Augmented Things: Enhancing AR Applications leveraging the Internet of Things and Universal 3D Object Tracking

Jason Rambach*[1], Alain Pagani†[1] and Didier Stricker‡[1,2]

[1]German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
[2]TU Kaiserslautern, Kaiserslautern, Germany

ABSTRACT

With Augmented Reality (AR) reaching its technological maturity, there is a constantly increasing request for AR applications that address a broadened user base. Current AR applications still tend to be spatially and temporally confined either by applying to a single use case or by their tracking requirements. At the same time, with the rise of the Internet of Things (IoT), more and more everyday objects are fitted with wireless connection interfaces. In this work, we present the concept of Augmented Things, in which objects carry all the necessary tracking and augmentation information required for AR applications. This allows a user to connect to them, load this information on his personal device and have augmentations such as maintenance instructions or product origin and usage displayed. For this, we also present and evaluate a universally robust 3*D* object tracking framework based on high quality 3*D* scans of the objects.

Index Terms: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities—; I.4.8 [Scene Analysis]: tracking—

1 INTRODUCTION

Augmented Reality is an emerging technology with numerous applications in a variety of fields such as industrial construction and maintanance, medicine and rehabilitation, entertainment, education and many others [4].

Accurate camera pose tracking is one of the key enabling technologies for AR using handheld or wearable devices. Precise tracking of the camera's six degree of freedom pose (6DoF) corresponding to its orientation and position in every video frame allows for realistic placement of 3*D* augmentations and integration in the real world [10, 16]. Existing tracking approaches rely either on uncovering predifined targets such as markers or natural features (e.g. edges and lines) of objects [6, 7] or employ Simultaneous Localization and Mapping (SLAM) methods to uncover the structure of the environment. In any case, an AR application needs to contain its own tracking solution, either self-implemented or as an integration of a commercially available tracking framework.

In parallel, with the advance of Internet of Things (IoT), originating from the reduced cost and size of sensors and miniature hardware components, more and more devices and everyday objects are embedded with microcomputers and wireless network connectivity [12]. This leads to the idea of Augmented Things presented in this paper. We propose to have objects that carry all the information that is required for them to be used as tracking targets and the augmentation models that should be displayed for this model. Thus, a user that wishes to display AR augmentations on that object (e.g. for usage instructions, maintenance or simply entertainment) can use a universal object tracking application from his personal device that will connect to the object, receive the object AR package (tracking information and augmentations) and directly track the object and display augmentations. This leads to a separation of tracking and content creation for AR, that greatly simplifies the creation of AR applications.

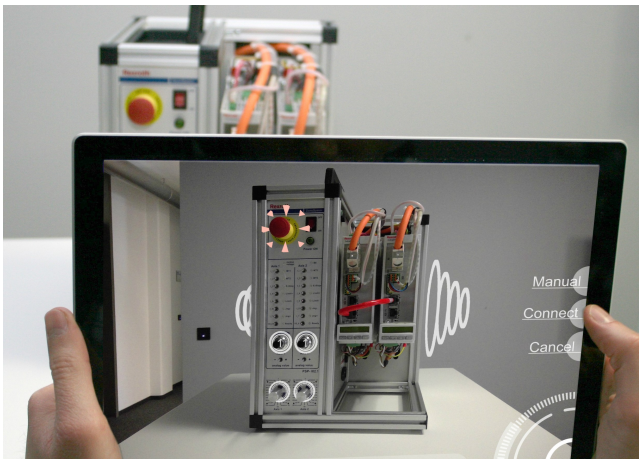

Figure 1: In Augmented Things connected objects carry and share their AR information

The main contributions of this work are:

- The presented concept of Augmented Things and the application architecture for its realization.
- A robust universal 3*D* object tracking framework suited for the Augmented Things architecture utilizing off-screen rendering of the object 3*D* model and a high degree of parallelization.
- A fast and robust mulithreaded seamless initialization and reinitialization method for tracking.
- The use of the pencil edge enhancing filter to improve KLT tracking.

The rest of this paper is structured as follows: In the next section we discuss related work on object tracking and AR applications (Section 2). In Section 3 the Augmented Things concept is discussed further and an application architecture is proposed. In Section 4 we present in detail our object tracking framework implementation. Subsequently, in Section 5 we present an experimental evaluation of our object tracking framework in terms of tracking accuracy and runtime performance. Finally, some concluding remarks are given.

