Interactively Refining Object-Recognition System

Mike Eissele[1] Harald Sanftmann[2] Thomas Ertl[1]
[1]Visualization and Interactive Systems Group [2]Visualization Research Center
Universität Stuttgart, 70569 Stuttgart, Germany
{eissele|sanftmann|ertl}@vis.uni-stuttgart.de

ABSTRACT

Existing techniques for object recognition often make use of a combination of multiple algorithms and sensors to achieve adequate results. In this paper we propose a real-time system to efficiently combine multiple object-recognition techniques, appropriate for mobile Augmented Reality applications. We focus on the challenge to differentiate objects with only marginal distinguishing features that can often only be identified from specific points of view, and solve this problem by interactively guiding the user during the recognition process. The system is based on a hierarchy to organize model data and control the corresponding feature-detection techniques as shown in a prototypical implementation. Furthermore, recognition techniques are chosen based on context information, e.g. feature type, reliability of sensor data, etc.

Keywords: Multi-Technique Object Recognition, Mobile Augmented Reality.

1 INTRODUCTION

The problem of object recognition is a common issue in many real-world applications. Marker-based systems are efficient for object identification and pose estimation, but require a deployment of markers to target objects. In contrast, there are numerous scenarios where markers cannot be used, either because of esthetic reasons or technical problems. Therefore, other methods have evolved to provide a marker-less object recognition. For interactive applications it is further required that the recognition processed is performed in realtime. Examples are most Augmented Reality (AR) applications where mobile users interactively control the camera via direct manipulation.

Most available recognition techniques have specific advantages and disadvantages in certain situations. Therefore, many setups exist which make use of multiple sensors—optical, inertial, etc.—with different algorithms to achieve improved results. However, most of these systems are not applicable to mobile scenarios and are designed for very specific problems and do not focus on an easy extensibility with additional sensors or algorithms. Therefore, we proposed a general concept to combine arbitrary object-recognition techniques to build a robust, reliable, and efficient real-time object-recognition system. In contrast to existing sensor-fusion methods the proposed system detects and selects the most appropriate algorithm to differentiate objects based on various context information.

A specific goal of the proposed system is to efficiently differentiate object classes with a huge number of only marginal different entities. The differentiation can be so fine granulated that even individual object can be uniquely identified. Thereby, distinguishing features might not be captured by any of the available sensors from some locations. The system is targeted for interactive mobile AR applications where virtual geometry is accurately aligned with a real-world image as depicted in Figure 1b,c. Therefore, a primary challenge—when supporting multiple, possibly alternative, algorithms for object recognition—is to prevent a degradation of performance due to numerous object-algorithm combinations that have to be evaluated.

The proposed system prevents such a performance degradation by utilizing a hierarchical structure to organize model data. During the recognition process, the hierarchy is traversed and the number of possible object matches is continuously reduced. The hierarchy also allows to present intermediate recognition results, e.g. object classes, and trigger actions that are needed to further descent the hierarchy, if the system cannot differentiate an object. This triggers can effectively be used to initiate user-operated sensor adjustments on handheld devices that commonly lack of mechanical installation to, e.g., change the view direction of a camera.

2 CONTEXT AWARENESS

The usage of context information to support object recognition is already motivated by Oliva and Torralba in [13]. They show the importance of context for the human visual system and propose to use context data also in Computer Vision systems. They focus on semantic context information of captured scenes, in contrast, we

Figure 1: Steps of an interactive refining object recognition: a) recognition of the object class *Auto Data Switch* and presentation of a hint to capture further distinguishing features, as seen in b) or c). For illustration, different models of the *Auto Data Switch* shown in b) and c) are augmented with enhancements.

propose to widen this concept and further include arbitrary information related to the current situation.

