object. It is a translation along a straight line. Besides the shown rotations and translations of some sequences, the translation and (if applied) rotation of the objects have been drawn for all sequences. Fig. 10 contains also the shape of the speed during the manipulation of the object. The manipulation of the object in seq. 5 along a straight line shows clearly the increasing speed after the pickup, the (in average) constant speed during the transportation and the decreasing speed before the placement.

The results of the angle analysis between the original position of the object and its rotated position during the manipulation are in Table III. The magnitude of the average angle along the trajectory indicates if the object needs to be kept in a vertical orientation or can be tilted during manipulation. As it can be seen, Seq. 2, 6 and 8, which contain rotations, have a clearly higher average angle than seq. 1, 5 and 7 without rotations. In seq. 3 and 4 the system switched to wrong features during tracking resulting in a bias in angle estimates. As table II shows, the number of tracked features in seq. 3 and 4 is really low, therefore, it is obvious that the computation of the rotation and translation, which is based on these features and their number, is challenging for the complex movements in seq. 3 and 4. The fact that there is a small average angle for seq. 1, 5 and 7, which do actually not contain rotations, is also caused by the human operator, since it is hardly possible to move an object without any rotation at all. The results for the maximum angles between the original position and the rotated position (table III) show a similar result: The maximum angle of seq. 2, 6 and 8, which contain rotations, is much higher than for seq. 1, 5 and 7 without rotations. The remaining angle between the original

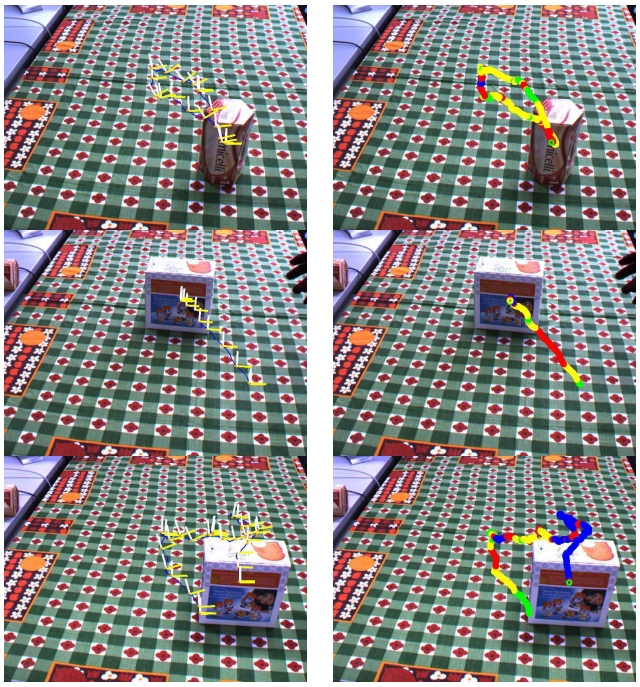


Fig. 10: Rotation and translation of the object in different sequences (Seq. 4, 5, 8). *Left column:* The drawn coordinate system shows the computed rotation and translation of the tracked object. The object trajectory is drawn in yellow. *Right column:* The development of the speed of the object trajectory is shown in different colors: green = no movement, yellow = slow movement, red = movement, blue = fast movement.

position and the final position should be close to zero, since the object is placed on the table again. The results in table III show, that there is a relatively high remaining angle in seq. 2 and 6. These sequences have a small number of constantly tracked features, similar to seq. 3 and 4. The remaining angle of seq. 7 reaches with 0.36 nearly zero. This sequence has the highest number of constantly tracked features among all sequences, therefore it can be concluded that the number of constantly tracked features influences the performance.

TABLE III: Analyzed angles of the sequences

Seq.:	Average	Maximum	Remaining angle (end)
1	4.28	11.11	5.54
2	10.36	31.99	19.69
3	10.08	20.72	4.57
4	6.60	14.23	1.83
5	4.40	10.87	4.86
6	11.30	21.50	20.07
7	5.47	11.28	0.36
8	10.33	25.99	4.08

IV. Conclusions and Future Work

The initial representation developed in the current system is the our testbed how to represent knowledge in a manipulation system and how to define action representations that are necessary for a successful surprise detection. The detection accuracy is already sufficient and will be improved through usage of a bifocal setup in the near future, where the object is observed with a long focal length camera that will allow an even better spatial resolution.

Our next goal is to focus more on the representation of actions in the local environment and to include them in the predictions of the system. We started already work on registration of generic shape descriptions that will allow a classification of objects to a global category. This will allow to provide a-priori suggestion about the manipulation capabilities of an object which may still be unknown to the system.

References

[1] A.D. Baddeley and G.J. Hitch. Working Memory. *New York: Academic Press, vol. 8*, 1974.

[2] D. Burschka and G. Hager. Scene Classification from Dense Disparity Maps in Indoor Environments. In *Proc. ICPR*, 2002.

[3] D. Burschka and G. Hager. V-GPS – Image-Based Control for 3D Guidance Systems. In *Proc. of IROS*, pages 1789–1795, October 2003.

[4] D. Burschka and G.D. Hager. Vision-Based 3D Scene Analysis for Driver Assistance. *ICRA*, 2005.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based Object Tracking. *Transactions on Pattern Analysis and Machine Intelligence, vol. 25*, pages 564–577, 2003.

[6] A.K. Dey. Understanding and Using Context. In *Personal and Ubiquitous Computing*, volume 5(1), pages 4–7, 2001.

[7] K. Grauman and T. Darell. Unsupervised learning of categories from sets of partially matching image features. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 2006.

[8] M. Isard and A. Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision, vol. 29, no. 1*, pages 5–28, 1998.

[9] E. Mair, K. Strobl, M. Suppa, and D. Burschka. Efficient Camera-Based Pose Estimation for Real-Time Applications. *IROS*, 2009, to appear.

[10] J. Modayil and B. Kuipers. Bootstrap learning for object discovery. *IROS*, 2004, vol. 1.

[11] J. Modayil and B. Kuipers. Autonomous Shape Model Learning for Object Localization and Recognition. *IEEE Int. Conf. on Robotics and Automation*, 2006.

[12] S. Savarese and F. Li. 3D Generic Object Categorization, Localization and Pose Estimation. *IEEE International Conf. in Computer Vision (ICCV)*, 2007.

[13] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. *Proc. of International Conference on Computer Vision*, 1998.

[14] S. Simhon and G. Dudek. Selecting targets for local reference frames. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2840–2845, 1998.

[15] S. Tran and L. Davis. Robust Object Trackinng with Regional Affine Invariant Features. *IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[16] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. *Proc. ECCV, vol. 1*, pages 18–32, 2000.

[17] T. Winograd. Architecture of Context. In *Human Computer Interaction*, volume 16, pages 401–419.

[18] Guangqi Ye, Jason Corso, Darius Burschka, and Gregory D. Hager. VICs: A Modular Vision-Based HCI Framework. In *Proceedings of 3rd International Conference on Computer Vision Systems*, pages 257–267, 2003.

[19] A. Yilmaz, X. Li, and M. Shah. Contour-based Object Tracking with Occlusion Handling in Video Acquired using Mobile Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 11*, pages 1531–1536, 2004.