Tracking and Integration Aspects of a Mobile Augmented Reality Tool for Shipbuilding

J-P. Arimaa, R. Suominen, A. Euranto, O. Lahdenoja, T. Knuutila and T. Lehtonen
Business and Innovation Development (BID)
20014 University of Turku, Finland
{ jpaari,rajusuo,akjeur,olanla,knuutila,tetale }@utu.fi

***Abstract*—In this paper, some of the architectural and tracking aspects of the software engine used in our on-going MARIN (Mobile Augmented Reality Tool for Marine Industry) project are described. The purpose of the project is to construct an augmented reality tool for shipbuilding. The future target usages of the system vary from inspection to actually improving the efficiency of some of the repeating procedures in shipbuilding. The system also includes means for indoor localization and access to the pre-existing CAD model of the ship. In our case, the tool is hosted by Unity 3D, which allows the interfaces between the inertial sensors and the visual tracking thread. Some of the implementation issues related to these interfaces are described, while some challenges related to visual tracking are also discussed.**

I. Introduction

The mainstream approach for visual tracking is to extract keypoints [1] [2] [3] and to determine the pose of the observer from correspondeces between the preceding image frames and the camera input frame. For pose initialization and tracking, also the possibility of storing reference keypoints related to particular objects may be used [4]. With mobile devices, one of the challenges in real-time analysis is the computational cost of the keypoint extraction [5]. On the other hand, the performance of mobile devices is increasing and the usage of integrated modern GPUs (Graphics Processing Units) may provide solutions to this in the future.

MARIN (Mobile Augmented Reality Tool for Marine Industry) project is an on-going project between the industry partners, TEKES (The Finnish Funding Agency for Technology and Innovation) and University of Turku, Finland. The two-year project is in its midpoint as it was started in June 2012. The purpose of this project is to construct an augmented reality tool for shipbuilding. One of our project partners has provided access to the local shipyard at Turku for testing the developed algorithms and determining the usage requirements. The specific conditions in shipbuilding such as varying illumination, temperature, humidity, dust and interference assess special demands for the tracking.

The paper is organized in the following way. Section II shows the overall system architecture. Sections III and IV describe the implementation aspects of the tracking and rendering threads, respectively. Finally, discussion and conclusions are given in sections V and VI.

II. System Architecture and Integration

In our case, very exact weekly updating 3D designs of the operation environment exist. The overall system diagram of the tool is described in Fig. 1. Unity3D [6] performs rendering. The tracking thread uses a plug-in to the OpenCV [7]. The main advantage of Unity3D is its cross-platform and plug-in support. We use OpenCV library functions to calculate the pose of the observer from the corresponding points between the 3D world model and the image plane. A gyroscope is interfaced from Unity3D allowing the integration of inertial tracking data from the sensors within AR glasses. The tracking and rendering threads are planned to co-operate forming a feedback from the pose of the 3D world model to the pose of the user [8] [9].

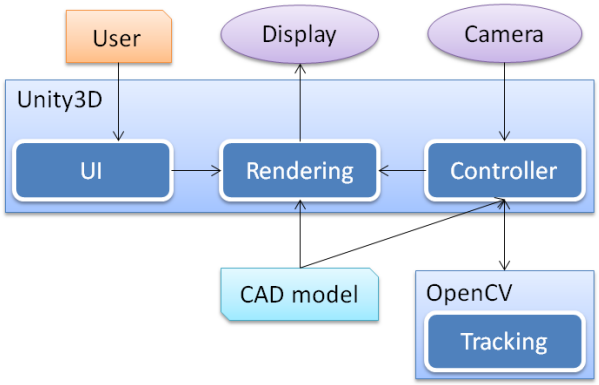

Fig. 1. System diagram, including tracking and rendering threads in Unity3D.

III. Initialization and Tracking

Pose estimation is carried out in initialization and tracking. For initialization, there may be a need for guided manual input so that the user picks points from the screen space and from the 3D model. Before these, either manual (e.g. map based) or automatic (e.g. visual recognition, RFID, WLAN or NFC) coarse grain positioning determines the location of the user. Then it may be assumed, that the tool can provide the exact 3D model of the environment where the user is currently located. To assist manual initialization, corner detection (e.g. Harris [10], FAST [11], SUSAN [12]) can be used for localizing the points corresponding to 3D model. After the pose initialization, the tracking thread in Unity3D can acquire information from the inertial sensors to assist the point tracking in OpenCV.