Acquisition, (pre-)processing, and storage of arbitrary context data can easily be provided by Nexus, a framework for mobile context-aware applications [4, 5]. The open system offers the possibility to query basic context information or even an estimation of a high-level situation description. The concept allows to connect any data provider and any data consumer. The support of quality metrics further helps to weight the received context data. Using Nexus as an underlying service enables access to a variety of data—e.g. user position, lighting conditions, available hardware, etc.—to control and enhance the process of object recognition.

An overview of the proposed system is given in Section 4.1 followed by a detailed description of the hierarchical structure of model data. The execution of an object-recognition operation is detailed in Section 4.3. Prototype implementation and results are discussed in Section 4.5.

3 MULTI-TECHNIQUE OBJECT-RECOGNITION METHODS

For the survey of previous work, we primarily concentrate on methods which utilize multiple different techniques/sensors for the recognition of objects.

In general, research on object recognition can be separated in two categories: marker-based and marker-less techniques; both are supported by the proposed framework. Object identification and pose estimation based on synthetic markers is, amongst others, shown by Ababsa and Mallem in [1].

The advantage of using hierarchies in terms of decision trees is proposed, e.g., by Mehrotra et al. [12]. They group distinctive features of objects to generate the tree and traverse it during the object recognition phase. Our work is based on this general concept, however we extend several aspects: Support for multiple techniques to evaluate the decisions, even on a per-node basis, the possibility to return intermediate results, and therewith the triggering of actions to allow an unambiguous object recognition or even identification. Also, Viola and Jones propose to use a degenerated decision tree to achieve fast recognition results [16]. They concatenate multiple weak continue/reject classifiers to build stronger classifiers, however near-equal objects that cannot be differentiated in arbitrary captured views are not handled adequately. Grabner et al. use SIFT-like features in [8] to distinguish objects. The system groups similar objects in a hierarchical structure, however only a single recognition technique is utilized.

Dhome et al. propose a method to find an analytical solution for the attitude of a 3D object in space [7]. Simplifications for the special cases of coplanar lines and three-line junctions are given which reduce the problem to four-degree equations. Beier et al. present an application of Dhome's method on mobile devices [2]. In addition, a simple image-based 2D filter is used to differentiate similar objects. Lowe presents an algorithm that iteratively refines an initially guessed view point via Newton's method [11]. Kang et al. propose a technique to efficiently extract topology information, i.e. line junctions, within an image [9] and use it to perform object recognition and pose estimation [10]. Vacchetti et al. present in [15] a marker-less registration method based on the combination of image feature points and edge tracking. The authors compare different setups for sensor fusion and show that with multiple hypotheses the initial result can be improved. A system which uses vision and inertial sensing for tracking is proposed by You et al. [17]. An inertial sensor provides changes in orientation since the previous frame to estimate new camera orientations used as input for a Computer Vision approach.

If objects that have to be recognized are very similar often a single view is insufficient to distinguish the objects, independent of the applied method. Therefore, a number of systems have been presented that evaluate the best view of an object to achieve a reliable recognition. After a first recognition phase, these *active vision* systems build rules to, e.g., move the camera to a view, which allows further refining the object recognition. However, most active-vision approaches assume that a automated camera movement is available, which is practically impossible for handheld devices.

