A VISION-BASED HYBRID SYSTEM FOR REAL-TIME ACCURATE LOCALIZATION IN AN INDOOR ENVIRONMENT

Vincent Gay-Bellile, Sylvie Naudet Collette, Romain Dupont, Mohamed Tamaazousti
CEA, LIST, Laboratoire Vision et Ingénierie des Contenus, Point Courrier 94, Gif-sur-Yvette, F-91191 France;
vincent.gay-bellile@cea.fr

Keywords: Human Localization, Indoor Environment, Real-Time, monocular vision.

Abstract: This paper presents an indoor vision-based system using a single camera for human localization. Without *a priori* knowledge of the operating environment, a map has to be built on-line to estimate the relative positions of the camera. When a model is *a priori* known, only the camera poses are computed. It results in distinctive algorithms which have both assets and drawbacks. Localization in an unknown environment is much more flexible but subject to drift while localization in a known environment is almost drift-less but suffer from recognition failures. We propose a new approach to localize a camera in an indoor environment. It combines both techniques described above benefiting from the knowledge of Georeferencing information to reduce the drift (comparatively to localization in unknown environment) while avoiding the user to be lost during long time intervals. Experimental results show the efficiency of our method.

1 INTRODUCTION

Human localization in an Indoor Environment is a very challenging issue. Indeed, GPS sensor (Brusningham et al., 1989) is indoor ineffective and useful information, used for vehicles or robots localization, such as odometry (OKane, 2006) or nonholonomic motion model (Scaramuzza et al., 2009) can not be exploited. Other technologies such as WI-FI (Ocana et al., 2005) or RFID (Hahnel et al., 2004) permit to overcome such limitation but they require environment instrumentation through installation of specific equipments in the building. For our targeted application, *i.e.* localization of security troops in a building, a mobile wearable technology equipped with a single camera appears to be a solution with higher acceptability.

This paper proposes a purely vision-based system for human localization in an indoor environment. Vision-based localization algorithms can be classified into two major types: Localization in Known and UnKnown Environments. Thereafter, we use the acronyms LKE and LUKE respectively.

The former exploits a Georeferencing Structure from Motion point cloud that is built and stored off-line, see *e.g.* (Arnold et al., 2009; Schindler et al., 2007). Interest points, extracted from the images, are matched with the 3D points of the database: the camera poses are then computing through robust linear and non linear optimizations. LKE algorithms suffer from recognition failures mainly due to appearance variations since the database has been learned. They are introduced by illumination changes (in the windows and lighting source neighborhoods for indoor environments), furniture relocation, *etc*. It results in many unrecognized areas.

On the other hand LUKE algorithms build a map (i.e. a 3D point cloud) on the fly, i.e. along with the camera localization (Davison, 2003; Mouragnon et al., 2006). They appear to be more generic since no *a priori* information is necessary but suffer from drift due to accumulation of errors (building the map depends on the camera pose and inversely) and from the scale ambiguity of monocular vision algorithms.

An hybrid system that constitutes the best of two worlds is described in this paper. The proposed idea is to progressively correct the camera poses returned by the LUKE algorithm when parts of the model are recognized in the images through the LKE algorithm. It presents several assets:

- **compare to LUKE**: the drift is limited.
- **compare to LKE**: the user is not lost when recognition fails. An other advantage is that the environment must not be fully learned anymore. This is very useful for an embedded system since the memory space takes up by the database can be drastically reduced.

Finally, combining these two algorithms may be time consuming. Real-time processing is maintained through parallel computing and a specific thread management.

Roadmap. Details on LUKE and LKE algorithms are given in §2 and in §3 respectively. In §4, we describe how these algorithms are combined together. Experimental results on real data are reported in §5. Finally, we give our conclusions and discuss further work in §6.