*Jason_Raphael.Rambach@dfki.de
†Alain.Pagani@dfki.de
‡Didier.Stricker@dfki.de

2 RELATED WORK

A large amount of work has been conducted over the years on the topic of camera pose tracking for augmented reality [10]. Approaches using markers or other fiducials provide robustness but disrupt the natural scene and are sensitive to occlusions of the tracking target and spatially confined to the area where the markers are visible. An alternative to that are Simultaneous Localization and Mapping (SLAM) approaches that extract features 3*D* and map their environment during tracking using structure from motion methods [7, 13]. Some of these systems have shown very good results, however they lack absolute orientation and tend to be computationally demanding.

Using the object that should be augmented in an AR application as a tracking target frees from the requirement for fiducials and directly sets the correct coordinate frame for augmentations on and around the object. 3*D* CAD line models of objects were used in [6] and [24] to perform tracking by minimizing the error between the lines of the observed object and its CAD line model. In order to provide more robustness it was proposed to fuse the edge-based tracking with texture features in [23]. Textured 3*D* rendering of the tracked object model is used to extract edges from depth and texture discontinuities in [17]. Some of these approaches however appear to be computationally very intensive as low frame-rates of $10-15$ fps were achieved. These aforementioned methods using tracking of edges from object models also show a deteriorating performance in cluttered scenes. Direct methods matching the image intensity of the object model to the scene in an iterative optimization have recently emerged [20, 21]. Vuforia [3] is offering an object tracking framework based on CAD models which are not available for all types of objects or scans made by the user himself by placing the object on some marker pattern. The latter requires some expertise of the user and does not recover the full structure of the object. To our knowledge there are no approaches using textured 3*D* rendering to perform pose tracking in a Frame-to-Frame (F2F) and Rendered-to-Frame (R2F) feature matching manner in real time as is presented in our work.

Concerning the Augmented Things concept, some authors recognize the impact that the Internet of Things can have to augmented reality mainly in terms of wearable AR devices [11] or AR as a cloud service to reduce latency for computationally intensive operations [5]. ThingWorx [2] is developing an IoT platform of objects that carry a specific marker and can be connected to in order to display among other things AR content that uses the Vuforia tracking framework based on CAD models. Reality Editor [1] allows placing visual markers on objects and using them to track and create AR information to add a digital layer to them. However, to the best of our knowledge, the notion of connected objects that carry and share their tracking and augmentation information has not been explored yet.

3 APPLICATION ARCHITECTURE

The main concept of the Augmented Things framework is having connected objects that carry all the information necessary to be tracked and are able to share this information upon request. Coupled with a universal 3*D* object tracking application that utilizes this information to display useful or entertaining augmentations to the end user/product consumer, it can provide a platform for shipping products with customized AR content without the need for defining a tracking and rendering infrastructure alongside. To give a comprehensive overview of the proposed application architecture we split the description of the components and requirements to the object producer side and the object user or consumer side (Figure 2).

On Figure 3 examples of objects that could be integrated into the augmented things application framework are displayed. These objects can be common household devices (e.g. sewing machine)