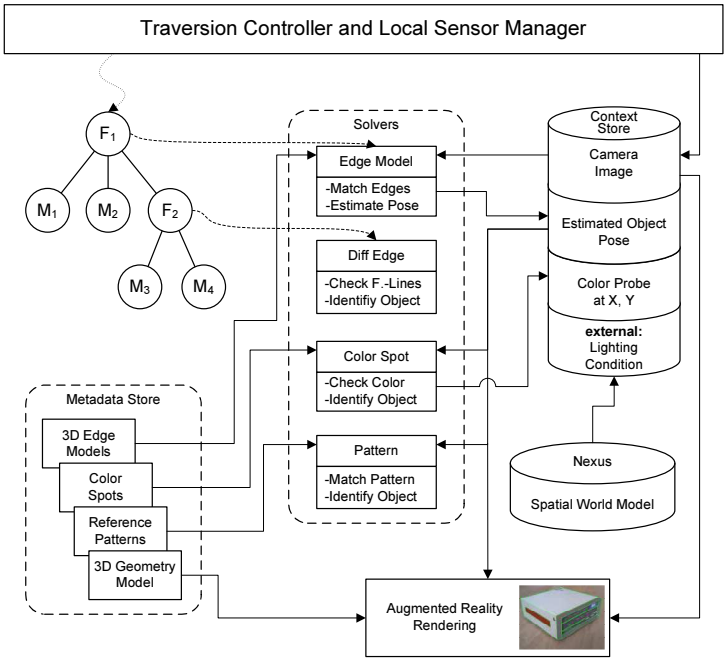

Figure 2: Overview of the object-recognition framework.

Borotschnig et al. present a comparison of three approaches for these so-called active object-recognition systems [3]. The examined techniques are based on different uncertainty calculi: probability theory, possibility theory, and Dempster-Shafer theory of evidence. Reinhold et al. present a statistical appearance-based object-recognition approach combined with an active view point selection [14]. Deinzer et al. presents a non-realtime method which uses a training set to learn *good* next views unsupervised [6].

4 HIERARCHICAL RECOGNITION

There are already existing methods (e.g. [9]) which provide promising results for recognition of dissimilar objects. Most are not well suited for the task of recognizing many similar objects. In contrast, similar objects motivate a grouping of objects with partially equal features, e.g. same overall shape. By iteratively repeating this division within the resulting subgroups an object hierarchy is generated, thereby each distinguishing feature may be detected using a different algorithm.

4.1 System Components

The architecture of our presented object-recognition system is illustrated in Figure 2. All algorithms for object recognition need some information about the models that have to be recognized. This can be reference images, texture information, 3D models, or any other information. Such information—in the following referred to as metadata—is typically generated in an offline preprocessing task for each node and stored in the *Metadata Store* as shown in Figure 2.

The data contained in the metadata store is primarily accessed by integrated solvers. The framework is based on the concept that multiple different solvers—seen in the center of Figure 2—are utilized subsequently during the object recognition. This way, arbitrary object-recognition techniques using various metadata can be integrated and combined to calculate the final result of the object recognition. In addition, solvers have access to a further data source the so-called *Context Store*.

The context store's content can be considered as data that describes the current conditions of the application, environment, or any other attribute which might dynamically influence the recognition. We use this store for locally acquired sensor data, intermediate recognition results, and *external* context data from Nexus [4].

The previously mentioned hierarchy with different algorithms (F_x) to detect features is shown in the upper left corner of Figure 2. The traversing of this hierarchy is controlled by a simple controller which executes the referenced solvers (Fig. 2, dashed arrows).

4.2 Organization of Recognition Nodes

The hierarchical structure used for the recognition (recognition tree) utilizes several node types that define different behaviors that are triggered during the traversal of the graph.

A basic node type is the *Search Node* F. It references multiple alternative solvers S_x that are adequate to iterate through all of the Search Node's children and search for the best matching. A behavior similar to simple traditional recognition systems that iterate through all integrated object models can be simulated this way, where all models M_x are checked for a match in se-

quence. A corresponding example recognition tree is shown in Fig. 3. During traversing, solver S_A and S_B access corresponding metadata, referenced by the current node, e.g. relevant line features of real-world objects.

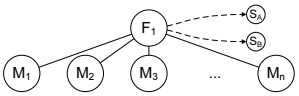

Figure 3: A simple recognition tree to simulate a traditional recognition system which linearly checks each model to search for the best match.

An advantage of the hierarchical structure is that it allows to group similar objects based on common features. These intermediate group nodes reference common metadata of all their child objects that is stored in the *Metadata Store*. Meta information that is only adequate for intermediate nodes may also be stored in the metadata store as it is required by solvers to match intermediate nodes, like a model of common feature lines of a group of objects. Furthermore, the framework is able to present intermediate results of the object recognition even if it cannot entirely differentiate all objects, referenced by a *Search Node*. Applications may already benefit from such intermediate results—as presented in our prototype in Section 4.5—to, e.g., show a coarse 3D proxy model presenting the common appearance of the entire model group. An example configuration is depicted in Figure 4. The search node F_1 will execute solver S_A during object-recognition traversal to differentiate models $M_{1..3}$ and the group of models subsumed under F_2. Models $M_{4..6}$ are further distinguished based on the referenced metadata of F_2.