2 LOCALIZATION IN AN UNKNOWN ENVIRONMENT

LUKE algorithms are used when no *a priori* on the observed environment is available. The environment and the trajectory of the moving camera are simultaneously reconstructed from a video. The main drawback of LUKE algorithms is the unavoidable drift due to accumulation of errors and the scale ambiguity. Various approaches have been proposed for real-time localization in an unknown environment. They can be classified into two major types: local bundle adjustment (Mouragnon et al., 2006; Nister et al., 2004) and Kalman filter (Davison, 2003) based algorithms. We rather used the former approach since it has been proved to be more accurate, see (Klein and Murray, 2007). We describe briefly the solution proposed in (Mouragnon et al., 2006) which is used in our experiments. A triplet of images is firstly selected to set up the world coordinate frames and the initial geometry. After this initialization, robust pose estimation is carried out for each frame of the video using points detection and matching. Note that in our experiments, we used Harris corners (Harris and Stephens, 1988) detector and SURF descriptors (Bay et al., 2008). A crucial point described in (Mouragnon et al., 2006) is that 3D points are not reconstructed for all the frames. Specific ones are selected as key-frames and are used for triangulation. A key-frame is chosen when the motion is sufficiently large to accurately compute the 3D positions of matched points but not too much to keep matching. The system operates in an incremental way, and when a new key-frame and 3D points are added, it proceeds to a local bundle adjustment: cameras associated to the latest key-frames[1] and 3D points they observed are updated. This algorithm is summarized in figure 1 (Thread # 1).

3 LOCALIZATION IN A KNOWN ENVIRONMENT

LKE algorithms are used when *a priori* on the observed environment is available. We concentrate on algorithms which use a 3D point cloud as model. It is built through a learning stage that associates accurate 3D positions to images covering the considered environment. Each image is also resumed in a set of interest points with their descriptors (about 100-500 points with their SURF descriptors for each image) and their corresponding 3D points in the scene. All this information is saved in a database. The online localization process consists in comparing the observed image of the scene to all images of the database using their descriptors. The most similar image, i.e. with the highest correlation score, should correspond to the currently viewed scene. The camera pose is then computed using the 3D points observed in this image. As the covered environment grows, it becomes impossible to compare, in a systematic way, the query image with all images of the database. Therefore, a vocabulary tree structure is used to speed-up this retrieval step. This structure has proved to be very efficient even for very large database (more than 100000 images) (Arnold et al., 2009; Nister and .Stewenius, 2006; Schindler et al., 2007). It is a hierarchical tree (with branching factor k and l levels) storing descriptors by similarity in such a way that an exhaustive search can be done in only $k * l$ descriptor comparisons (done with the L1-distance). Therefore, this permits a quick comparison between a descriptor from the query image and the whole set of descriptors of the database. In detail, for each query image, we extract about 400 interest points with their SURF descriptors. Our vocabulary tree has 6 levels and a branching factor of 10. Hence, for each descriptor of the query image, only 60 comparisons are computed.

[1] the three latest key frames are updated in (Mouragnon et al., 2006).

4 COMBINING LOCALIZATION IN KNOWN AND UNKNOWN ENVIRONMENTS

4.1 Overview

In this section, we describe how the algorithms presented above are combined together. The idea is to progressively correct the drift of the LUKE algorithm when parts of the database are recognized in the images through the LKE algorithm. Fusion of LUKE and LKE algorithms is difficult. The latter may not provide data for long time interval due to unlearned areas, lighting variation, *etc*. LUKE algorithm may have drift too much in the meantime yielding in inconsistent data. Fusion through Kalman filter requires data covariance. However, methods that compute the covariance of local bundle adjustment such as (Eudes and Lhuillier, 2009) do not take into account the scale factor drift of monocular LUKE algorithm. The covariance are then underestimate and deconfliction problems (Mittu and Segaria, 2000) will probably appear.

We propose an alternative approach. Data fusion is achieved through a drift correction module that adjusts the LUKE algorithm history *i.e.* camera poses and 3D points used in the local bundle adjustment only. This additional module compute a Similarity between the poses returned by the LUKE and LKE algorithms, as described in §4.2. This fusion process is reasonable if the poses returned by the LKE algorithm are accurate. This is not always true in practice due to matching errors. We add a decision module to control the output of the LKE algorithm. It is composed of a cascade of filters such as epipolar geometry, temporal filtering and quality of tracking (defined by the fraction of inliers when computing the pose through RANSAC). If one of these filters gives a negative answer, the pose is rejected and the drift correction not performed. This ensures a false-positive rate tending towards zero.

Finally, the LUKE and LKE algorithms are performed on two different cores in parallel to keep processing time consistent with real-time constraints. We also propose in §4.3 a specific thread management. It allows maintaining the same frequency than the LUKE algorithm alone. An overview of our approach is given in Figure 1.

