Augmented Reality based on Edge Computing using the example of Remote Live Support

Michael Schneider
Bosch Rexroth AG
Research & Development
Lohr am Main, Germany
Michael.Schneider10@boschrexroth.de

Jason Rambach
German Research Center for
Artificial Intelligence (DFKI)
Augmented Vision Department
Kaiserslautern, Germany
Jason_Raphael.Rambach@dfki.de

Didier Stricker
German Research Center for
Artificial Intelligence (DFKI)
Augmented Vision Department
TU Kaiserslautern
Kaiserslautern, Germany
Didier.Stricker@dfki.de

***Abstract*—Augmented Reality (AR) introduces vast opportunities to the industry in terms of time and therefore cost reduction when utilized in various tasks. The biggest obstacle for a comprehensive deployment of mobile AR is that current devices still leave much to be desired concerning computational and graphical performance. To improve this situation in this paper we introduce an AR Edge Computing architecture with the aim to offload the demanding AR algorithms over the local network to a high-end PC considering the real-time requirements of AR. As an example use case we implemented an AR Remote Live Support application. Applications like this on the one hand are strongly demanded in the industry at present, on the other hand by now mostly do not implement a satisfying tracking algorithm lacking computational resources. In our work we lay the focus on both, the possibilities our architecture offers regarding improvements of tracking and the challenges it implies in respect of real-time. We found that offloading AR algorithms in real-time is possible with available WiFi making use of standard compression techniques like JPEG. However it can be improved by future radio solutions offering higher bandwidth to avoid additional latency contributed by the coding.**

***Index Terms*—Augmented reality, Industrial WLAN, Wireless Networking, Maintenance engineering, Tracking, Sensor fusion, Edge computing, Distributed computing, Smart industry**

I. INTRODUCTION

The huge potential of Augmented Reality primarily reveals in industrial applications. A myriad of prototypes has been implemented in the last few years covering diverse use cases like training [1], marketing [2], assembly [3] and many others.

In defiance of this enourmous market volume most of the AR prototypes were not able to evolve into merchantable products [4]. At least two reasons for this circumstance are evident: First, the availabe AR hardware still lacks computational performance for delivering convincing AR experiences to the user. The google glass introduced in 2012 suffered from overheating and an unacceptably short battery life running the demanding AR algorithms. But even more modern devices like the Microsoft HoloLens continue to have restrictions, for example regarding the quantity of renderable polygons, which applies to smartphones and tablets as well. Second, putting AR applications into operation and keeping them up-to-date locally on different mobile devices (meaning different hardware, OS, platform etc.) is very costly. Traditional (non-AR) applications therefore often draw on a client-server architecture, where only one central unit (the server) has to be configured and maintained.

To adress these two issues we propose a revolutionary approach for the real-time offloading of AR computations to a high performance server using Edge Computing in a wireless network. Edge Computing is an upcoming technology which brings computational resources and services closer to the end-user. The main advantage over Cloud Computing is the reduced latency, so that real-time applications are facilitated [5]. For AR applications the edge server can be used for providing the service of executing CPU- and GPU-intensive AR algorithms remotely. This architecture puts high requirements on the data transmission path between the client device and the edge server because high data rates are to be sent in a short period of time.

One well known use case for Augmented Reality in industries is the so-called Remote Live Support, where a machine operator confronted with an insoluble machine error gets assistance by a remote expert through video transmission and AR. One main point of criticism on current solutions is that due to the limited resources of AR devices no tracking is performed for placing and retaining the expert's annotations in the proper place. To demonstrate the benefits of AR Edge Computing we implemented an AR Remote Live Support application using the proposed architecture, which enables a powerful tracking algorithm, described in chapter III-C.

II. RELATED WORK

A. AR Remote Assistance Systems

For manufacturing companies it is essential that the down time of their machines is reduced as much as possible. Admittedly a machine operator is able to fix most of the occuring errors, but it happens not uncommonly that further advice is needed from a machine or component manufacturer. In some cases telephone support is not enough to get the machine running again and a service engineer will have to attend to it, which implies high expense through longer down time and travel cost [6].