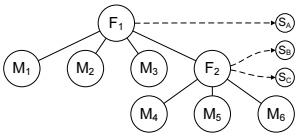

Figure 4: Hierarchical organization of the models. F_1 differentiates models $M_{1..3}$ and the model group of F_2. Models $M_{4..6}$ are distinguished via node F_2.

Context-Switch Node C is a node type that is able to route the traversing of the tree dependent on context attributes. Context-switch nodes can also have a number of child nodes but do not reference any solvers, they only evaluate context data to decide how the traversing continues. This way, alternative sequences for the recognition can be integrated in the mostly static hierarchy. Figure 5 shows a configuration where the context-switch node C_1 decides into which child the traversing descents, based on context information like the current pose of the to-be-identified object. As can be seen in Figure 5 the two subgraphs are simply swapped versions of each other. This way, sequences where the system benefits most—e.g. does not have to request user interaction (Section 4.3)—can dynamically be selected.

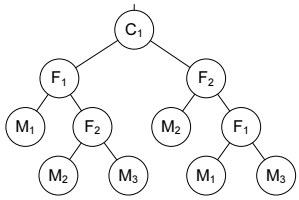

Figure 5: Alternative sequences for the recognition process via a Context Switch C_1.

4.3 Object Recognition by Traversing the Recognition Tree

The object recognition is performed by traversing the recognition tree and, dependent on the node type, processing the results after executing the referenced solvers to select the child node in which to descend. Exemplary setups of recognition trees are illustrated in Figure 6. We integrated multiple solvers that identify objects based on completely different features: 3D geometry, color, patterns, or line features. A more detailed description of the utilized techniques is given in Subsection 4.5. An important aspect of the proposed system is that the knowledge gained about the current and previous iterations of the object recognition, e.g. the pose of the to-be-identified object or the estimated camera position in previous iterations, is gathered and stored as context data. During traversing of the recognition tree, subsequent solvers have access to this information and may consume, correct, or extend it.

If a leaf of the hierarchy is reached during the traversing then a model has unambiguously been identified and the recognition task is finished, returning the identified object and the gathered context information (see Figure 6a, following steps I_1:1, I_1:2, I_2:1). An important advantage of using a model hierarchy is that our system is able to descent the hierarchy as soon as an adequate match is found in the referenced models, as similar models will be summarized using a group in the hierarchy. In contrast, simple approaches would have to linearly check *each* referenced model if it matches to find the best matching which could potentially be the last reference.

Whenever a node can no further distinguish its children and therefore cannot further descend the recognition tree, intermediate recognition results are returned. For that purpose, our system supports a novel interactive refinement of intermediate recognition results by triggering actions that help the system to further differentiate the objects as can be seen in Figures 7, 1a, and 6a, following steps I_1:1, I_1:2, and I_1:3. This feature can be used, e.g., to instruct the user to perform appropriate camera movements and optimize the point of view to capture additional features (see Fig. 6a step I_2:1). User performed adjustments are therefore utilized to compensate for the lack of automatic (e.g. mechanical) installations, which are