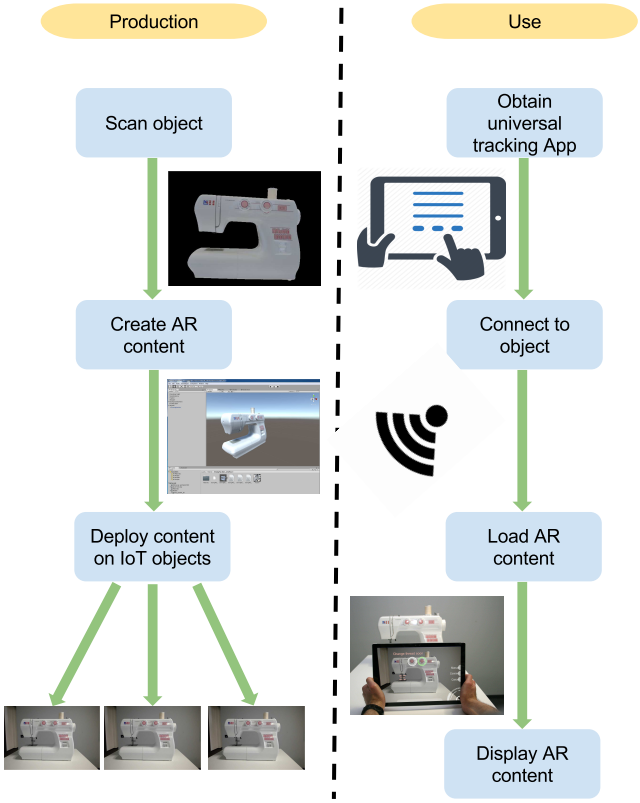

Figure 2: The augmented things architecture separated in object production and use

or complex industrial devices where augmentation regarding their correct usage and maintenance can be overlayed to assist the user. The idea can also apply to consumable products (e.g. cereal box) that can display for example nutrition facts, expiration date, or origin. Objects presented in an exhibition (e.g. chinese lion statue) can also be tracked to display historical information or other interactive information in AR.

3.1 Producer side

The producer of an object with a wireless connection interface who wishes to offer augmented reality content can use a predefined 3*D* scanning technology (e.g. such as the one from 3Digify used in this work [14, 8]) to create a high-quality 3*D* model of the product. Additionally, a graphics/rendering engine such as Unity can be used for the creation of the specific augmentations that should be displayed with this object. The augmentations could be maintenance or correct usage instructions for a household device, information about the production conditions for some consumable product, or even general information and multimedia support for an exhibition object. The augmentations can be placed in the same coordinate system as the scanned 3*D* model without additional concern about the tracking of the object. The 3*D* model, the tracking information, and the augmentations can then be placed as a package on the memory of the connected object to be shared upon request. The estimated memory requirements of about 10 to 100 MB should not pose a storage capacity issue.

3.2 User/consumer side

On the user side the only requirement is having the universal 3*D* tracker application on their personal device. This application consists of a tracker (described in Section 4.3.2) wrapped in a Unity

Figure 3: Sample candidate objects for the Augmented Things application. These objects were also used for testing and evaluating our tracking approach

interface that is responsible for the rendering of augmentations using the pose obtained from tracking. A user who is interested in displaying augmentations over some object can use the application to scan the space for objects that can be augmented and connect to them via bluetooth or wireless network. Upon connection a copy of the package of tracking information (object $3D$ scan and augmentations models) placed by the object producer can be transmitted to the user application and the tracking and displaying of augmentations can on the object can begin. The tracking information can be stored by the application for as long as the user wishes to.

4 OBJECT TRACKING

In this Section we present our object tracking approach, implemented for the realization of the Augmented Things idea. Thus, the main requirements are the ability to accurately track arbitrary everyday objects in a computationally efficient manner in order to provide a good user experience. The main components of the presented tracking framework are the object registration scheme based on the generation of different poses from the scans of the object, a KLT feature tracker modified for robustness to illumination changes, and a quick initialization scheme based on ORB feature matching.

4.1 Problem Formulation

The addressed problem of $6DoF$ camera pose tracking consists of estimating the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3x3}$ and translation vector $\boldsymbol{t} \in \mathbb{R}^3$ that relates the camera coordinate frame to the tracked object coordinate frame. Using a homogeneous representation of $3D$ points $\boldsymbol{P} \in \mathbb{R}^4$ in the object coordinate system and a homogeneous representation of $2D$ points $\boldsymbol{p} \in \mathbb{R}^3$ in the camera image coordinate system, the camera pose estimation problem requires an estimation of the camera pose $[\boldsymbol{R}|\boldsymbol{t}]$ such that the mapping

$$\boldsymbol{p} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{P}, \tag{1}$$

best fits a set of known $3D$ to $2D$ correspondences $C = \{\boldsymbol{P}_i \leftrightarrow \boldsymbol{p}_i\}$, $\boldsymbol{K} \in \mathbb{R}^{3x3}$ representing the camera instrinsics matrix.