4.2 Drift Correction Module

This additional module computes a Similarity $\mathsf{W}(\mathsf{R}_{\mathcal{W}}, \mathbf{T}_{\mathcal{W}}, s_{\mathcal{W}})$ between the poses returned by the

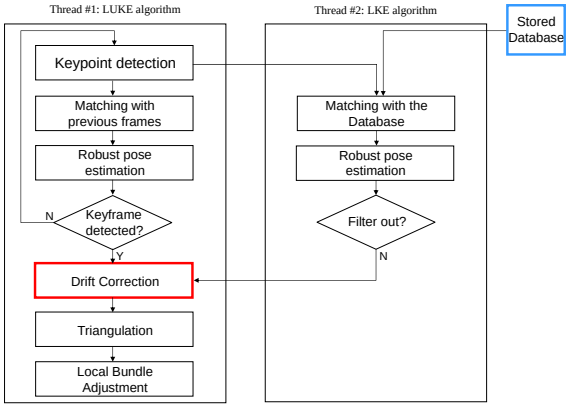

Figure 1: Overview of the processing for one input frame. A drift correction module is added in the LUKE algorithm. It takes the output of the LKE algorithm to compute a Similarity. LUKE and LKE algorithms are performed on two different cores in parallel.

LUKE and LKE algorithms. A Similarity is required due to the drift introduced by the scale ambiguity, a simple Euclidian transformation is not enough.

The Similarity is given by:

$$\begin{aligned} \mathsf{R}_{\mathcal{W}} &= \mathsf{R}_{\text{LKE}}\mathsf{R}_{\text{LUKE}}^{\mathsf{T}} \\ \mathbf{T}_{\mathcal{W}} &= -\mathsf{R}_{\text{LKE}}\mathsf{R}_{\text{LUKE}}^{\mathsf{T}}\mathbf{T}_{\text{LUKE}} + \mathbf{T}_{\text{LKE}} \\ s_{\mathcal{W}} &= \frac{\mathbf{T}_{\text{LKE}} - \mathbf{T}_{\text{LKE}}^{u}}{\mathbf{T}_{\text{LUKE}} - \mathbf{T}_{\text{LUKE}}^{u}}, \end{aligned}$$

where $(\mathsf{R}_{\text{LKE}}, \mathbf{T}_{\text{LKE}})$ and $(\mathsf{R}_{\text{LUKE}}, \mathbf{T}_{\text{LUKE}})$ are the current poses returned by the LKE and LUKE algorithms respectively. $\mathbf{T}_{\text{LKE}}^{u}$ and $\mathbf{T}_{\text{LUKE}}^{u}$ are respectively the positions of the camera returned by the LKE and LUKE algorithms when the scale was lastly updated.

This Simarity is only applied to the cameras and the 3D points considered in the local bundle adjustment[2]. For example, updated a camera pose is given by:

$$\begin{aligned} \mathsf{R}_{\text{LUKE}} &\leftarrow \mathsf{R}_{\mathcal{W}}\mathsf{R}_{\text{LUKE}} \\ \mathbf{T}_{\text{LUKE}} &\leftarrow s_{\mathcal{W}}\mathsf{R}_{\mathcal{W}}(\mathbf{T}_{\text{LUKE}} - \mathbf{T}_{\text{LUKE}}^{u}) + \mathsf{R}_{\mathcal{W}}\mathbf{T}_{\text{LUKE}}^{u} + \mathbf{T}_{\mathcal{W}}. \end{aligned}$$

Time interval between two consecutive localization of the LKE algorithm may be long. Camera poses returned by the LUKE algorithm are then not all corrected. However, for our application, *i.e.* real-time localization of security troupes in a building, we are

[2] As in (Mouragnon et al., 2006) we use the cameras associated to the last three key-frames, the 3D points observed by these cameras and the other cameras observing these 3D points.

only interested in the current position of the Security Officer. It is also essential to propagate the drift correction to accurately estimate his future positions. This is achieved automatically with our fusion scheme during the LUKE process.

4.3 Parallel Computing

As described above, we opt for parallel computing to keep processing time consistent with real-time. The proposed framework described in §4.1 implies the drift correction module to wait the poses from the LUKE and LKE algorithms, as seen on figure 1. The problem is that computing a pose with the LKE algorithm is quite time consuming. The reasons are:

- matching with the database generally requires more comparisons for each descriptor than temporally matching.
- additional processes are essential to verify the consistency of the computed pose.