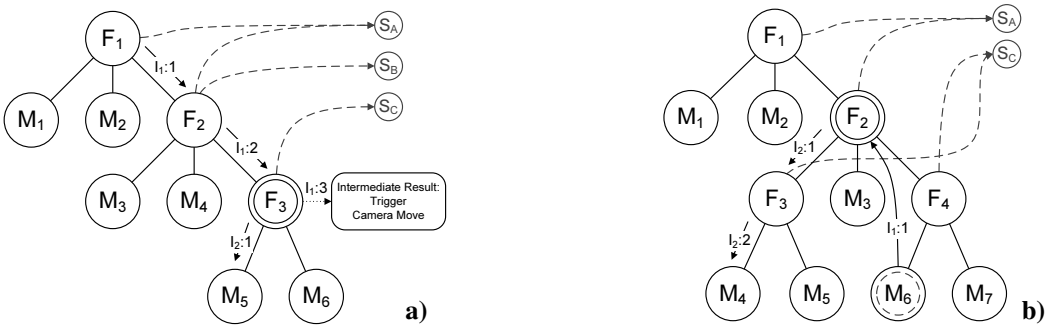

Figure 6: Hierarchical organization of models via multiple search nodes F_x. The recognition performed in a) stops at the intermediate model referenced by F_3. Afterwards a trigger is executed to request a camera movement (I_1:3) to capture further distinguishing features. In b) the previously identified model has changed, therefore the system performs a backtracking (I_1:1) and initiates the next iteration at F_2.

Figure 7: During object recognition the system triggered a request to adjust the lighting condition.

unavailable on handheld devices. Within our prototype we limited these actions to manual camera-movement (Fig. 1a) and light-adjustment (Fig. 7) commands, but other actions—automatic or requests for manual adjustments—like adjusting camera shutter/focus might be triggered.

For achieving a high performance in the recognition phase, the hierarchy helps in two ways: First, the number of models that have to be searched within each search nodes are reduced due to the tree-like structure and second, the temporal coherence—i.e. in most subsequent frames the same object is captured—can effectively be used to skip large parts of the recognition tree. This is achieved by starting the traversing at the node returned in the previous iteration, symbolized as double-framed nodes in Figure 6. If the starting node is not the recognition-tree root the system has to check if the to-be-recognized object is still the same. Furthermore, some context information might got invalid since the previous frame and has to be updated, e.g. due to slight camera movements. Therefore, the system has to ensure that the information required by subsequent nodes are up-to-date by executing the corresponding solvers. This dependency can be determined and stored in a preprocessing step. For the prototypical implementation, we optimized the restart by using a combined solver that checks for a change of the object and estimates its new pose. The system simply checks if the object is still the same by trying to match line features that are referenced by the node where the recognition process continued. The matching is executed quite fast since line features are already known and a good approximation of the camera position is provided as a result of the previous iteration. For the prototype (Section 4.5) the estimation of the camera movement between subsequent iterations was improved using an inertial sensor to measure the acceleration and approximate the new camera position.

If the matching returns a positive result, the newly corrected pose estimation is stored as context information and the traversing continues, thereby keeping all recognition results of previous iterations. This is illustrated by step I_2:1 in Figure 6a. In contrast, if the verification fails the system assumes that the object which has been recognized in the previous iteration has changed and therefore has to check if other objects are present in the captured scene.

Therefore, the system can simply *reset* and start the recognition from the root of the recognition tree. This, however, will result in an inefficient behavior whenever an object cannot be recognized continuously in subsequent iterations: The system will have to descent the entire hierarchy. To overcome this limitation a simple backtracking mechanism is integrated to ensure that the system remains efficient, i.e. restarting the recognition from a previous node on the node path back to the root. This recursive process continues until the recognition is able to descend the hierarchy again or—in the worst case—a restart of the recognition is initiated at the root node of the tree.

During construction time of the recognition tree a link can be stored per node that is followed during backtracking to skip in-between nodes to increase the efficiency as shown in Figure 6b step I_1:1. Afterwards, a following iteration (I_2:1, I_2:2) traverses again to a leaf node.

In the proposed interactive refining, special cases occur that are annoying for users: The system might re-

quest a camera movement to the right to continue the traversing. However, a following search node might request a movement back to the left whereby it possibly would have been satisfied with the camera position at the beginning. In worst cases, users are required to adjust the, e.g., camera view multiple times whereas one adjustment would have been sufficient. Our system overcomes such scenarios by integrating context-switch nodes *C* to select alternative sequences for the recognition. The subgraphs of *C* nodes are simply permuted in their order of execution as illustrated in Figure 5 and selected based on the current context.