4.2 Object Registration

During the registration of an object for tracking a set of N anchor points with known $3D$ positions in the object coordinate system and strong edge characteristics is acquired. The idea is to select $3D$ points on the surface of the object that have a high probability to be detected by a feature detector. To this aim, we use synthetic rendering and statistical analysis of the frequency of detection of $3D$ points. This is done by first rendering a highly accurate $3D$ scan of the object from a large number of random rotations of the camera in all 3 axes (Figure 4). In every rendered pose $[\boldsymbol{R}|\boldsymbol{t}]_i$ of the object strong corner features are detected using a standard feature detector, and their $3D$ positions $\boldsymbol{P}$ are determined by using the depth buffer of the renderer as follows. If $\boldsymbol{p} = [x, y, 1]$ is the $2D$ homogeneous representation of the point (pixel coordinates of the corner feature center) and z is the depth obtained from the z-buffer of the renderer, then the 3D coordinates $\boldsymbol{P}$ of this point are defined as:

$$\boldsymbol{P} = \boldsymbol{R}_i^T \left((\boldsymbol{K}^{-1}\boldsymbol{p})z - \boldsymbol{t}_i \right). \tag{2}$$

For each random pose, all the found $3D$ points are registered in a $3D$ accumulator with millimeter accuracy. After the keypoints have been collected for a large number of poses (usually in the order of magnitude of 10.000 different poses), a non-maximum suppression algorithm is run over the accumulator to keep only the local maximum number of occurrences. The remaining cells are then sorted according to the decreasing number of occurrences. The first N $3D$ positions are kept as features with the statistically highest probability of being detected by the feature detector.

Apart from the registered $3D$ points, ORB[18] feature descriptors and their $3D$ coordinates are extracted from a small subset N_{orb} of the rendered images and are stored to be used for initialization of the tracking as described in Section 4.3.1.

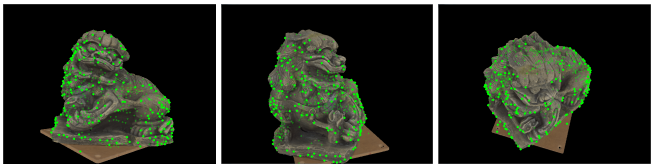

Figure 4: Rendering of an object $3D$ model from different poses with marked detected anchor points

4.3 Tracking Algorithm

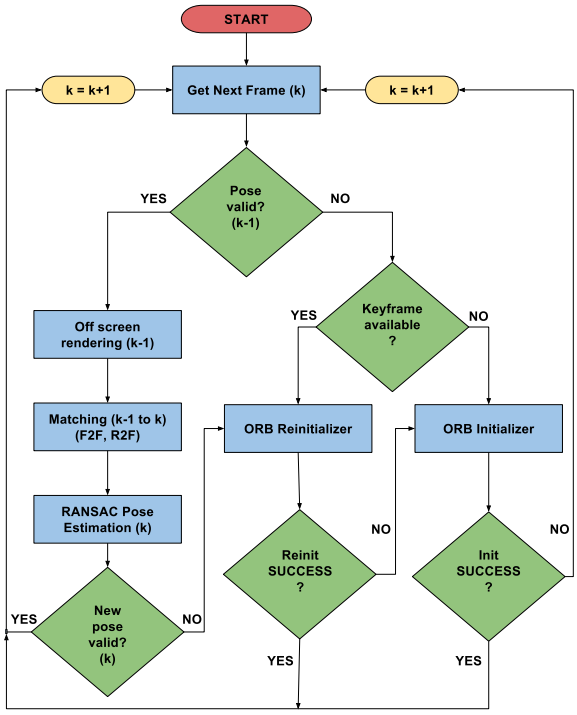

Figure 5: Flow diagram of the proposed tracking framework

The outline of the proposed pose tracking framework is given in Figure 5. Whenever a new video frame k is acquired, if the tracking was successful on the previous frame $k-1$, an off-screen rendering of the $3D$ model using the pose from the previous frame $[\boldsymbol{R}|\boldsymbol{t}]_{k-1}$ is done and optical flow tracking is performed from the previous frame to the current (F2F) and from the rendered model of the object to the current frame (R2F). If enough matches are obtained, the current pose $[\boldsymbol{R}|\boldsymbol{t}]_k$ is estimated within a RANSAC framework for outlier rejection (Section 4.3.2).

Whenever the tracking is lost, the initialization modules are activated (Section 4.3.1). A quick reinitialization scheme uses ORB feature [18] matching between the current frame and a stored keyframe. If this immediate reinitialization fails or a keyframe is not available, the ORB Initializer matches features from the current frame to a number of frames collected during the registration of the object.

