Indoor Localization using SLAM in parallel with a Natural Marker Detector

Lucas Teixeira
Department of Informatics
Pontfical Catholic University of Rio de Janeiro
lucas@tecgraf.puc-rio.br

Alberto B. Raposo
Department of Informatics
Pontfical Catholic University of Rio de Janeiro
abraposo@tecgraf.puc-rio.br

Marcelo Gattass
Department of Informatics
Pontfical Catholic University of Rio de Janeiro
mgattass@tecgraf.puc-rio.br

ABSTRACT

Indoor localization poses is a challenge to computer vision research, since one may not make use of GPS-based devices. A classic approach commonly used in museums, research institutes, etc, is the use of fiducial marker to track the users position. However, this approach is intrusive into the ambient and not always possible. A possible solution would be natural marker detection, but algorithms for this, such as SURF, have not yet achieved real-time performance. A promising approach is a Visual Simultaneous Localization and Mapping (VSLAM) algorithm, which, starting from a known position, is capable of generating a map of the surrounding environment in portable systems. The problem of SLAM algorithms is theirs error accumulation that builds up during the movement. This work presents an algorithm to locate 3D positions in non-instrumented indoor environments using a web camera. We define a hybrid approach, using a pattern-recognition method to reinitialize whenever possible a VSLAM algorithm. An implementation of the proposed algorithm use well-known computer vision algorithms,such as SURF and Davison's SLAM. In addition, tests were made on datasets from walks inside a room. Results indicate that our approach is better than a fiducial marker tracking and pure SLAM tracking in our test environment.

Categories and Subject Descriptors

I.2.10 [**Vision and Scene Understanding**]: Motion; I.5.4 [**Applications**]: Computer vision

General Terms

Algorithms;Experimentation

Keywords

SLAM ; Tracking ; Indoor Localization ; Picking ; Computer Vision

SAC'13 March 18-22, 2013, Coimbra, Portugal.

1. INTRODUCTION

Augmented Reality (AR) is becoming an important means of communication between digital systems and users. The entertainment industry, mobile map guidance, and geographically distributed collaborative work in large industrial projects are areas where AR is blooming. Nevertheless, commercial and industrial applications are still in their infancy because AR algorithms are still not robust and accurate enough to sustain critical usage.

Tracking is one of the most difficult areas in AR systems. Tracking algorithms give the position of the camera as it moves in an environment. This information is crucial to align the virtual world with the real world. Thus, this alignment creates the augmented world, the augmented reality. Tracking in commercial and industrial sites is difficult for many reasons: GPS approaches do not work inside buildings; magnetic solutions do not work with metals around; mechanical encoders, like FaroArm, have a small functional area; infrared trackers have problems with occlusions and do not allow other sources of infrared light, such as sunlight and incandescent light bulbs.

There are many applications of tracking solutions in industrial facilities. For instance, distribution centers and warehouses store hundreds of thousands of objects and need an efficient retrieval method. An important task is to quickly locate where a given object is stored. Codes are usually employed to identify shelves and places within them. When an object is needed, the worker finds the code in the inventory and looks for it in the warehouse. This operation is called *picking*. Schwerdtfeger et al. [14] present an interface evaluation of a picker using an HMD (head-mounted display) in order to improve the accuracy and the efficiency of the picking process. The HMD shows the path to the object. However, this application requires a tracking system to present the correct path, as does a GPS car navigation system. As a proof of concept, Schwerdtfeger et al. use an infrared tracking system, like Vicon[1], but this kind of solution is not feasible in practice, as explained above.

Davison [5] presented a revolutionary system using a SLAM (Simultaneous Localization and Mapping) approach which has a great potential to solve these industrial problems. From a known position, SLAM systems are capable of moving in space and keeping their position known while generating a map of the surrounding environment. These systems work incrementally. From a known position, they map new features that are in the field of vision. These new points

[1] www.vicon.com

cover an area larger than the original view frustum and, as the camera moves, they are used in the calibration process. That is, from a new view point there are known points to localize the camera and unknown points that are mapped to further use. Davison's system was the first to achieve real-time processing in a common desktop computer and using images from a single camera as input. However, it has some drawbacks: the localization error builds up quickly , invalidating the camera tracking as the user moves in a large environment ; instant occlusion by another person or fast movements that produce blurred images also invalidate the tracking.