In the given situation a video conference system is of high value. The machine operator puts on some AR glasses and contacts a remote expert via Voice over IP (VoIP). Additionally a live video stream is recorded by the glasses integrated camera and sent to the remote expert, who thereby is in a position to assess the situation on site. Augmented Reality can now extend such an application by providing the facility for the expert to draw assistant virtual overlays into the video stream, which are then shown in the AR glasses of the operator. By this means potential ambiguities, e.g. which button to press, which cable to check etc. can be excluded and the operator is enabled to solve the problem quickly and thus to reduce costs.

A couple of those AR Remote Live Support applications have already been implemented, but most of them do not make use of a proper tracking method, so the annotations made by the remote expert are not registered correctly in $3D$ according to Azumas initial definition of AR [7]. Instead these solutions are either solely able to present data sheets and manuals in the operator's glasses [8] or they use outdated methods like marker tracking [9]. One reason for that are the above-mentioned insufficient computational resources of mobile AR devices.

B. Camera Pose Tracking for Augmented Reality

Camera pose tracking is a fundamental technology for Augmented Reality applications [10]. It enables realistic rendering of $3D$ augmentations and seamless integration into the real world. To this extent a six Degree of Freedom (6DoF) pose consisting of the camera position and orientation in a reference coordinate system is estimated [11].

The camera pose is commonly tracked using the images captured by the camera itself and locating landmarks such as markers or natural features such as edges or lines [12], [13], [14]. These vision-based approaches perform well in general but face challenges when the image quality is compromised, e.g. because of blurring due to fast motion, illumination changes or occlusions. For this reason Sensor Fusion approaches for camera tracking that use additional sensors like inertial measurement units (IMUs) apart from the camera images are often deployed and have been shown to achieve robust tracking especially under fast motion. Such fusion systems are typically based on Statistical Filtering (e.g. Extended Kalman Filter (EKF), Particle Filter (PF)) or learning approaches [15], [16], [17].

C. Wearable AR devices

AR head-mounted displays (HMD) based on see-through optics have been around for a few decades now, although being dedicated solely to defense applications until recently [10]. Nowadays, AR headsets are applied to various markets, such as firefighting, police, engineering, logistics, medicine, education, and more, with emphasis on sensors, specific digital imaging, and strong connectivity [18], [19], [20]. Consumer applications are also emerging rapidly, focused on connectivity and digital imaging capabilities, in an attractive and minimalistic package. Such segmentation has been possible, thanks to recent technological leaps in the smartphone industry (connectivity, on-board CPU power with miniaturization of ICs, development of complex sensors, novel micro-displays, novel digital imaging techniques and battery technology).

Current head-mounted display device manufacturers consider two types of pose tracking methods: inside-out and outside-in tracking. In outside-in tracking, the users head is observed by an external sensor. This is done usually by adding some sort of markers to the head-worn device, and using a camera as an external sensor. One major problem of outside-in tracking is that the range of movements of the user is limited to the field of view of the external camera. In inside-out tracking, the sensor is placed on the device and its pose is computed from the observations. Inside-out pose tracking has several advantages compared to outside-in. The range of movements is not limited by the field of view of an external sensor, but merely by the ability to recognize the surrounding. Fusion with an inertial sensor for more robust pose estimation also becomes possible. However, pose tracking using the visual input from the camera can be computationally very intensive, especially when there is no existing information on the environment and Simultaneous Localization and Mapping (SLAM) approaches have to be deployed [21]. Because of this computational load which can be overwhelming for a real-time application on an embedded processor of an HMD, it is often considered to outsource part of the processing through the network to a powerful cluster of processing units [22], [23]. The delay caused by the network is of crucial importance in this case.