4.3.1 Initializer and Reinitializer

As the pose tracking from frame to frame can fail in some cases when the image quality is compromised, e.g. due to motion blur, occlusions or illumination changes, it is crucial for the tracking system to be able to regain the correct pose efficiently. In this work we propose a multi-threaded scheme based on ORB features that allows the pose tracker to initialize and reinitialize seamlessly and robustly.

The ORB initializer operates by matching the current video frame to the ORB feature descriptors of rendered object poses stored during registration (Section 4.2). Parallelization is used to split the matching of descriptors sets from the N_{orb} images among all available threads. This allows to match to a large number of descriptor sets without producing a large delay. A ratio test[9] is applied to the matches from every descriptor set and the set having the most good matches is used for pose estimation. The pose estimation is done by solving the perspective PnP problem using the matches that passed the ratio test in a RANSAC scheme for outlier rejection[15].

The ORB reinitializer is designed to rapidly regain the current pose whenever tracking fails. It does matching of ORB features from the current image to a reinitialization keyframe. During tracking the reinitialization keyframe is constantly renewed by a low-frequency thread. Only frames of good tracking quality with a high number of tracking matches and low reprojection errors are used as keyframes. If a failure of the tracking is detected the reinitialization module is immediately activated, attempting to do ORB matching from the keyframe to the current image to regain the camera pose. The reinitializer is highly effective because it matches between images that are very similar, especially when the keyframe gets renewed at a high rate. If the reinitializer fails to produce a correct pose, the ORB initializer module is activated.

4.3.2 F2F and R2F KLT Tracking

The tracking module is given a camera pose $[\boldsymbol{R}|\boldsymbol{t}]_{k-1}$ from the previous video frame $\mathscr{I}_{k-1}$ as input and attempts to correctly estimate and output the correct pose $[\boldsymbol{R}|\boldsymbol{t}]_k$ for the current video frame $\mathscr{I}_k$. Subsequently, the following steps are performed by the tracking algorithm:

- Perform an off-screen rendering $\mathscr{R}_{k-1}$ of the tracked object $3D$ model using the pose $[\boldsymbol{R}|\boldsymbol{t}]_{k-1}$.
- **If** the previous pose was obtained through initialization, project the object's $3D$ anchor points $\boldsymbol{P}_i$, $i = 1,\ldots,N$ on the rendered frame $\mathscr{R}_{k-1}$ using Equation 1 to obtain their $2D$ image coordinates $\boldsymbol{p}_i$. Before projection a visibility test is performed on the $3D$ points, in order to exclude those that are not visible from the current pose because of object self-occlusion. Square small image patches centered on the $2D$ points are extracted from the rendered image to be used as KLT tracking features [22].
- **If** the previous pose was obtained through initialization, do optical flow matching [19] (Rendered-to-Frame - R2F) of the KLT features between the rendered frame $\mathscr{R}_{k-1}$ and the current frame $\mathscr{I}_k$.
- **Else** first do optical flow KLT matching between previous $\mathscr{I}_{k-1}$ and current frame $\mathscr{I}_k$ (Frame-to-Frame) and then refine the matches by R2F matching using the F2F output as initial positions.
- After the matching phase, if enough matches are present, solve a perspective PnP problem within a RANSAC framework using the recovered $2D$ - $3D$ correspondences to estimate the current camera pose $[\boldsymbol{R}|\boldsymbol{t}]_k$. If RANSAC fails in case of too few feature matches or large reprojection error the tracking is interrupted and the initialization modules are activated (Section 4.3.1).

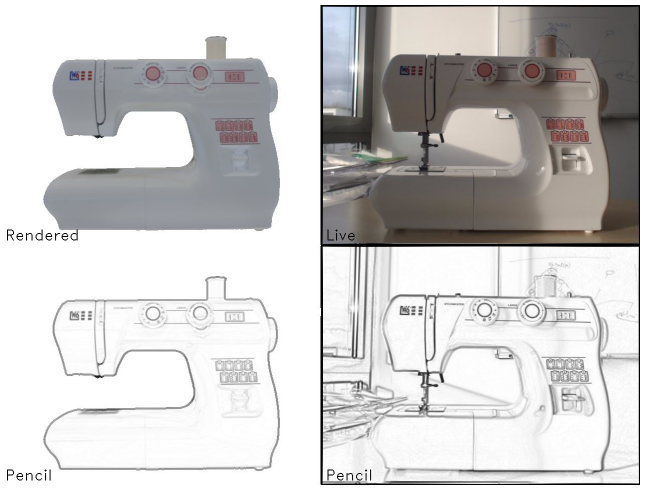

Figure 6: Result of pencil filter applied on rendered image (left) and live image in challenging illumination condition with shadows and reflections (right)