We propose a specific way to manage the threads based on the key-frame property of (Mouragnon et al., 2006). Our thread management is described below and summarized in table 1. The main steps are the following: the LKE algorithm (thread 2) tries to localize the key-frame n until a new one $(n+1)$ is labeled by the LUKE algorithm (thread 1). If the process of the LKE algorithm is finished before the next key-frame $(n+1)$, thread 2 is ordered to sleep. At key-frame $n+1$, if the LKE algorithm successfully localizes the key-frame n, the correction module computes the Similarity $\mathcal{W}$. Thread 2 is finally waked up and the associated LKE algorithm tries to localize the key-frame $n+1$. Note that if a new key-frame is labeled before localization through LKE algorithm is finished then thread 2 is stopped, the Similarity is not computed on this key-frame.

This procedure allows keeping the frequency of the LUKE algorithm alone. It introduces an unnoticeably time delay in the drift correction since the Similarity computed for key-frame n is applied when key-frame $n+1$ is labeled.

	Thread 1 (LUKE)	Thread 2 (LKE)
Key-frame n (frame j)	Start Thread 2	Localization on key-frame n
frame j+1		Localization on key-frame n
frame j+2		Localization process is finished
frame j+3		Sleep
Key-frame n+1 (frame j+4)	- Localization through LKE algorithm is successful: ⇨ The Similarity is computed and the drift corrected	Sleep
	- Wake up thread 2	Localization on key-frame n+1

Table 1: Thread management. Only the communications with thread 2 are reported for thread 1.

5 EXPERIMENTAL RESULTS

We use a low-cost IEEE1394 GUPPY camera providing gray-level images with 640x480 resolution at 30 frames per second. Two video sequences (sequence 1 and sequence 2) have been acquired along walking trips though different rooms and corridors. The covered distances are about 90m for both sequences. Note that this kind of environment is quite texture-less compared to outdoor urban environment as seen on figure 2. Sequence 1 start in a known environment: the two first localizations of the LKE algorithm are used to fix the scale factor and the coordinate frames. Sequence 2 start in an unknown environment: specific markers are then used to fix the scale factor and the coordinate frame, as in (Davison, 2003).

The database, represented on figure 2, has been acquired one and two month before compare to sequence 2 and 1 repectively. It is composed of 6374 3D points provided by the LUKE algorithm. We use the post-processing procedure described in (Lothe et al., 2009) which exploits a coarse CAD model of the operating environment to correct the reconstruction drift. The "ground truth" of the two trajectories, represented in figures 3 and 4 (top left), are obtained with the same procedure. Note that they differ from the trajectory follow for buidling the database: several offices have not been learned. Moreover, illumination conditions have drastically changed in the meantime (cloudy vs sunny weather, light on vs off) and objects have disappeared and sometimes been substituted.

Figure 3 shows the results obtained with the LUKE algorithm (bottom right), with the LKE algorithm (top right) and with the one we propose (bottom left) on sequence 1. The former is subject to accumulation of errors. Indeed, at point R *i.e.* where the drift reaches its peak, the error in position is approximately 7 meters. On the other hand, some areas are not localized with a LKE algorithm. Recognition fails in region 2 due to lighting variations in the windows surroundings and in region 3 since notice boards have changed in the meantime[3]. Recognition is also im-

[3]Regions 1, 2 and 3 are illustrated on figure 2.