III. REMOTE ASSISTANCE SYSTEM ARCHITECTURE

A. Overview

Our proposed AR Edge Computing architecture consists of two main actors: The client (here: the machine operator), who wants to get some augmented reality information superimposed on his perception of the physical world, and the edge server, which has the task to bring the AR experience to the user and therefore to run the computationally challenging AR algorithms. While the client device can be any mobile device having an integrated camera, the edge server should be a high-performance PC with an appropriate graphics card.

As we can see in figure 1 in the actual use case additionally a third party is involved, namely the remote expert. As he has to be capable of drawing annotations into the video stream coming from the operator, he should be equipped with a laptop or a tablet PC.

The activity diagram in figure 2 shows how the overall Remote Live Support process looks like. We assume that the connection between the operator and the remote expert has been established beforehand.

First, the operator's mobile AR device camera grabs frames at a configured frame rate and sends them together with some IMU data to the edge server. It is important to understand that the process there is divided into two asynchronous subprocesses. On the one hand the server accomplishes the

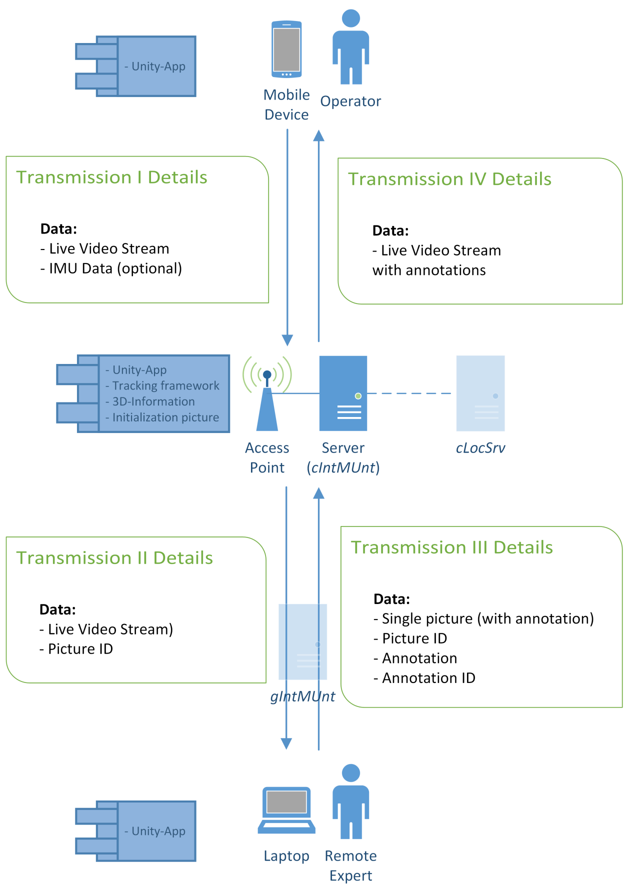
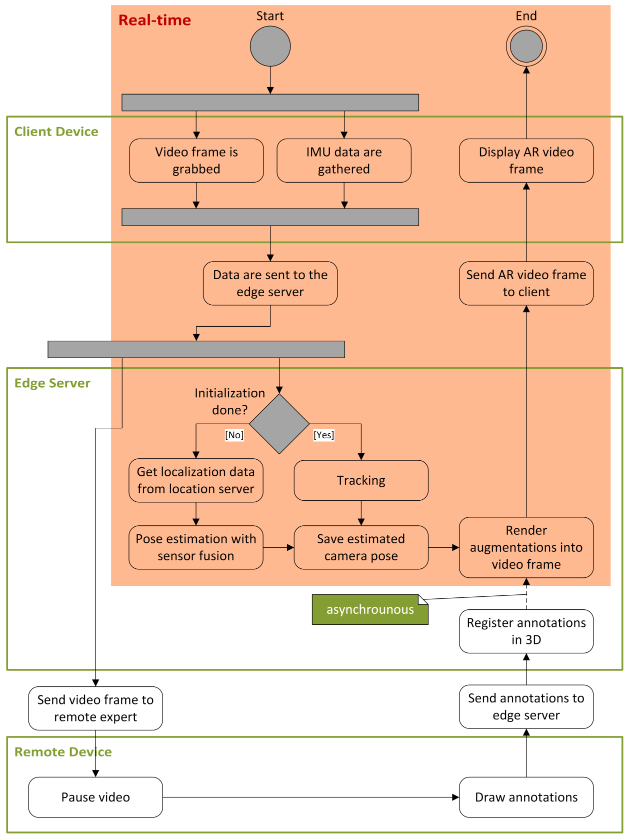


Fig. 1. Remote Support System Architecture Overview.