The KLT optical flow matching is constrained to do matching by local search around each feature's position to improve the matching speed since a feature is expected to have a displacement of only a few pixels from frame to frame especially in a high camera frame rate (>25fps). For algorithm speed-up purposes both the feature matching and the RANSAC iterations are parallelized.

4.3.3 Pencil filter

The KLT tracking for objects with sparse texture is supported by applying the pencil filter to the images before matching features. The pencil filter is commonly used as an artistic effect on images, however we found that its edge enhancing properties are very beneficial for tracking. the filter consists of a dilation of the grayscale image with an ellipse followed by local normalization. As can be seen in Figure 6 the filter allows the recovering of edges under difficult illumination conditions. The rendered model and its corresponding pencil image are depicted on the left, while on the right side the live image and its pencil image are shown. The two pencil images are highly similar and this significantly improves the R2F KLT feature matching. Furthermore the edges of the pencil filter are very consistent throughout an image sequence, compared to e.g. Canny edge detection.

5 EVALUATION

In this section we present our results from the evaluation of our object tracking framework described in Section 4. We evaluated the

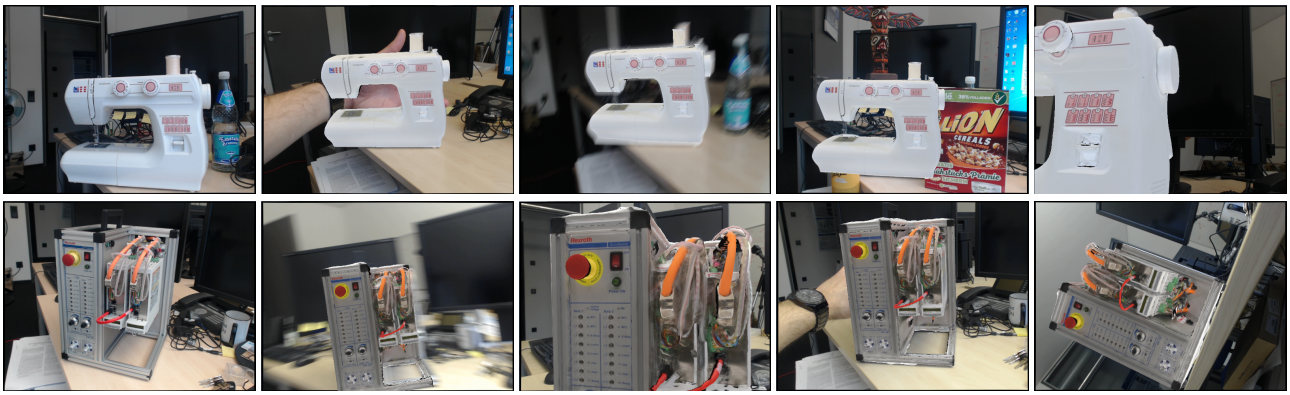

Figure 7: Tracking results on a sparsely textured sewing machine and an industrial device. The object itself is shown on the first image of each set and on the rest the object with its $3D$ model rendered over it. We demonstrate robust tracking from different angles and distances from the objects, during occlusions or partial view of the object, motion blur, low illumination conditions and cluttered background.

two most important traits expected of a tracker, namely the tracking accuracy (Section 5.1) and robustness and the runtime performance (Section 5.2).