This paper presents a methodology to improve a Visual SLAM-based tracking algorithm in order to reduce the localization error of the camera using only computer vision. We use texture-detection algorithms to improve the SLAM algorithm, achieving a tracking algorithm that works in well-known environments of any size, without illumination restrictions, and performs well in partially dynamic environments, such as a warehouse where part of the environment (e.g. shelves) is static and part (e.g. the contents of shelves) is dynamic. We define a hybrid approach, using a pattern-recognition method to reinitialize, whenever possible, a Visual Simultaneous Localization and Mapping (VSLAM) algorithm. So, we expect that the camera will always be relatively near to a known individual natural marker/texture. When a marker shows up, the map is cleaned and the localization error is eliminated. Using this strategy, the VSLAM algorithm becomes scalable, allowing it to perform in large environments. The use of a pattern detector invariant under illumination changes allows the presence of windows in the indoor environment. Theoretically, there is no restriction to the use of the proposed algorithm in outdoor sites, but differential GPS is better and easier to use, because it does not need prior mapping of the textures. The methodology presented expects a prior mapping of some textures on the environment to be used as a natural marker. However, no instrumentation is required. It is necessary only to use geographically located photos in the mapping stage.

This paper is organized as follows. In Section 2 we discuss related work. The proposed algorithm is presented in Section 3. Section 4 tests the limits of the proposed algorithm and compares it with similar solutions. Finally, conclusions are presented in Section 5.

2. RELATED WORK

AR researchers have been studying solutions to be used inside buildings for a while. Wagner and Schmalstieg [15] presented an AR system to be used inside a building with fiducial markers spread throughout its walls. These markers must be distributed every two meters on all walls. For each captured image, the position of the PDA is calculated, and the user is given directions to the room where he/she wants to go. The low processing cost is the best quality of this system, since it processes a single fiducial marker at a time. On the other hand, this system has severe limitations. One of them is the absence of tracking in the area between two markers, restricting its use to objects close to the markers. Another problem is the interference in the environment. In most places it is impossible to put the large number of fiducial markers required by this system. The algorithm proposed in the present paper can perform the same kind of tracking as that system, but it uses a reduced number of natural landmarks, since it uses a SLAM algorithm to obtain the camera position in these spaces in between landmarks.

At the other extreme, there are algorithms that are capable of localizing the camera in completely unknown environments, as long as the camera starts from a known position. They are VSLAM algorithms, which generate a map of the environment while the camera moves. However, these algorithms have a serious problem regarding the accumulated error along the camera movement. Figure 1(a) shows the result of the tracking throughout the internal courtyard of Pembroke College (Figure 1(c)) using the implementation of the VSLAM algorithm [5] that we used in the present work. The resulting error is caused by the error accumulated during the movement. Since the localization and mapping process in SLAM has an approximation error, and new points added to the map use points that were included in the map by the same process as reference, maps generated by SLAM algorithms have exponential error. To correct the generated map in a post-processing step, loop-close algorithms may be used [16, 4]. In this kind of correction, the camera needs to complete a closed loop and continue for a few more meters. With this procedure, it is possible to match the points at the end of the loop with those at the beginning, and then correct the whole map. We can see the result of this kind of algorithm in Figure 1(b). Therefore, for the type of real-time application we are considering, a VSLAM algorithm cannot be used alone, since the exponential error is not acceptable for the camera tracking.

