Visual Indoor Localization in Known Environments

Claudio Piciarelli, *Member, IEEE*

Abstract—**In this letter, we propose a visual indoor localization technique, which localizes a camera sensor by comparing the acquired images to a reference model of location-tagged visual features. The proposed method relies on an efficient way to search for feature matches, which can run in real time. Experimental results show good localization accuracy even in challenging scenarios.**

Index Terms—**Image sequence analysis, robot vision systems.**

I. Introduction

Outdoor localization is nowadays a solved problem, thanks to standard technologies such as GPS. However, the same approach cannot be adopted in indoor scenarios, because of the GPS signal attenuation in closed environments. Despite of this limit, there is an increasing need for robust indoor localization systems. A large number of practical applications could benefit from such systems, e.g., guided tours in museums, helper apps for students in large school campuses, finding the path to a specific shop in a shopping mall, and in general any context where the users can benefit from positional information inside large buildings, such as hospitals, airports, factories, etc.

Many approaches to indoor localization exist, relying on different sensing technologies such as triangulation based on WiFi signals or proximity detection using RFID tags. All these approaches, however, require a proper hardware infrastructure to support the localization task. In contrast, visual-based approaches try to estimate the position of a visual sensor (such as a smartphone camera) by comparing the acquired image with a known model of the world. However, the task is challenging since the amount of reference data could be too large to be processed in real time.

In this letter, we propose an efficient scheme to perform real-time visual indoor localization. The main idea is to exploit the temporal coherence of video sequences to track visual features, thus greatly reducing the amount of data to be processed in the matching phase.

II. Related Work

Several approaches have been proposed in literature to address the problem of localization, either indoor or outdoor, however, they can be grouped in four main classes:

1) triangulation/trilateration;
2) proximity;
3) scene analysis; and
4) dead reckoning techniques [1].

Triangulation/trilateration-based techniques rely on the computation of angles and/or distances of the sensor with respect to known points in space. The most popular approach of this type is GPS, using signals from artificial satellites; however, it requires a clear view of the sky. For indoor localization, this approach uses alternative signal sources such as WiFi [2], GSM [3], BlueTooth [4], etc. Proximity-based approaches localize the sensor by detecting its presence near specific devices, e.g., RFID tags [5], while the scene analysis approach uses information from the surrounding environment, e.g., acquired by radar [6] or sonar [7]. Finally, dead reckoning means to trace the position of the sensor by knowing its starting location and using gyroscopes and accelerometers to detect is direction and speed. Since this approach is prone to cumulative errors, it is generally used in conjunction with other techniques [8].

Image-based localization falls in the category of scene analysis techniques. Performing localization by image comparison has some drawbacks, most notably it does not work in zones not covered by the reference data, or observed from different points of view. However, it still has possible applications in environments with narrow paths and few sensible points of view, such as building hallways, or systems where the imaging sensor is forced to move on predefined paths, such as in the case of a patrolling robot. In other contexts, the problem could be mitigated for example by using wide-baseline matching [9] or omnidirectional sensors [10].

Kim and Jun [11] use indoor localization to propose a wearable augmented-reality navigation system. Their approach is based on detection of black-and-white markers placed in known positions, plus several image processing steps to improve the detection performances both in terms of robustness and speed. The main drawback of their system is the need of specific visual markers in the observed environment. Werner *et al.* [12] use visual features to select the best match from a database of images taken at known locations. They use coarse WLAN-based localization to reduce the search space. In comparison, our approach adopts a technique to reduce the search space based only on visual features. Remazeilles and Chaumette [13] use image recognition for robot navigation, where a set of images is used to define a path, and the robot plans its movements by searching the given images in the surrounding environment. Other works use a 2D-to-3D approach, where the reference data is composed of 3-D point features, and the system searches for correspondences in the 2-D view [14]–[16]. The main problems in these systems are the need of a 3-D reference model and the inherent visual 2D-to-3D ambiguity. To overcome this issue, Micusik and Wildenauer [17] propose to use lines, rather than points, as features that best describe typical indoor environments.

Manuscript received March 08, 2016; revised July 01, 2016; accepted July 19, 2016. Date of publication July 22, 2016; date of current version August 15, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Charles Kervrann.

The author is with the Dipartimento di Scienze Matematiche, Informatiche e Fisiche, Università degli Studi di Udine, 33100 Udine, Italy (e-mail: claudio.piciarelli@uniud.it).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2016.2593958

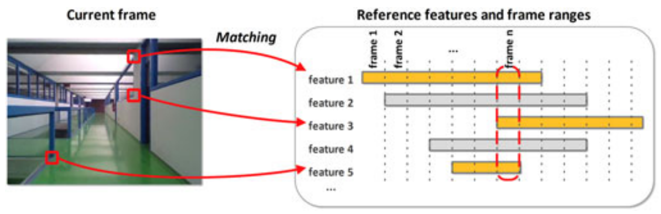

Fig. 1. Main idea behind the proposed algorithm. Features detected in each frame are matched with the ones stored during the offline phase. The intersection of their frame ranges identifies the matched reference frame (and hence the current position).

More generally, image-based localization can be seen as an application of image retrieval techniques. One of the most popular algorithms in this field is the bag-of-visual-words (BoW) approach [18]. In BoW, visual features are extracted from all the reference images and quantized to get a small number of feature prototypes (visual words). Each image is then represented as an histogram of visual words occurrences. The histograms can be used for image classification, matching, etc. In Section II-B we will compare the proposed method with BoW.