Real-time
Start
End
Client Device
Video frame is grabbed
IMU data are gathered
Display AR video frame
Data are sent to the edge server
Send AR video frame to client
Edge Server
Initialization done?
[No]
[Yes]
Get localization data from location server
Tracking
Pose estimation with sensor fusion
Save estimated camera pose
Render augmentations into video frame
asynchrounous
Register annotations in 3D
Send video frame to remote expert
Send annotations to edge server
Remote Device
Pause video
Draw annotations

Fig. 2. Remote Live Support activity diagram

tracking process, which is introduced in detail in III-C. The estimated camera pose of each frame is saved on the edge server. On the other hand it forwards the video stream over a gateway (gIntMUnt in figure 1) to the remote expert using available WAN infrastructure as the expert can sit thousands of kilometres away.

The remote expert now sees the incoming video stream and is able to pause it at any time to draw some helpful hints into it. Those annotations are sent back to the edge server together with the corresponding picture ID. In this manner the edge server is able to match the annotated picture with the saved camera pose and the annotations can be registered properly in $3D$. As soon as new annotations are registered the server can render them into the camera feed, which is sent back to the client permanently, regardless of whether annotations are available or not.

Attention should be paid to the fact that the cycle time between the grabbing of a camera frame to its augmented visualization (red background in figure 2) has to be kept as small as possible. Experts estimate the maximum end-to-end latency a user wearing a HMD can perceive at $10ms$ [24], where in some non-HMD applications considerably higher latencies are tolerable [25].

Furthermore a bidirectional audio link is established between both human actors, on which we do not elaborate in this paper.

B. User side - Mobile device

As mentioned before the client device can be any mobile device running an operating system and posessing an integrated camera and a IMU. The only task of this device is to grab live data from these two components, send it to the edge server and then to display the incoming video data. For faster devices it would also be possible only to offload the tracking process and to do the rendering on the client device itself. Due to this option in the current system we decided upon a Unity 3D application as client front-end, because of the strong rendering capabilities and usability of Unity. Nevertheless any other technology, e.g. a web application with WebGL, could do this job equally well.

C. Server side - Edge Cloud

The image processing server (cIntMUnt in figure 1) is responsible for all operations that are computationally too demanding for the mobile device at the user side, namely the camera pose tracking and possibly the rendering of augmentations.

The cIntMUnt constantly receives the stream of captured video and inertial data (if applicable) from the user. For the task of pose tracking an initialization step is required. This is achieved by extracting visual edge features (e.g ORB) [26]) from the received images and matching to a preregistered database of features previously collected from the application environment. This procedure can be challenging and prone to failure if repeating patterns occur in the environment. These issues can be overcome by using an additional source of positional information such as a network wireless signal based localization (received from cLocSrv) to reduce the set of possible feature matches from the database.

After successful initialization a robust frame-to-frame tracking approach can be initiated. The proposed approach considers tightly coupled visual-inertial fusion using an EKF with inertial measurements as control inputs for the prediction step similarly to [15]. The tracking procedure outline is as follows:

- The appearance of the visual features to be tracked is rendered based on the prediction step of the EKF done using the inertial measurements.
- The predicted features are matched to the visible ones in the current video frame. This requires only a fast search in the neighbourhood of each predicted feature position.
- The matched $2D$ feature positions in the image together with their known $3D$ positions are used for the correction step of the EKF. The current pose can then be read from the filter state.