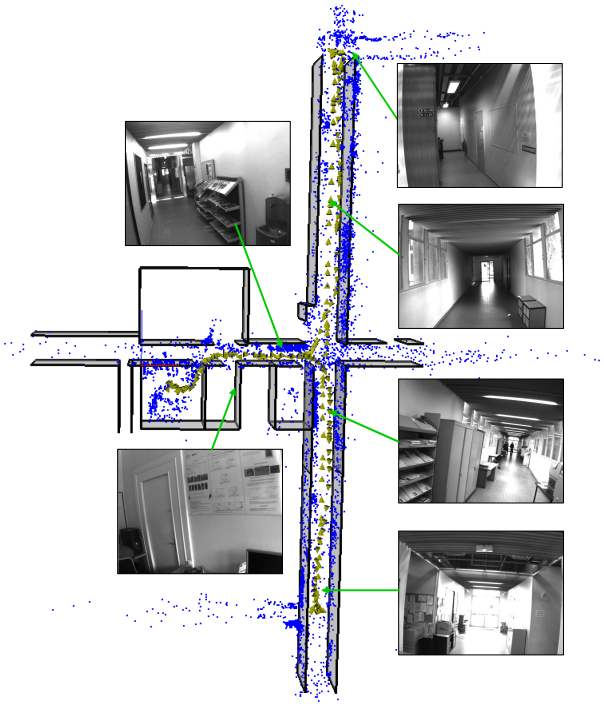

Figure 2: The database used for the LKE algorithm. This Structure from Motion point cloud comes from the LUKE algorithm. The drift is *a posteriori* corrected with the procedure described in (Lothe et al., 2009).

possible in unlearned areas such as region 1. The proposed system compensates for the drawbacks of the two above algorithms. It results in a "dense" localization with a limited drift. The obtained trajectory is closed to the ground truth since the maximum error in position is around 1 meter. Same conclusions are obtained on sequence 2. LUKE algorithm suffers from an important drift in the trajectory estimation while the localization returned by the LKE algorithm is very sparse due to several unknown areas (region 1). The proposed hybrid algorithm yields in a trajectory very closed to the ground trouth.

Finally, parallel computing with our specific thread management allows keeping the same frame rate as the LUKE algorithm alone. The mean frame rate on the two sequences is approximately 39 fps on a Pentium IV dual-core 3GHz.

6 CONCLUSION AND FURTHER WORK

We have presented a new system for indoor localization using two reliable and complementary methods: the absolute approach (LKE) and the relative one (LUKE). Experimental results have shown its ability to provide real-time accurate user localization in indoor environment. Further Work will studied the fusion with others sensor such as inertial ones. This additional information is useful for both LUKE and LKE algorithms. It has been proved to improve the robustness of LUKE algorithm and may help to filter erroneous localization of LKE algorithms.

ACKNOWLEDGEMENTS

This work has been conducted as part of the ITEA2 EASY Interactions project and partially funded by the French industry government.

REFERENCES

Arnold, I., Zach, C., Frahm, J., and Horst, B. (2009). From structure-from-motion point clouds to fast location recognition. In *CVPR*, Miami.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. (2008). Surf: Speeded up robust features. *CVIU*, 110(3):346–359.

Brusningham, D., Strauss, M., Floyd, J., and Wheeler, B. (1989). Orientation aid implementing the global positioning system. In *NEBEC*, Boston.

Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, Nice.

Eudes, A. and Lhuillier, M. (2009). Error propagations for local bundle adjustment. In *CVPR*, Miami.

Hahnel, D., Burgard, W., Fox, D., Fishkin, K., and Philipose, M. (2004). Mapping and localization with rfid technology. In *ICRA*, New Orleans.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *AVC*, Manchester.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *ISMAR*, Nara.

Lothe, P., Bourgeois, S., Dekeyser, F., Royer, E., and Dhome, M. (2009). Towards geographical referencing of monocular slam reconstruction using 3d city models: Application to real-time accurate vision-based localization. In *CVPR*, Miami.

Mittu, R. and Segaria, F. (2000). Common operational picture (cop) and common tactical picture (ctp) management via a consistent networked information stream (cnis). In *ICCRTS*.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real-time localization and 3d reconstruction. In *CVPR*, New-York.

Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *CVPR*, Washington.

Nister, D. and .Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*, New-York.