4.4 Using external Context Information for Recognition-Technique Selection

During traversing of the hierarchy the system gathers context information required by subsequent solver nodes to perform their task. In addition to this internally generated data, *external* context information also helps to control and improve the recognition process.

Each search node of the recognition tree that summarizes multiple nodes, references at least one technique to search through its children. The selection of the solver to differentiate features of referenced models is done in a preprocessing step during the recognition-tree construction. Therefore, an algorithm is chosen which is assumed to deliver best results on average in terms of reliability, robustness, or performance. However, selecting the most adequate technique to distinguish groups of similar objects during the setup of the recognition tree in a pre-processing step is not always possible: Changes in the environmental context, application states, sensor quality, etc. might occur during runtime and therewith invalidate the selection of recognition algorithms based on these attributes.

In order to overcome this limitation, the proposed system allows to assign multiple alternative recognition techniques per node that are dynamically selected, based on *internal* or *external* context information. The external context information is provided via Nexus [4, 5], as mentioned in Section 2.

4.5 Prototype System and Results

The presented recognition system has been specifically designed to share the context and metadata store with other application parts. Therefore, Augmented Reality visualizations can easily access the position and orientation information using the context store. The prototypical implementation of a mobile interactive assistant system to help users to identify and augment objects makes use of this concept. A key concept of the system is that very similar objects are differentiated using the proposed refining technique for controlled user intervention. An exemplary target application for the prototype is an information system for customers and consultants who are interested in HIFI appliances. These

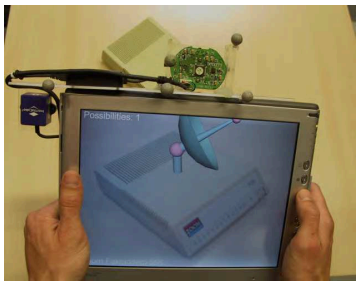

Figure 8: The prototype hardware in use. A standard TabletPC was equipped with a webcam (top middle) and an inertial sensor (top left).

items have many similar aspects which cannot easily be differentiated. Augmented Reality provides an intuitive way to present different instances of an object—probably not yet available—like extra attachments, different colors, or even custom case modifications. For evaluation of the proposed concept, a prototype as seen in Figure 8 was built.

4.5.1 Implementation

Multiple solvers have been implemented to support the recognition of various feature types that might be useful to differentiate similar objects. The most advanced solver is—in addition to identification—also able to estimate the pose of objects. It is implemented using Computer Vision methods: First, feature lines and three-junctions are searched in the captured image. These are linked to the model's geometry description—stored in the metadata store—to generate hypotheses of possible models and their orientation [7]. The second part of the solver is based on a technique proposed by Lowe [11] and is used to check generated hypotheses and further improve the pose-estimation accuracy by minimizing the matching error. We refer to this solver as *edge-model solver* S_E.

A cut-down version of the *edge-model solver* is integrated to differentiate similar models, where one model has additional feature lines. A prerequisite for this solver is a pose estimation, calculated by any previous *search node* in the recognition tree. The solver can then project the additional line features, stored for one specific object of a group, and search for matching edges in the captured image, therefore the solver is termed as *diff-edge solver* S_D. Distinguishing features, e.g. additional lines, might be visible only from a specific point(s) of view therefore a differentiation of the objects based on a captured image from an arbitrary view is not always possible and an intermediate result might be returned. As our approach is especially targeted for mobile clients, where often only a single camera with a fixed viewing direction is available, we cannot expect that other sensors might capture a distinguishing feature. The novel approach of our proposed interactive refining technique to solve such un-

Figure 9: Rendering of augmented camera images at different steps of the recognition process. Detected and vectorized edges of the camera images are shown in a). A hypothesis (blue) generated by the approach of Dhome [7] and model's feature lines are seen in b). A finally checked and improved solution—based on Lowe [11]—is displayed in c). The red line segment could not be matched in the camera image.

determined cases is that solvers can report intermediate results with hints/triggers that state which changes have to be made in order to continue the recognition. This includes change requests for, e.g., camera movements or lighting conditions that are displayed to the users as can be seen in Figure 1a and Figure 7.