The position and the viewing direction (pose) of the observer can be calculated, for example, by solving perspective-n-point problem (PnP) [13]. PnP requires the coordinates of an object in world space (x_w, y_w, z_w) and the corresponding

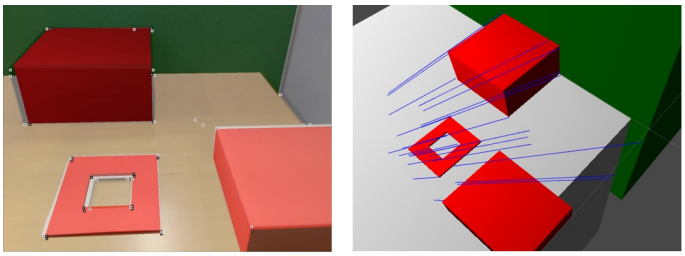

Fig. 2. Ray shooting method. Image on the left shows the camera 2D view of the scene aligned with the 3D objects. The image on the right shows an example of rays shooted from the camera pose to the 3D model in Unity3D.

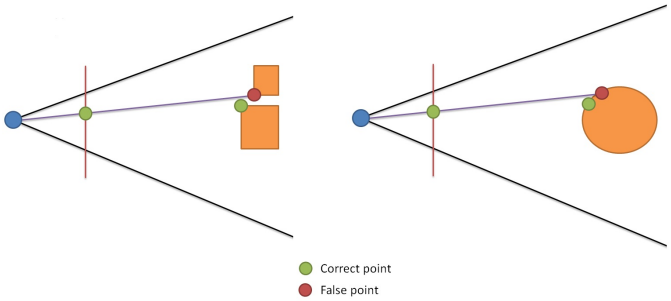

Fig. 3. An example of a missed hit between the image plane and the actual 3D world object.

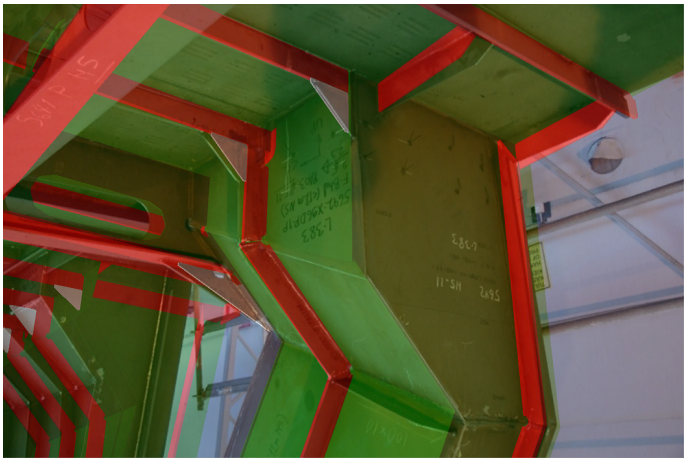

Fig. 4. Image from the ship aligned with rendered image.

coordinates in screen space (x_i, y_i) which holds the view of the real world. For tracking purposes, the model of the world space does not change in our case, but by moving the camera the respective screen points do change. Therefore, the screen points need to be tracked. Nevertheless, it is possible to lose screen points if they drift outside the image plane or by other reasons such as rapid camera movements. Thus, it is required that new world-screen-point pairs are found. In addition, the tracked screen points need to be discernible (for example a blank white wall is very difficult to track).

An initial method would be finding any visible point in the world space and to project that particular point to image plane. The downside is that the point is probably very difficult to track. Another method would be to project the model vertices and assume they are easier to track. Nevertheless, this method has the same downside though it is probably a better solution.