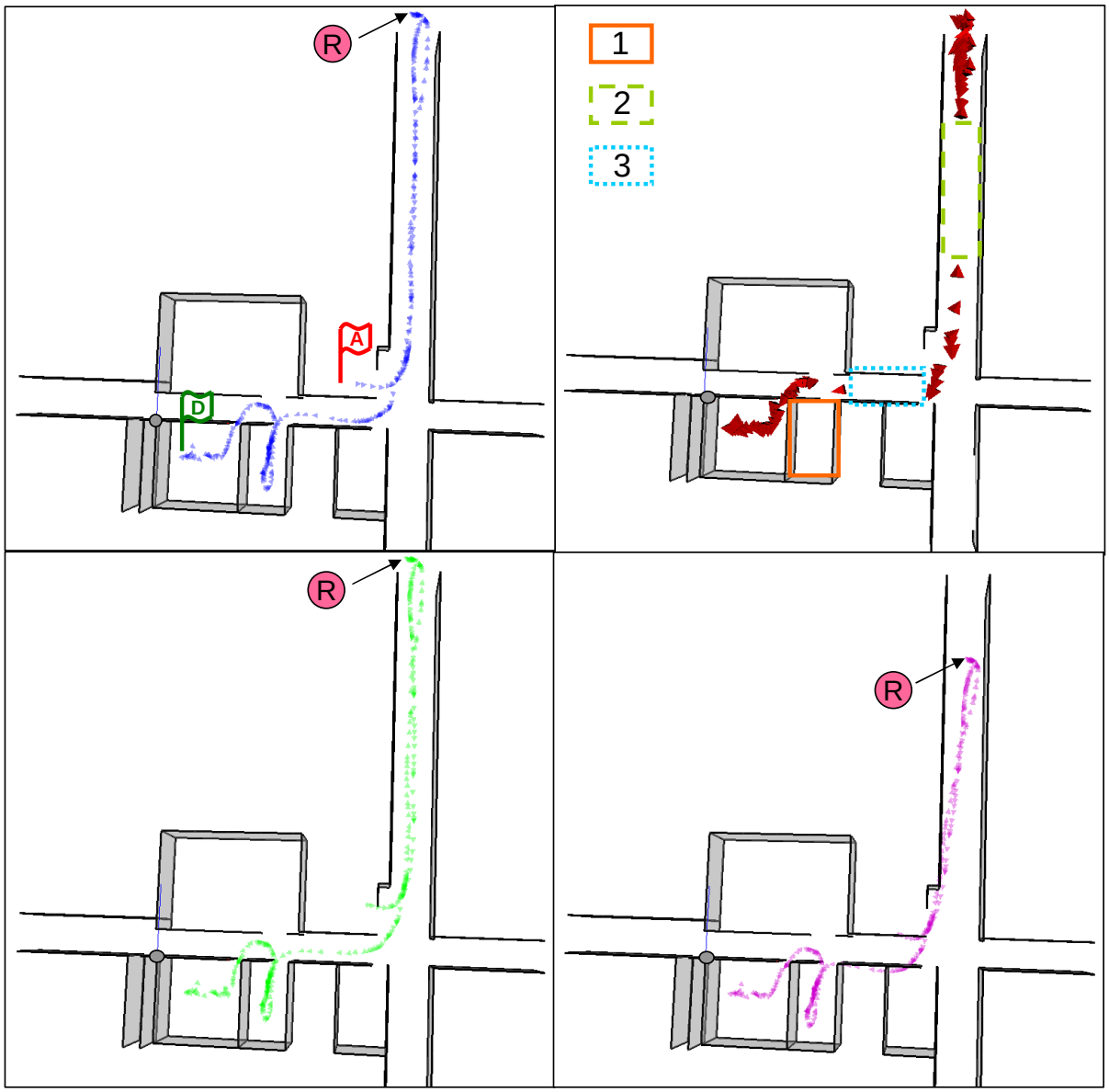

Figure 3: (top left) The trajectory follow for the first sequence: the green flag and the red flag represent the departure and the arrival respectively. This "ground truth" visually helps to have an idea about the drift and to roughly quantify them. (top right) The trajectory obtained with a LKE algorithm. (bottom left) The trajectory obtained with the proposed algorithm. (bottom right) The trajectory obtained with the LUKE algorithm described in (Mouragnon et al., 2006).

Figure 4: (top left) The trajectory follow for the second sequence (”ground truth”): the green flag and the red flag represent the departure and the arrival respectively. (top right) The trajectory obtained with a LKE algorithm. The two first poses, estimated with the specific markers, are also represented (bottom left). The trajectory obtained with the proposed algorithm. (bottom right) The trajectory obtained with the LUKE algorithm described in (Mouragnon et al., 2006).

Ocana, M., Bergasa, L., Sotelo, M., Nuevo, J., and Flores, R. (2005). Indoor robot localization system using wifi signal measure and minimizing calibration effort. In *ISIE*, Dubrovnik.

OKane, J. (2006). Global localization using odometry. In *ICRA*, Orlando.

Scaramuzza, D., Fraundorfer, F., Pollefeys, M., and Siegwart, R. (2009). Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *ICCV*, Kyoto.

Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. In *CVPR*, Minneapolis.