In contrast to line features, the *color-spot solver* S_C is able to check the color value at pre-defined locations on the object surface. It simply projects pre-defined color-probe locations using the previously estimated object pose to the image space and examines the pixel color in the captured image. Therefore, objects with (partially) different colors can efficiently be identified even if colored features are only visible at specific points.

Similar real-world objects are often labeled or marked in order to express their differences. Good examples are electronic appliances like HIFI components which might only be different in their insides and probably their serial numbers. Our prototype utilizes a recognition approach to compare previously acquired reference images to the captured image information to support an identification based on patterns. This *pattern solver* S_P benefits of a previously calculated pose estimation in two ways: It is able to determine if the pattern is entirely visible in the camera image, i.e. it is not hidden or occluded, and the perspective distortion is a priori known due to the previously calculated object orientation and the pattern location in model coordinates, stored as metadata.

Many additional techniques can easily be integrated to detect more complicated features like, e.g., a solver to recognize curved surfaces. But also identification components like optical character recognition or barcode scanners can be applied.

The proposed recognition system is applicable for various applications, for the prototype we chose to implement an Augmented Reality rendering module to present real-world aligned virtual information. It is used to show information about the traversal of the recognition tree and object information. If the system is unable to totally identify an object, only its category is displayed. We further make use of AR rendering to track and evaluate the recognition process and its accuracy. The precision of the object's pose estimation can easily be seen via an augmentation of the camera image with superimposed 3D geometry model (see Fig. 9).

For the proposed scenario, presentations based on AR further benefit from the possibility to show different virtual instances of objects, similar to the illustrations in Figures 1b,c. These renderings depict two different augmentations, which might represent future configurations or not-in-stock items.

As mentioned in Section 4.3, intermediate nodes may trigger actions to improve the recognition. We implemented triggers to initiate camera-movement requests which display messages to guide users interactively how the camera should be positioned in order to achieve better recognition results, as seen in Fig. 1a, and Fig. 7.

4.5.2 Results

For the evaluation of the prototype implementation a data set with five computer-appliance objects were used. Two objects are identical except for a red stripe on one object. Therefore, these objects can only be distinguished if the part where the red stripe is located on is within the camera view. A direct comparison to existing systems cannot be given, since the proposed setup is rarely examined, often a specialized algorithm is utilized where the objects used to evaluate the system fit to the proposed algorithm. The theoretical complexity in terms of executions of *solver-model* pairs in the hierarchical implementation is optimally $O(log(n))$. For a simple linear search method the complexity is $O(n)$ which corresponds to the worst case of our approach. In practical setups the system therefore achieves a performance in-between both extremes. However, these numbers strongly depend on the number, type, and quality of the objects and the structure used for the recognition tree. The implemented recognition algorithms and the selected techniques also have a great influence to the overall performance.

Measurements in Table 1 present the performance for an Intel Core2 Quad Q6600@2.4GHz CPU (only a sin-

gle core was utilized). The video stream of the camera was simulated with a static 320×240 resolution image in order to achieve comparable results.

	non-coherent	*coherent*
Preparation	21.3	21.3
hypOther	163.1	-
hypValid	5.8	-
hypCheck	0.8	0.8
Rendering	1.4	1.4
Overall	192.4 ms	23.5 ms

Table 1: (Re)start performance of an object-recognition process utilizing temporal coherence.