Other way would be finding points that are easy to track and shoot a ray from the camera through the image plane and find the collision point in the world space. These points can be found with corner detection (such as Harris corner detection [10]). In addition, it is possible to compare the collision points to vertices and if the distance between a collision point and a vertex is shorter than a given threshold it could be assumed that they are the same point. This would allow the calculation of correction factors. Using corner detection and ray casting is naturally slower than simple world point projection (though finding visible vertices requires also more calculation). Finding the corresponding screen and world points also requires that the initial pose is known with a very small error marginal. Nevertheless, that method has the possibility of miss hit or finding wrong vertex (false positive). Fig. 2 shows the shooting of rays into the 2D image plane and the paths of the rays in Unity3D. Fig. 3 shows a potential missed hit of the ray shooting.

Tests implemented so far have indicated that it is difficult to find the exact corners that correspond to model vertices solely based on 2D image data. The corner detection methods are, for instance, sensitive to proper parameters used as thresholds. Another alternative would be to use lines and to find their endpoints in order to support corner detection. Lines can also be used to determine the pose of the observer directly [2]. Line extraction with Hough transform, for example, have been demonstrated to work with mobile phones with limited accumulator space resolutions [14]. Based on our tests so far on a laptop PC, this approach provided better results. Keypoint locations of the scale-invariant descriptors such as SIFT [1] and SURF [2] do not typically match to the vertices of the 3D model, but they could still be used to assist the tracking of other specific locations.

IV. Performance and Rendering

Tests with 640x480 sized images taken from the shipyard indicated that the time elapsed for Harris and Shi-Tomasi corner detection was typically less than 20ms and time for Hough transform was appriximately 20-40ms with standard OpenCV functions when using i5 @ 2.5GHz laptop PC. Naturally the running time depends on the selected parameters, especially for the Hough transform. The elapsed time for solving the perspective-n-point problem (PnP) [13] in OpenCV was typically less than 2ms. Fig. 4 shows an example of the camera view aligned to the 3D world model in Unity3D.

Table I shows the rendering performance of Unity3D with different devices when one block of the ship (shown in Fig. 5) is rendered. Rendering is possible to complete with only 17 draw calls due to the usage of static batching (in parenthesis is the number of draw calls without batching). Even though the amount of frames per second (FPS) is acceptable it must be kept in mind that there are several other tasks that need also to be done (such as possible point tracking or keypoint detection and pose estimation). Table I only considers the time required for rendering. Devices used for performance testing were Nokia Lumia 920 (S4 @ 1,5 GHz), Apple iPad 3 (A9 @ 1,0 GHz) and Samsung 700T Tablet (i5 @ 1,7 GHz).

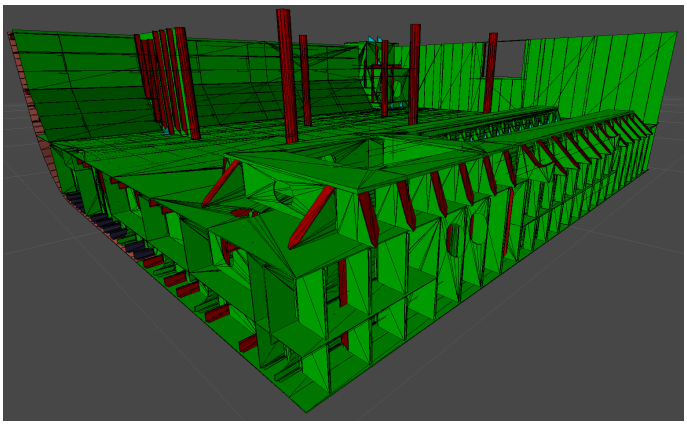

Fig. 5. Rendered model for the performance tests.

Vertices	Triangles	Draw calls	700T (FPS)	iPad (FPS)	Lumia (FPS)
156.1k	130,6k	17(1431)	60	15	45

TABLE I. PERFORMANCE TESTS IN RENDERING A CERTAIN SHIP PART MODEL.

V. DISCUSSION

In our case, very exact 3D designs of the whole ship exist during the manufacturing. The tracking architecture was designed to utilize this information. In the future, also SLAM (Simultaneous Localization and Mapping) based tracking [15] could be implemented to the system, for instance, by using OpenCV library functions. Although in some cases there might not be enough details in the steel walls of the ship for keypoint based tracking, given a limited camera FOV, there usually exist various machines and other temporaty objects in the scene, which could provide the necessary keypoints. Another issue to study in the future is how to distinquishing the model objects from other possible obstacles in the scene.