5.1 Tracking accuracy evaluation

Object	Avg. Reprojection error (pixels)
Cereal Box	0.9799
Chinese Lion Statue	1.3651
Sewing Machine	1.7256
Industrial Machine	1.2292

Table 1: Average tracking reprojection error on all tested objects (Figure 3)

A qualitative evaluation of our tracking framework is given in Figure 7. For this evaluation we selected two of our most challenging objects, a sewing machine with very sparse texture and an industrial machine whose reconstruction had several flaws and was less precise than the our other tested objects. In the presented images taken during tracking time we rendered the $3D$ model of the objects over the real objects using the estimated pose. It is clear that there is almost perfect overlap between the real object and the rendering, which is a strong indication of the accuracy of our tracking. Furthermore, we show several challenging situations where our tracking was successful, namely occlusion of the target object by the user's hand or other objects, partial view of the target object, low illumination and motion blur due to fast movement of the camera. The tracker's success in these demonstrated cases is mainly due to the robustness of our modified KLT tracking, since the pencil filter applied before matching is very successful at retrieving edges even in low contrast images (Section 4.3.2). However, even in cases where the tracking inevitably fails, the ORB reinitializer 4.3.1 directly operates on the same frame and efficiently recovers the pose in a seamless manner for the observer.

In Table 1 we provide measurements of the average reprojection error during tracking for all 4 objects that were used for testing. The measurements were made on challenging sequences with fast camera translational and rotational motion. The accuracy of our tracking framework is confirmed by the low reprojection error (< 2 pixels) achieved throughout the sequences for all objects. For the more sparsely textured objects and for the objects whose $3D$ model is less accurate the error is higher than for the others but still low enough to provide stable tracking.

5.2 System Runtime evaluation

We evaluate the runtime performance of the different components of our tracking framework on two devices. Primarily on a desktop PC with an Intel Xeon processor at 3.07 GHz and 12 GB RAM in order to make fairer comparison to other tracking frameworks and secondly on a less computationally powerful Microsoft Surface Pro 4 with an Intel Core i5 processor at 2.4 GHz and 8 GB RAM which is a good example of a device often used for handheld device augmented reality applications. A standard image resolution of 640×480 was used in the experiments.

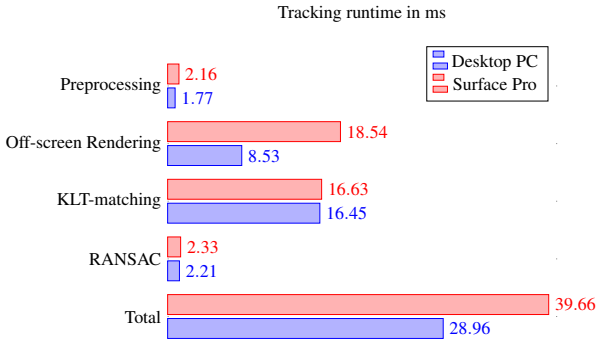

Figure 8: Runtime measurements in ms. per frame of all stages of the tracking algorithm from Section 4 for a desktop and a handheld device

	Desktop PC (ms)	Surface Pro (ms)
Re-init	18.5	33.94
Init $N_{orb} = 8$	31.5	39.97
Init $N_{orb} = 16$	44.0	55.11
Init $N_{orb} = 32$	73.0	89.1
Init $N_{orb} = 64$	133.0	155.2

Table 2: Runtime measurements in ms of the initialization and re-initialization scheme

In Figure 8 we present the average required time in ms to perform the computations of the main parts of the tracking algorithm corresponding to Figure 5. Preprocessing corresponds to receiving and undistorting the captured frame, Off-screen rendering is the rendering of the model of the object with the previous frame pose, and KLT matching and RANSAC is the core of the tracking as explained beforehand. We observe that the KLT matching is one of the most costly operations. For the Surface Pro, we see that all results are comparable to the Desktop apart from the Off-screen rendering which requires more than double the time, due to the graphics card which is the bottleneck of the system in this case. From the total time required for the tracking we compute that frame rates of 34

and 25 fps can be achieved on the desktop PC and the Surface Pro respectively. These high frame rates leave a good amount of time for rendering high quality 3*D* augmentations on the user side.

One of the main contributions of our tracking framework are the fast reinitialization and initialization schemes (Section 4.3.1). For a good experience with the system it is of high importance that these initialization schemes are computationally efficient. In Table 2 we present runtime measurements of the initialization on the two devices used in the experiments. The ORB reinitialization scheme is very fast in both devices since it does matching only between two images which are very similar because of the high rate of refreshment of the saved reinitialization keyframe. Because of the image similarity between keyframe and current image, the reinitialization module is also very effective, managing to compute a pose from the matches in more than 95% of the attempts. The initialization time required depends heavily on the number of candidate poses N_{orb} used for matching and on the degree of parellelization that can be achieved by the device processor. There is a clear trade-off between initialization success rate and speed of initialization. We have found that N_{orb} values of 16 or 32 ensure high success rates of initialization while not requiring too much time.