(a) SLAM results (b) Corrected results

(c) Pembroke College

Figure 1: Results of loop-closing correction. Source: [16]

Another option for tracking was proposed by Castle et al. [2]. This approach uses Klein and Murray's PTAM algorithm [8], which maps the environment in real time based on keyframes and bundle adjustment. PTAM has a limitation on tracking amplitude, since, at each new keyframe, the optimization time increases considerably. To tackle this issue, Castle et al. proposed the concept of interest islands, where AR data can be shown. For each island, one can select a tex-

ture to be the island's identifier. Using a real-time texture recognizer, Castle's algorithm identifies the island, calculates the position of the camera, and starts PTAM tracking. This approach is better than that of Wagner and Schmalstieg because it uses natural markers instead of fiducial ones, and because it is possible to find objects until approximately 1 meter away from the marker. However, the detection of natural markers is very slow, so out of the interest island, real-time performance cannot be achieved. In addition, this approach requires many markers; otherwise it lacks tracking when the camera is not pointing to an island. Finally, the last paper from Castle et al. [3] proposes a solution similar to our work, but they use an inertial measurement unit (IMU) and two cameras, where one is a pan-tilt camera and other is normal. Our work uses just a normal webcamera in order to reduce the system requirements.

3. PROPOSED ALGORITHM

The proposed algorithm attempts to reach a balance between two different strategies to compute the camera localization as it moves in the interest area. The first strategy is based on the idea presented by Wagner and Schmalstieg [15], which relies on calibration markers placed all over the interest area to recover the camera position. Although, we use natural markers, instead of intrusive fiducial markers. Natural Markers are regions of the environment, which have enough texture to be considered unique. The second strategy is based on the SLAM algorithm, which performs simultaneous camera localization and feature mapping from a single known marker in the area. The strategy proposed here uses only a few natural markers in the area to serve as spots for the automatic reinitialization of the SLAM algorithm. The identification of markers occurs in parallel with the updating process of SLAM, so when a marker is identified and it is in a position that yields good calibration results, the SLAM algorithm is reinitialized to eliminate the accumulated error.

Figure 2 illustrates the basic idea of the proposed algorithm, which works with two parallel threads. The main thread is responsible for the following processes: (a) image capture; (b) update of the VSLAM algorithm; and (c) rendering of the captured image with additional virtual information (application layer). This thread needs to be fast enough for interactive time (± 20Hz), so it is not feasible to use a natural marker detector on this thread. The second thread is responsible for the Relocalizer. This process is constantly looking for natural markers using the SURF algorithm. If a marker is found, the process consults its database of mapped markers, called Marker Map, to identify where the camera is. The Integrator receives the position of the camera, in real world coordinates, as well as additional information about the marker and the SLAM state to estimate the positioning error of both parallel tracking processes. With all of this information, the Integrator decides whether or not to reinitialize the SLAM algorithm. We will provide details about these sub-algorithms in the following sections.

3.1 Relocalizer

The Relocalizer is responsible for consistently seeking markers in the images captured by the camera. This sub-algorithm uses pattern recognition to localize natural markers. We use

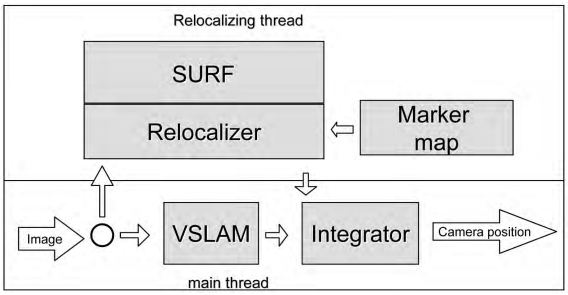

Figure 2: Basic architecture of the proposed algorithm.

the SURF algorithm [1] as the natural marker detector because of its performance, of about 3 fps in our tests, despite being slightly less accurate than SIFT [13], according to Lieberknecht et al. [10]. We chose the CPU version of SURF because our algorithm is designed to run on a high-performance tablet PC, which normally does not have a good GPU. When a marker is found, the detector calculates the local position of the camera relative to the marker. To calculate the global position of the camera, the Relocalizer queries the global position of the marker in Marker Map and then converts the camera position from the local coordinate system to the global one.

The SURF algorithm may have its performance affected if the group of natural markers is very large, because it searches by comparing the keypoints of the image captured by the camera with the keypoints of each possible marker. To reach 3 fps, it is necessary to maintain the number of possible markers around 20. Therefore, we reduced the group of possible markers to the 20 markers closest to the last known position of the camera. This way, the search is delayed only in the initial frame, when there is no previous position and when the camera moves too much while the tracking is lost, because the estimation of the last known position may be useless.