The first phase of our approach is equal for both cases: Undistortion of the camera image, generation of a monochromatic image, execution of a Sobel/Canny edge detection, and the merging of collinear line fragments which is summarized as *Preparation*. In the *non-coherent* case, where no object registration and pose estimation is available from previous frames, a large number of hypotheses have to be evaluated. The timing values of *hypOther* refer to hypotheses that are evaluated with 3D object models which do not correspond to the captured camera image. The *hypValid* measurements refer to hypotheses that are evaluated with a matching 3D object model. In the *coherent* case we already have a valid object pose from a previous iteration which is already accurate, as the camera is fixed during the evaluation. Timings of *hypCheck* refer to the Lowe based approach for checking and improving the pose-estimation hypothesis. Timings for display of the captured camera image and optional augmentations are summarized in *Rendering*.

The measurements show the benefit of utilizing temporal coherency as the most time-consuming part of the algorithm is efficiently skipped. With the previously mentioned performance improvement due to the utilized hierarchy the system is especially suited for mobile, interactive applications.

5 CONCLUSION

We have presented an approach to combine arbitrary recognition techniques within a single object-recognition and pose-estimation system. A hierarchical structure is utilized to achieve highly efficient and robust object recognition. The recognition process is performed by traversing the hierarchy whereby referenced solvers are executed. The returned result is either the unambiguous object identification or an intermediate result. To improve intermediate results the system can trigger actions—e.g. relocation of the camera—in order to capture additional important features for the recognition process. These triggers are used to implement an interactive process to refine recognition results by providing hints to guide users how to improve the recognition, thereby providing the possibility to adjust an otherwise static setup to sensors.

In future we will concentrate on automatic construction of a balanced model hierarchy.

REFERENCES

[1] F. Ababsa and M. Mallem. Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems. In *VRCAI '04: Proceedings of the ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 431–435. ACM, 2004.

[2] D. Beier, R. Billert, B. Brüderlin, D. Stichling, and B. Kleinjohann. Marker-less vision based tracking for mobile augmented reality. In *2nd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, page 258, 2003.

[3] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. A comparison of probabilistic, possibilistic and evidence theoretic fusion schemes for active object recognition. In *Computing*, volume 62, pages 293–319, 1999.

[4] Collaborative Research Centre (SFB627). Nexus: Spatial world models for mobile context-aware applications. http://www.nexus.uni-stuttgart.de, 2008.

[5] P. Coschurba, U. Kubach, and A. Leonhardi. Research issues in developing a platform for spatial-aware applications. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 153–158. ACM Press, 2000.

[6] F. Deinzer, J. Denzler, and H. Niemann. Viewpoint selection - a classifier independen learning approach. In *IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 209–213, 2000.

[7] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives. Determination of the attitude of 3-d object from a single perspective view. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 11, pages 1265–1278, 1989.

[8] M. Grabner, H. Grabner, and H. Bischof. Fast visual object identification and categorization. In *Proceedings of NIPS Workshop in Interclass Transfer 2005*, pages 1–8, 2005.

[9] D. J. Kang, J.-E. Ha, and I.-S. Kweon. Fast object recognition using dynamic programming from combination of salient line groups. *Pattern Recognition*, 36(1):79–90, 2003.

[10] D. J. Kang, J.-E. Ha, and I.-S. Kweon. 3-D Pose Estimation Algorithm for Model Based Vision. In *6th Asian Conference on Computer Vision (ACCV)*, pages 115–119, 2004.

[11] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. In *Artificial Intelligence*, volume 31, pages 355–395. Elsevier, 1987.

[12] R. Mehrotra, W.I. Grosky, and F.K. Kung. Decision-tree based two-dimensional object recognition. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics.*, pages 1380–1383, 1988.

[13] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520–527, 2007.

[14] M. Reinhold, F. Deinzer, J. Denzler, D. Paulus, and J. Pösl. Active appearance-based object recognition using viewpoint selection. In *VMV*, pages 105–112, 2000.

[15] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR04)*, pages 48–57, 2004.

[16] P. Viola and M. J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[17] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented realityregistration. In *Proceedings of IEEE Virtual Reality*, pages 260–267, 1999.