In case inertial measurements are not available the features are matched from the previous frame to the next and the camera pose is calculated from the matches within an outlier rejection framework (e.g. RANSAC [13]). The use of an EKF and inertial measurements however benefits the system in that it increases the robustness of the tracking and reduces the number of features that is required to be tracked, thus significantly decreasing the processing time per frame needed.

The recent history of camera poses corresponding to IDs of images are saved in the cIntMUnt. When an annotation drawn by the remote expert is received, the server cIntMUnt uses the pose information and the existing $3D$ environment information to define a plane in the real world that the annotation is attached to. This is done by a simple calculation of the highest overlap of the annotation in the image to the real environment planes. Subsequently, the annotation can be rendered to every incoming frame from the mobile device after performing localization. For the rendering, a framework like Unity or any other rendering engine can be used. The rendered image is then transmitted back to the end user to be overlaid on the live view.

D. Remote Expert

The application for the remote expert has to provide at least the following functionalities:

- Display the received video stream from the machine operator
- Pause the video stream
- Provide tools for drawing annotations into the video stream (and for deleting them again)
- Send the annotations back to the edge server

Since a Unity 3D application was deployed on both the client and the edge server, we also use Unity 3D for the remote expert's application in order not to run into compatibility conflicts of the exchanged data.

IV. EVALUATION

To evaluate the real-time capabilities of our developed system we analyzed the emerging latencies for the transmission and the tracking process seperately.

A. Network Real-Time capability

As mentioned in section III-A the requirements of AR applications regarding end-to-end latency are very demanding. Our proposed architecture unfortunately introduces a new source for delay, namely the transmission path between the client and the edge server. Disregarding all other advantages the AR Edge Computing entails, the time saved by processing the algorithms remotely should exceed the additional transmission time overhead.

Basically there are two possibilities how to transfer the video data. We can either send them as raw data or encode them using a intraframe or interframe compression technique, where in our implementation we only were intent on using the second-mentioned. If no compression technique was used, a 752×480 video stream ($24Bit$ per pixel) at $30fps$ would produce an application data rate of $\approx 260Mbps$.

Whereas compression means strongly economizing the produced data rate the process itself contributes some delay at the same time. In figure 3 we measured the average time needed for coding and decoding several test pictures with different JPEG compression levels and resolutions and compared it to the calculated time needed for transmitting raw data. Here we only took the data rate produced by the application into account, applied protocols like TCP will add up some overhead.
As we can see in figure 3 for an adopted bandwidth of $150Mbps$ it takes around $14.5ms$ in total to encode a 752×480 frame with JPEG75 on the client-side, transmit it to and decode it on the server for further processing while sending the raw data would involve a theoretical latency of $57,75ms$. Considering the way back from server to client we can assume similar prequisites, but no decoding has to be performed on the client, since the returning JPEG images are to be displayed immediately (corresponds to approximately $10.5ms$).

B. Camera Tracking Resources

In this section we present measurements of the runtime of the proposed tracking and rendering approaches from section III-C on a regular PC processor. With these results we demonstrate the need for a processing unit more powerful than the end-user devices such as smartphones or AR-glasses. Furthermore, by combining the processing time calculated here with the estimated network delays from section IV-A we obtain an estimate of the total delay that the end-user is affected by.