The Marker Map is filled during the pre-processing phase with the positions of the markers in the global coordinate system. It also stores the points to be tracked by SLAM. This is necessary because the feature-point detector of SURF is different from the detector of the SLAM algorithm used in this article. For this reason we cannot use the keypoints of natural markers as features to be tracked by SLAM. To automatically select points that are good to be tracked by SLAM, the detector of the SLAM algorithm first detects features. Then the nearest point to each of the four corners of the image is chosen from among the 10% best features. Figure 3 shows the points selected for the natural markers, which are selected automatically.

This work does not focus on how to perform the global mapping of markers. We used laser measuring equipment called Total Station to perform our tests due to its accuracy. However, the measures could also be made using various computer vision methods to recover the geometry of the environment, including a SLAM algorithm with loop-closure refinement [16] and others using several markers [9].

When a natural marker is found, there is a synchronization problem with SURF because it does not work in real time, as explained above. To solve this issue, we propose a modi-

Figure 3: Configuration of the measured points in natural markers.

fication of the original SLAM [5] algorithm to minimize the synchronization problem. This modification will be detailed in the next section, followed by a section that explains how to automatically reinitialize the algorithm proposed here.

3.2 VSLAM

Theoretically, the algorithm proposed here is not related to any specific VSLAM algorithm, because all VSLAM algorithms have the same behavior. They determine the self-location in a position known to the camera/robot, and build a dynamic map to be capable of updating the self-location from positions around the initial position as well. In the present work, we need a SLAM algorithm capable of self-localizing from a camera pose, like the Relocalizer does. To accomplish this, we use a modified version of the SLAM algorithm proposed by Davison [5]. The original version of this algorithm does not have an initialization process . The Davison approach supposes that the camera will be initialized only once in front of a known black rectangle. Our modification of this algorithm gives it the ability to be initialized by markers while moving.

Davison's algorithm is based on an Extended Kalman Filter (EKF). Below, we describe the mathematical model used by Davison and then explain how to initialize the values of the model starting from data supplied by a marker. We use the same notation as [5]. The mathematical model of the EKF has two structures that store data: the state vector $\hat{x}$ and the covariance matrix P. The state vector $\hat{x}$ is composed of the sub-vector $\hat{x}_v$, which stores the camera data, and of other sub-vectors $\hat{y}_i$, each one representing a 3D position of the i feature point on the SLAM map. The covariance matrix P is composed of covariance sub-matrices P_{ab}, which represent the covariance between sub-vectors a and b. The components of the camera data($\hat{x}_v$) are: r^W, the 3D position of the camera in the real world; q^{WR}, the camera orientation quaternion ; v^W, the velocity vector; and ω^R, the angular velocity vector. The notation W means that the parameter is in the global coordinate system, while R means that it is relative to the local camera system.

$$\hat{x} = \begin{pmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \vdots \end{pmatrix}, P = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \cdots \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \cdots \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$x_v = \begin{pmatrix} \mathrm{r}^W \\ \mathrm{q}^{WR} \\ \mathrm{v}^W \\ \omega^R \end{pmatrix}$$

In order to initialize SLAM, the Relocalizer provides the 3D positions of the feature points to be tracked ($\breve{y}_i$), the camera position in global coordinates ($\breve{\mathrm{r}}^W$), its orientation ($\breve{\mathrm{q}}^{W\breve{R}}$), and the time elapsed between the image capture and the end of the marker detection ($\breve{\Delta}\mathrm{t}$). We assume that the camera movement remains unaltered during the time interval needed for the marker detection. Then, we update the camera parameters calculated by the detector as shown below. We use the following notation: matrix M^{VY} converts from a coordinate system Y to a coordinate system V; $\breve{\mathrm{R}}$ is the local coordinate system of the new camera position; $q(\omega^R\breve{\Delta}t)$ is the quaternion corresponding to the rotation axis $\omega^R\breve{\Delta}t$; and O is a zero sub-matrix.