5.3 Discussion

The main limitation of our tracking approach is that there is a group of objects (e.g. highly reflective, completely textureless) that are not easily scanned with existing technologies. Furthermore, these types of objects with no texture would not provide enough features to be tracked. However, using the object texture for tracking instead of lines or direct optmization methods creates a robust system that is not affected by ambiguities due to the object shape and is resilient to illumination changes due to the use of the pencil filter. Furthermore, the absence of computationally expensive optimization processes allows our tracking system to operate at high frame rates. Finally, compared to AR platforms such as ThingWorx [2], the use of object 3*D* scans appeals to a wide selection of objects for which a CAD model is not available. Our tracking system showed good performance on a large set of different objects in a variety of different environments verifying its real world applicability.

6 Conclusion

In this work we presented the concept of Augmented Things for AR applications and an application architecture for Augmented Things was covering all stages from object production to usage. A universal object tracking framework that targets computational efficiency and user experience was presented as a key enabling technology for Augmented Things. In the evaluation of the system we show that our tracking framework is able to robustly track a variety of sparsely textured objects under challenging image conditions (motion blur, illumination variations, occlusion) at a high frame-rate even on a portable handheld device that is a realistic candidate for AR applications. Future work includes adding inertial sensor measurements to further improve the tracking under fast camera motion. For the same purpose, SLAM approaches can be applied to reconstruct part of the object's environment in order to also use this additional information for tracking.

Acknowledgements

This work has been partially funded by the Federal Ministry of Education and Research of the Federal Republic of Germany as part of the research projects PROWILAN and BeGreifen (Grant numbers 16KIS0243K and 16SV7525K) and the European project Eyes of Things under contract number GA643924. The authors would like to thank Torben Fetzer and Carol Naranjo for the scanning of objects and Ruben Reiser, Oliver Stock and Narek Minaskan for the design of AR content.

References

[1] http://fluid.media.mit.edu/realityeditor.

[2] https://www.thingworx.com/.

[3] https://www.vuforia.com/.

[4] W. Barfield. *Fundamentals of wearable computers and augmented reality*. CRC Press, 2015.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[6] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):932–946, 2002.

[7] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[8] J. Köhler. *Enhanced usability and applicability for simultaneous geometry and reflectance acquisition*. 2016.

[9] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[10] E. Marchand, H. Uchiyama, and F. Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 2015.

[11] F. Mattern and C. Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010.

[12] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[13] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[14] T. Nöll. *Efficient Representation of Captured Geometry and Reflectance*. 2015.

[15] T. Nöll, A. Pagani, and D. Stricker. Markerless camera pose estimation-an overview. In *OASIcs-OpenAccess Series in Informatics*, volume 19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[16] A. Pagani. *Reality Models for efficient registration in Augmented Reality*. Verlag Dr. Hut, 2014.

[17] A. Petit, E. Marchand, and K. Kanani. Tracking complex targets for space rendezvous and debris removal applications. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4483–4488. IEEE, 2012.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[19] T. Senst, V. Eiselein, and T. Sikora. Robust local optical flow for feature tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(9):1377–1387, 2012.

[20] B.-K. Seo, H. Park, J.-I. Park, S. Hinterstoisser, and S. Ilic. Optimal local searching for fast and robust textureless 3d object tracking in highly cluttered backgrounds. *IEEE transactions on visualization and computer graphics*, 20(1):99–110, 2014.

[21] B.-K. Seo and H. Wuest. A direct method for robust model-based 3d object tracking from a monocular rgb image. In *Computer Vision–ECCV 2016 Workshops*, pages 551–562. Springer, 2016.

[22] C. Tomasi and T. Kanade. Detection and tracking of point features. 1991.

[23] 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*, pages 48–57. IEEE Computer Society, 2004.

[24] H. Wuest, F. Vial, and D. Stricker. Adaptive line tracking with multiple hypotheses for augmented reality. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 62–69. IEEE Computer Society, 2005.