One of the repeating procedures in shipbuilding which is targeted to be made easier by the tool is assistance in making changes and change requests. For instance, if new openings for wiring need to be cut, the approval for ship structural strength must be transferred between the installation workers and the engineering staff. Information on the opening and its location could then be transferred to the ship 3D model with the tool.

Certain aspects on the tracking architecture used in ongoing MARIN project were described in this paper. In order to achieve real-time operation with a mobile platform, also the performance requirements were coarsely estimated. For real-time operation, all tasks should be done preferably in less than 40 milliseconds (which means 25 FPS or higher), which is challenging if the whole system is integrated to a mobile phone or tablet. To facilitate the tracking, our intention is to incorporate a smart camera [16] to the system, which enables high tracking frame-rates with compact size and very low power consumption.

VI. CONCLUSION

This paper presented some of the tracking status of the ongoing MARIN project. The architecture of the software engine, which performs tracking and rendering was described, as well as some alternatives to improve the tracking. The proposed architecture and tracking solutions were designed bearing in mind the current limitations of mobile phones and tablets, while still providing a scalable solution for AR architecture for various platforms. Our future work includes continuing the software development and studying which parts of the tracking could be benefited by an attached smart camera system.

ACKNOWLEDGMENT

The MARIN project is carried out in collaboration with partners Nokia Oyj, BA Group Oy, Cadmatic Oy, Kovilta Oy, Lloyd's Register EMEA Helsinki, Nextfour Group Oy, Premode Oy, STX Finland Oy, Wallius Hitsauskoneet Oy, Microsoft Oy, Southwest Finland Center of Expertise - Maritime Cluster Programme / Machine Technology Center Turku Ltd, and Southwest Finland Center of Expertise / Ubiquitous Computing Cluster Programme / Yrityssalo Oy. Most of the funding comes from Tekes - the Finnish Funding Agency for Technology and Innovation.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

[2] H. Bay, "From Wide-baseline Point and Line correspondences to 3D", PhD. Dissertation, Swiss Federal Institute of Technology, ETH, Zurich.

[3] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, "BRIEF: Computing a Local Binary Descriptor Very Fast", IEEE Transactions on Pattern Analysis and Machine Intelligence, 34, 7, pp. 1281 - 1298, 2012.

[4] M. Schaeferling, U. Hornung, G. Kiefer, "Object Recognition and Pose Estimation on Embedded Hardware: SURF-Based System Designs Accelerated by FPGA Logic", International Journal of Recongurable Computing, vol. 2012, Article ID 368351.

[5] R. Hoffman, H. Seichter, G. Reitmayr, "A GPGPU Accelerated Descriptor for Mobile Devices", IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 289-290, 2012.

[6] http://www.unity3d.com

[7] G. Bradski, "The OpenCV Library", Dr. Dobb's Journal of Software Tools, (2000).

[8] G. Reitmayr and T. Drummond, "Going out: robust model-based tracking for outdoor augmented reality", IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 109 - 118, 2006.

[9] G. Klein and D. Murray, "Full-3D Edge Tracking with a Particle Filter", Proceedings of the British Machine Vision Conference (BMVC), pp. 1119-1128, 2006.

[10] C. Harris and M. Stephens, "A combined corner and edge detector", Proceedings of the 4th Alvey Vision Conference. pp. 147151, 1988.

[11] M. Trajkovic and M. Hedley, "Fast corner detection", Image and Vision Computing 16, 2, pp. 7587, 1998.

[12] S. M. Smith and J. M. Brady, "SUSAN - a new approach to low level image processing", International Journal of Computer Vision, 23, 1, pp. 4578, 1997.

[13] D.F. Dementhon, L.S. Davis, "Model-based object pose in 25 lines of code", International Journal of Computer Vision, 15, 1-2, pp. 123-141, 1995.

[14] A. Hartl, G. Reytmayr, "Rectangular target extraction for mobile augmented reality applications", IEEE International Conference on Pattern Recognition (ICPR), pp. 81-84, 2012.

[15] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM", IEEE Transactions on Pattern Analysis and Machine Intelligence, 29, 6, pp. 1052-1067, 2007.

[16] http://www.kovilta.fi