$$x_v = \begin{pmatrix} \mathrm{r}^W_{new} \\ \mathrm{q}^{WR}_{new} \\ \mathrm{v}^W_{new} \\ \omega^R_{new} \end{pmatrix} = \begin{pmatrix} \breve{\mathrm{r}}^{\mathrm{W}} + (\mathrm{M}^{\mathrm{W}\breve{\mathrm{R}}}\mathrm{M}^{\mathrm{RW}}\mathrm{v}^{\mathrm{W}})\breve{\Delta}\mathrm{t} \\ \breve{\mathrm{q}}^{\mathrm{W}\breve{\mathrm{R}}} \times \mathrm{q}(\omega^{\mathrm{R}}\breve{\Delta}\mathrm{t}) \\ \mathrm{v}^{\mathrm{W}} \\ \omega^R \end{pmatrix}$$

To update the covariance matrix in the reinitialization, we need to fill it with zeros, because the map is filled only with trusted features from the Marker Map. The camera position is an exception, because the camera calibration given by the Relocalizer implies some uncertainty. Then P_{xx} needs to handle the uncertainty associated to each component of the 3D camera position. We assume that the uncertainties are independent from each other. As a result, matrix P_{xx} is formed as shown below. We empirically choose the parameter γ as 0.001 for both marker detectors.

$$P = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{O} & \mathrm{O} & \cdots \\ \mathrm{O} & \mathrm{O} & \mathrm{O} & \cdots \\ \mathrm{O} & \mathrm{O} & \mathrm{O} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\mathrm{P}_{xx} = \begin{bmatrix} \gamma & 0 & 0 & 0 & \cdots \\ 0 & \gamma & 0 & 0 & \cdots \\ 0 & 0 & \gamma & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

3.3 Integrator

The Integrator sub-algorithm is responsible for deciding when it is possible to reinitialize the SLAM algorithm. Whenever it is possible, it executes the reinitialization of the SLAM algorithm, verifies the camera position in the state of the Kalman filter, and then forwards it to the visualization. When it is not possible to reinitialize SLAM, the Integrator simply verifies the camera position and forwards it.

The reinitialization must occur either when the SLAM algorithm is lost or when the uncertainty of the camera position calculated by SLAM is larger than that of the detected marker. However, there is no objective way of calculating the uncertainty of the marker detection, although there are several works that measure the precision of these kinds of marker detectors [11, 12, 7]. Based on these works we defined a heuristic to determine when the reinitialization must

be executed. In this heuristic, we defined the minimal conditions to enable the SLAM algorithm to track the scene after the marker leaves the image.

When a marker leaves the camera image, other points must already be in the SLAM map, so that SLAM can keep calculating the camera position. For this reason, if the reinitialization is attempted with a marker very close to the image border, when the camera moves to make the triangulation and to calculate the 3D position of the new points, there is a high possibility that the marker will be already out of the image. To avoid this, we empirically defined 15% of the image at each border as the limit for the marker. If a marker crosses this limit, it is no longer considered for tracking reinitialization. Another way for a marker to leave the image is by zooming in or out on a point to an extent that it can no longer be tracked. To avoid this, we defined that the maximum area of the image that a marker may occupy is 90%, and the minimum is 10%. If a marker is out of these limits, it is not considered for reinitialization. These values depend upon the camera lens. We use a lens with 2.1mm of focal distance.

In the next section, we present tests of our method against the reference work.

4. PRATICAL EXPERIMENTS

In this section, we analyze the improvements using the low precision constraints compared with Wagner and Schmalstieg's [15] indoor localization approach. We also show the importance of reinitialization in a SLAM algorithm. Then, we are going to evaluate the reprojection errors of points from an automatic reinitialization following the empirical values explained in the section 3.3 constraints. The tests examine the accuracy with which our algorithm indicates objects in the scene without any marker in the image. For the tests, we used a 2.20 GHz dual-core computer and a firewire webcam with resolution of 320x240 and 30fps.