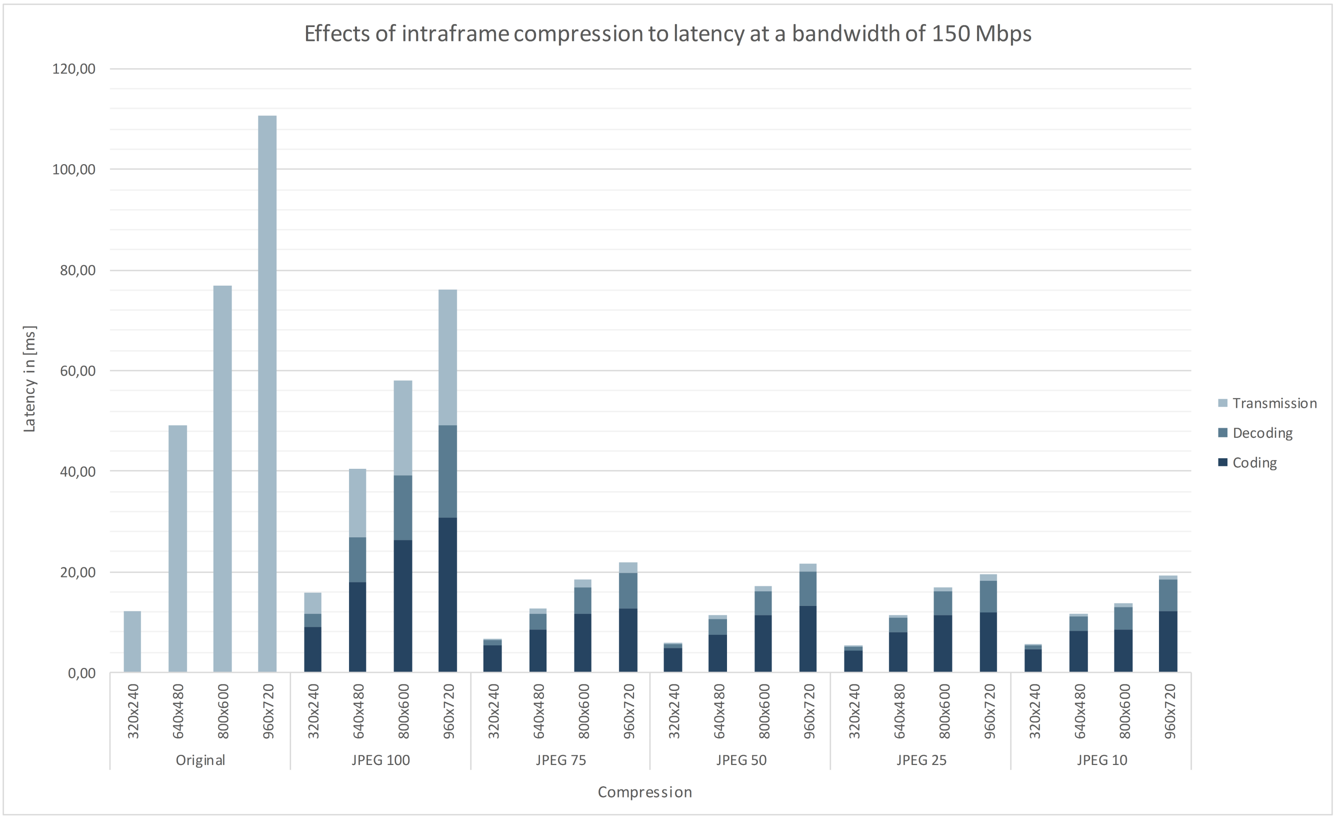


Fig. 3. Comparison of transferring coded and uncoded video data.