4.1 Controlled environment test

This experiment aims to analyze the idea of supplementing tracking algorithms with marker-based SLAM. In addition, We will evaluate the error in estimating the position of the camera. Therefore, the experiment will reproduce the scenario that was used in the work of Wagner and Schmalstieg. Also, we replace the algorithm for tracking natural markers by a fiducial marker tracking algorithm (ARToolKitPlus) equivalent to that used by Wagner and Schmalstieg in order to make a direct comparison. Then, we can use the same video to run the original algorithm of Wagner and Schmalstieg and our modified algorithm. In the next experiment, we will analyze the influence of the natural marker tracking in the quality of SLAM tracking, which is the step that was removed from this experiment. In the following sections, we show the setup of the experiment, the testing procedures and the analysis of the measurements.

4.1.1 Setup

Our test emulates a straight walk into a building. We built a camera dolly system to ensure that the path of the camera would be known, as shown in Figure 4. On the wall, there are two fiducial markers 3.5 meters apart. The wall measures about 4 meters. The markers and the dolly path were mapped using Total Station.In addition, we randomly fixed pieces of paper on the wall in order to give texture to our white wall. The parameters used in the Davison's SLAM are the same used in the original paper.

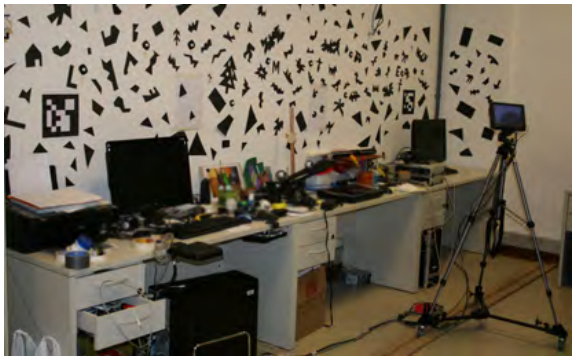

Figure 4: Test station for low precision mode. There are two markers on the wall. The camera and a portable monitor were placed on a tripod which is on top of a dolly (camera cart).

4.1.2 Procedures

The dolly system was pushed manually, trying to keep a constant speed, parallel to the wall at which the camera was pointed. The camera was panned/rotated 45 degrees in relation to the movement direction. We recorded and processed the same video following Wagner and Schmalstieg's [15] approach and our approach. The resulting graphs can be seen in Figures 5,6,7 and 8.

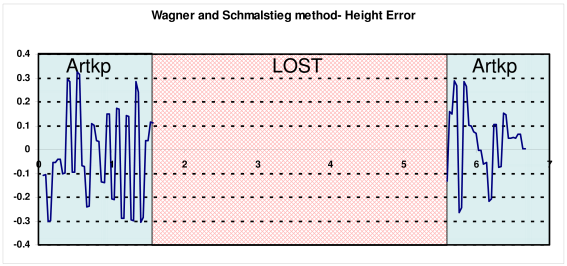

Figure 5: Graph of error (meter) vs time(second) using Wagner and Schmalstieg's [15] approach (Artkp means ARToolKitPlus)

4.1.3 Evaluation

Comparing Figure 5 and Figure 6, we can make the following observations. Wagner and Schmalstieg's method does not support such large distances, but our algorithm does. Our method uses the ARToolKitPlus just a few seconds less to be capable of swapping to SLAM mode. The camera height is more stable in SLAM mode. Moreover, the depth error (Figure 7) increases constantly in SLAM mode, but when a new marker appears the error returns to the level of ARToolKitPlus. This shows the importance of the SLAM reinitialization. However, it only works in well-textured environments, like shown in [5, 6].Figure 10 shows the distance traveled. Because the dolly was hand pushed, we don't have the ground truth. However, if we consider the precision as

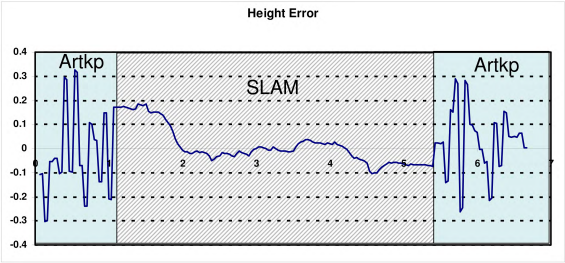

Figure 6: Graph of camera height error (meter) vs time(second) using our algorithm

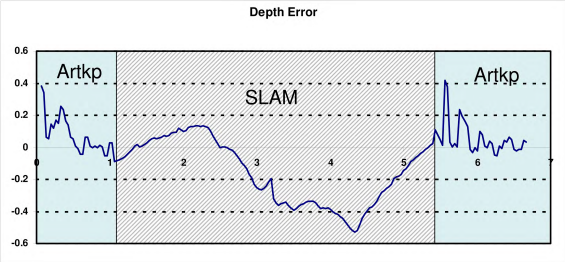

Figure 7: Graph of camera depth error (meter) vs time(second) using our algorithm

measured in [11], then we can observe the error when the second marker is found the traveled distance jumps 1.2 meters whereas the jump expected by the article [11] is a maximum of 10 cm. So we conclude that the SLAM algorithm evaluates a motion slower than the real motion.

4.2 Initialization accuracy of SLAM with SURF

This experiment aims to evaluate the accuracy expected in the camera position estimated from our SLAM algorithm initialized with some examples of natural markers in different positions. In this experiment, we run a unique initialization using the heuristic explained in section 3.3 over each natural marker. Next, we will show the setup of the experiment, the testing procedures and the analysis of the measurements.

4.2.1 Setup

We prepared the test station as shown in Figure 9. We spread eleven distinct objects on a 1.2m x 0.6m table. Spher-

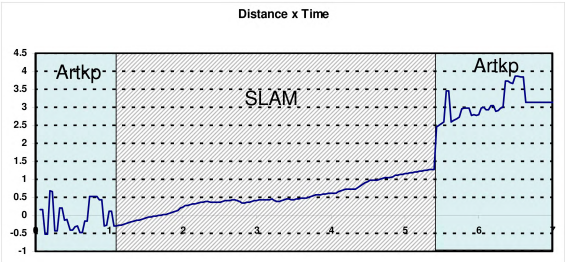

Figure 8: Graph of traveled distance (meter) vs time(second) using our algorithm

Figure 9: Test station

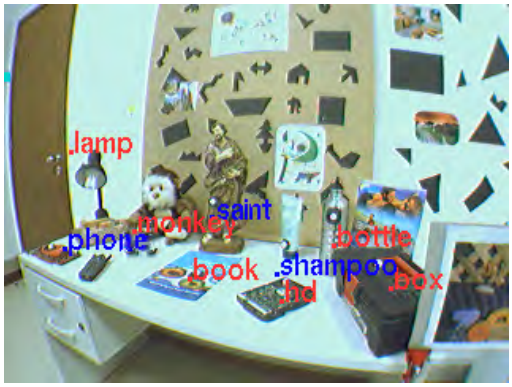

Figure 10: Reprojection error of maker b

ical targets were fixed on each object (the white dots over the objects in Figure 9). These targets represent the position of the objects. We also set six natural markers at different distances (**A**: 0.4m on the left, **B**: 0.4m on the right, **C**: 0.8m on top, **D**: 0.8m on the right, **E**: 1.0m and **F**: 2.0m) from the center of the table. The marker **A** has a rectangular black frame to help with the tracking. Markers **D** and **F** have small black rectangles at the corners, which work like the black frame but are less occlusive. The other markers have rounded corners in order to simulate the textures of non-rectangular objects. Almost on the center of the table, a fiducial marker was placed. The markers are identified by dotted ellipses in Figure 9.

4.2.2 Procedures

Regardless of the marker, we set the Marker Map database to include only the marker we want and 19 other markers out of the scene, in order to simulate the real conditions of the detection of natural markers. Then we pointed the camera at the marker and waited for the automatic initialization, and then walked towards the table. In front of the table, we moved the camera slightly in various directions to do the tracking from different viewpoints. Then we calculated the reprojection error of the sphere targets. The results are presented below. We did not test ARToolKitPlus markers because many other works have done this.