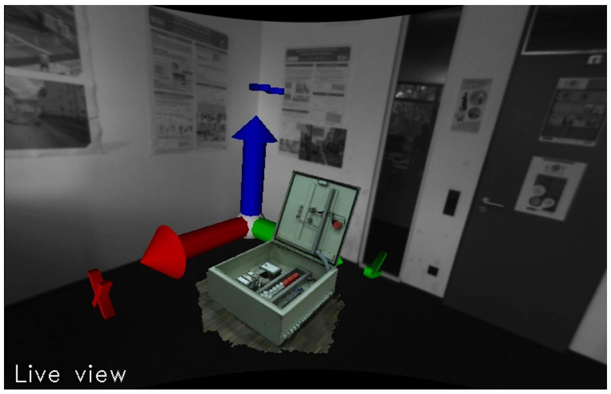


Fig. 4. Camera view from the tracking experiment with $3D$ augmentations

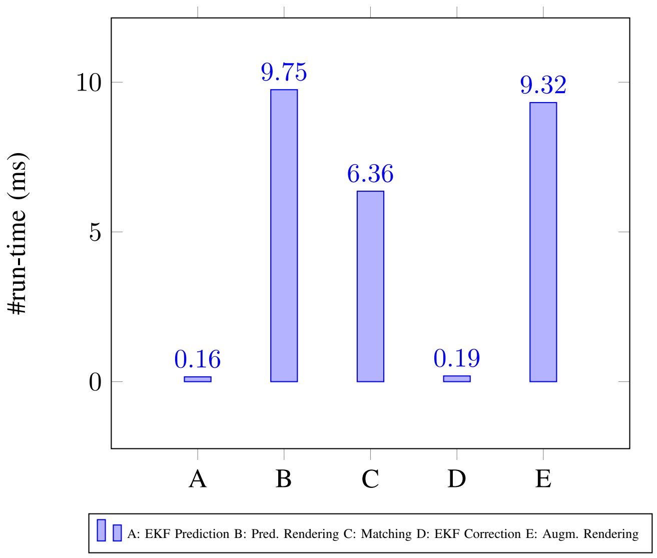


Fig. 5. Runtime measurements of tracking algorithm

We performed our experiment in a room environment with a $3D$ model of the walls (figure 4) and registered ORB features for initialization and KLT features for the tracking. An EKF as described in section III-C is used to fuse visual feature matches from a uEye camera with synchronized inertial measurements from an XSens IMU. The camera provides images at a resolution of 752×480 pixels and the CPU used was an Intel Xeon 3.07GHz, 12GB RAM. The measured average time required for the different operations during pose tracking and rendering of a $3D$ augmentation is given in figure 5: From our runtime measurements we observe that the filter prediction and update operations are executed very fast (less than $0.2ms$), and that the matching of the features and the rendering are the most expensive operations. In particular, the rendering of the expected scene based on the predicted pose and the final rendering of a $3D$ augmentation that is presented to the end user require about

$10ms$ each. The feature matching also requires a significant amount of time of about $6ms$ even though it is done by a local search for each feature instead of a full image search taking advantage of the pose prediction. Furthermore, a sparse feature set is used since the EKF only requires a few good feature matches to perform its correction step. The total time required for all server operations adds up to $25.78ms$, which leads to an overall end-to-end delay of roughly $50ms$ $(14.5ms + 25.78ms + 10.5ms)$ for a single frame using a resolution of 752×480.

V. CONCLUSION

In this paper we presented an AR application using Edge Computing. Referring to this innovative combination of the given technologies we pointed out that using the architecture mentioned, existing difficulties regarding the performance of mobile AR devices like smartphones, tablets and glasses can be overcome. Additionally it brings advantages like simplification of update rollouts or platform independence on client side. The crucial factor is the end-to-end latency, which is elevated by the video transmission between client and server. For sending, tracking, annotating and receiving a 752×480 compressed video frame we measured an end-to-end latency of around $50ms$, which is acceptable for handheld AR systems but too high for most HMD solutions.

To achieve better results, communication technologies with higher bit rates and smaller latencies are needed to relinquish the coding and decoding step and permit the transmission of raw video data. One project which attends to these requirements as well as to any others warranted by industry 4.0 applications is proWiLAN.

ACKNOWLEDGMENT

This work has been partially funded by the German Federal Ministry of Education and Research as part of the research project proWiLAN under grant numbers 16KIS0248 and KIS3DKI018. The responsibility for this publication lies with the authors.

REFERENCES

[1] B. Besbes, S. N. Collette, M. Tamaazousti, S. Bourgeois, and V. Gay-Bellile, "An interactive augmented reality system: a prototype for industrial maintenance training applications," in *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pp. 269–270, IEEE, 2012.

[2] A. Javornik, "Classifications of augmented reality uses in marketing," in *IEEE International Symposium on Mixed and Augmented Realities 2014*, IEEE, 2014.