4.2.3 Evaluation

A typical result is shown in Figure 10 and table 1 presents average values as well as the standard deviation of the reprojection error. These values are the Euclidean distance

between the position of the sphere in the image and the projected point. We considered only the frames where the camera was in front of the table. The tracking from marker **F** presented such a large error at the camera position that no projected point appeared in the image.

distance	mean error	standard deviation
A	10.4	3.2
B	14.1	4.3
C	108.3	30.8
D	13.3	3.7
E	178.7	26.1

Table 1: Average reprojection error (in pixels) and standard deviation.

We can see that, for distances of approximately 2 meters, the tracking achieves values close to those of the last section experiment. Moreover, the markers without the black rectangles or frames generated worse maps than those that had this feature. The marker **B**, despite the lack of this feature, had a good reprojection, but the tracking was lost after a short time in front of the table because its map had fewer features to be tracked. In our analysis, the use of these artificial black objects increases the tracking robustness because they are more scale-invariant, and this is important for enabling the camera to move back and capture more points after the initialization.

5. CONCLUSIONS

This paper presented a tracking approach for non-instrumented environments that uses SLAM to track non-marked areas. However, different from conventional SLAM approaches, in which the errors grow exponentially from an initial well-known position and require corrections in non-real time, our approach uses a pattern-recognition method to detect natural markers used to reinitialize the SLAM algorithm, eliminating the accumulated error at that moment. To enable the use of a non-real time detector of natural markers, we defined a multi-thread architecture and two heuristics that determine the cases in which a marker can be considered for SLAM reinitialization. The results presented have shown that the reprojection error can be small enough to track the camera using our algorithm. Also, it tracks the camera between markers, allowing a greater distance between them.

Further experiments are necessary to analyze the error of the camera on the motion axis. Moreover, we intend to study how to add spatial information from the building in order to restrict the possible position of the camera. Another interesting possibility is to add a WIFI based localization to give an approximation of the localization. With this additional information the candidate set of the natural markers will be smaller and it will also be easier to map an individual natural marker because we will only be required to map it in a small area, and the ambiguity will be resolved by the approximation of the camera localization given by the WIFI signal.

6. REFERENCES

[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[2] R. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proceedings of the 12th IEEE International Symposium on Wearable Computers*, 2008.

[3] R. O. Castle, G. Klein, and D. W. Murray. Combining monoslam with object recognition for scene augmentation using a wearable camera. *Image Vision Comput.*, 28(11):1548 – 1556, 2010.

[4] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems*, 2007.

[5] A. J. Davison. Real–time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1403–1410, 2003.

[6] E. Eade and T. Drummond. Monocular slam as a graph of coalesced observations. In *IEEE 11th International Conference on Computer Vision*, 2007.

[7] S. Gauglitz, T. Hollerer, P. Krahwinkler, and J. Rossmann. A setup for evaluating detectors and descriptors for visual tracking. *IEEE / ACM International Symposium on Mixed and Augmented Reality*, pages 185–186, 2009.

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

[9] M. Klopschitz and D. Schmalstieg. Automatic reconstruction of wide-area fiducial marker models. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[10] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab. A dataset and evaluation methodology for template-based tracking algorithms. In *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 145–151, 2009.

[11] P. Malbezin, W. Piekarski, and B. H. Thomas. Measuring artoolkit accuracy in long distance tracking experiments. In *In the First IEEE International Augmented Reality Toolkit Workshop*, 2002.

[12] K. Pentenrieder, P. Meier, G. Klinker, and M. Gmbh. Analysis of tracking accuracy for single-camera square-marker-based tracking. *Dritter Workshop Virtuelle und Erweiterte Realitt der GI-Fachgruppe VR/AR*, 2006.

[13] F. Schweiger, B. Zeisl, P. Georgel, G. Schroth, E. Steinbach, and N. Navab. Maximum detector response markers for SIFT and SURF. In *Vision, Modeling and Visualization Workshop (VMV).*

[14] B. Schwerdtfeger, , and G. Klinker. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality.*

[15] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, page 127, 2003.

[16] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.