[3] P. Horejsi, "Augmented reality system for virtual training of parts assembly," in *25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014*, vol. 100, pp. 699–706, Elsevier, 2015.

[4] M. Schneider, M. Aleksy, and D. Stricker, "Wireless network-communcation as an enabler for mobile augmented reality applications in industry," in *VDE-Kongress 2016 - Internet der Dinge*, VDE, 2016.

[5] R. Want, B. N. Schilit, and S. Jenson, "Enabling the Internet of Things," *IEEE Computer*, vol. 48, no. 1, pp. 28–35, 2015.

[6] M. Aleksy, E. Vartiainen, V. Domova, and M. Naedele, "Augmented reality for improved service delivery," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, pp. 382–389, IEEE, 2014.

[7] R. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments 6*, vol. 06, no. 4, pp. 355–385, 1997.

[8] H. Weckbrodt, "Druckerei-Techniker bekommen augengesteuerte Datenbrillen." http://oiger.de/2015/10/01/druckerei-techniker-bekommen-augengesteuerte-datenbrillen/155815, October 2015. Last checked: 2016-12-20.

[9] J. Wang, Y. Feng, C. Zeng, and S. Li, "An augmented reality based system for remote collaborative maintenance instruction of complex products," in *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, (New Taipei, Taiwan), pp. 309–314, Aug. 2014.

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

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

[12] A. Pagani, J. Koehler, and D. Stricker, "Circular markers for camera pose estimation," in *WIAMIS 2011: 12th International Workshop on Image Analysis for Multimedia Interactive Services, Delft, The Netherlands, April 13-15, 2011*, TU Delft; EWI; MM; PRB, 2011.

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

[14] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 932–946, 2002.

[15] G. Bleser and D. Stricker, "Advanced tracking through efficient image processing and visual–inertial sensor fusion," *Computers & Graphics*, vol. 33, no. 1, pp. 59–72, 2009.

[16] G. Bleser and D. Stricker, "Using the marginalised particle filter for real-time visual-inertial sensor fusion," in *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*, pp. 3–12, IEEE, 2008.

[17] J. Rambach, A. Tewari, A. Pagani, and D. Stricker, "Learning to fuse: A deep learning approach to visual-inertial camera pose estimation," in *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, 2016.

[18] M. Mekni and A. Lemieux, "Augmented reality: Applications, challenges and future trends," in *Applied Computational ScienceProceedings of the 13th International Conference on Applied Computer and Applied Computational Science (ACACOS 14) Kuala Lumpur, Malaysia*, pp. 23–25, 2014.

[19] M. Dunleavy and C. Dede, "Augmented reality teaching and learning," in *Handbook of research on educational communications and technology*, pp. 735–745, Springer, 2014.

[20] M. C. Macedo, A. L. Apolinário, A. C. Souza, and G. A. Giraldi, "A semi-automatic markerless augmented reality approach for on-patient volumetric medical data visualization," in *Virtual and Augmented Reality (SVR), 2014 XVI Symposium on*, pp. 63–70, IEEE, 2014.

[21] 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*, pp. 225–234, IEEE, 2007.

[22] P.-H. Chiu, P.-H. Tseng, and K.-T. Feng, "Cloud computing based mobile augmented reality interactive system," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3320–3325, IEEE, 2014.

[23] P. Hasper, N. Petersen, and D. Stricker, "Remote execution vs. simplification for mobile real-time computer vision," in *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, vol. 3, pp. 156–161, IEEE, 2014.

[24] W. Pasman and F. W. Jansen, "Distributed Low-latency Rendering for Mobile AR," in *IEEE and ACM International Symposium on Augmented Reality Proceedings (ISAR 2001)*, (New York, USA), pp. 107–113, Oct. 2001.

[25] F. P. Brooks, "What's Real About Virtual Reality?," *IEEE Computer Graphics and Applications*, vol. 19, no. 6, pp. 16–27, 1999.

[26] 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*, pp. 2564–2571, IEEE, 2011.