III. Proposed Algorithm

The problem of image-based localization is split in two tasks: 1) offline building of a reference model, and 2) online localization. In the offline phase, a video sequence of the indoor environment is acquired and processed to detect visual features. The features are stored together with manually-defined positional information. In the online phase, visual features from each frame are compared with the stored data, and the best-matching reference frame is identified. The current position is defined as the localization data associated with the matched frame.

In order to speed up the matching step, we propose to *track* the features detected in the offline phase, so that the same feature needs to be stored only once, rather than several times for each frame in which it appears. In the online phase, the best reference frame is identified as the intersection of tracking ranges of the matched features, as shown in Fig. 1.

A. *Offline Model Building*

In the offline phase, a video sequence of the environment is acquired. The sequence must cover all the areas in which localization will be needed, and from all the sensible points of view (e.g., the two possible directions of a hallway). Each frame is manually labeled with positional information with a time-consuming procedure that cannot be applied online. In our experiments we annotated the current frame number each time the camera was passing over a known reference point, and the remaining positions have been obtained by interpolation.

During the offline phase, a set F of SURF features [19] is extracted from the first frame. Their positions P are then tracked in the next frames using a standard KLT point tracker [20], as implemented in the MATLAB vision toolbox. To ensure a robust tracking, at each frame we use the RANSAC algorithm to estimate the projective transformation that best describes the feature displacement. It is thus possible to detect and discard outliers,

Algorithm 1: Offline model building.

```
1:  procedure Build-Model
2:      $\mathcal{M} = P = \varnothing$
3:      $F$ = detect-SURF-features ($I_0$) $\triangleright$ $I_0$ is the first frame
4:      $\forall f \in F : P \leftarrow f.Position$
5:      $\forall f \in F : f.First = f.Last = 0$
6:      for each frame number n do
7:          $P'$ = KLT-track($P$, $I_n$)
8:          for each tracked feature $f$ do
9:              $f.Last = n$
10:         for each lost feature $f$ do
11:             $\mathcal{M}.Descriptors \leftarrow f.Descriptor$
12:             $\mathcal{M}.Ranges \leftarrow [f.First, f.Last]$
13:         if $|P'| < Th$ then
14:             $F$ = detect-SURF-features($I_n$)
15:             $\forall f \in F$ : compute the min distance from $P'$
16:             $F$ = sort $F$ by decreasing distance
17:             $F = f_1 \dots f_{Th-|P'|}$
18:             $\forall f \in F$ : $f$.First = $f$.Last = $n$
19:             $\forall f \in F$ : $P' \leftarrow f$.Position
20:         $P = P'$
21:     $\mathcal{M}.tree$ = HKM-Tree($\mathcal{M}.Descriptors$)
22:     return $\mathcal{M}$
```

i.e., points where the distance between their real position and the position predicted by the projective model is larger than a threshold (4 pixels in our experiments). We assume that the feature descriptor of the tracked points will not significantly change during tracking because of the SURF invariance to translations, rotations, and scale changes.

When the number t of tracked points drops below a given threshold Th, new SURF features are extracted and their positions are added to the set of tracked points. In particular, only the $(Th - t)$ features that maximize their minimum distance from the tracked points are selected. This avoids duplicates due to overlaps between the added points and the previously tracked ones.

Whenever a tracked point is lost, its original SURF descriptor (typically, a vector of 64 real numbers [19]) is stored in the reference model $\mathcal{M}$, as well as the first and last frame numbers ($f.First$ and $f.Last$ in Algorithm 1) in which it has been tracked.

After the entire sequence has been processed, we also build a hierarchical k-means tree as described in [21]. The feature descriptors are thus clustered using k-means, and the process is recursively iterated on each detected cluster. The result is a tree where each nonleaf node is labeled with the corresponding cluster center and the original descriptors are stored in the tree leaves. This data structure will be used to perform approximate nearest neighbor matching in the online phase.

Algorithm 1 shows how the offline model is built. At lines 2–5, the algorithm is initialized by detecting the SURF features in the first frame and setting their first and last tracking frame to 0. Their positions (2-D coordinates) are stored in the set of points P. The tracking loop starts at line 6: the selected points

Algorithm 2: Online localization.

```
1: procedure Localize
2:     for each frame number n do
3:         $F$ = detect-SURF-features($I_n$)
4:         $F^{\mathcal{M}}$ = match($F, \mathcal{M}$)
5:         for each matched feature $f_j^{\mathcal{M}} \in F^{\mathcal{M}}$ do
6:             $a[r_j.First \ldots r_j.Last] += 1$
7:         b = $\text{argmax}_i\, a[i]$
8:         return position of reference frame #b
```

are tracked (line 7) and, for each tracked point, the corresponding last frame is updated with the current frame number (line 9). Points that are no longer tracked have their corresponding feature descriptor and frame ranges stored in $\mathcal{M}$ at lines 10–12. Lines 13–19 add new features if their amount is below a given threshold, starting from the farthest ones from the currently tracked points. Finally, once the entire sequence has been analyzed, the hierarchical *k*-means tree is built at line 21 (see [21] for full details).

At the end of the offline phase, the reference model $\mathcal{M}$ contains:

1) a set of descriptors of the tracked features;
2) the first and last frame in which each feature has been tracked;
3) the hierarchical *k*-means tree.

B. Online Localization

During the online localization, the aim is to match each acquired frame to a frame in the reference video sequence. The current position is defined as the localization data associated to the matched frame. At each frame, we detect the set F of SURF features and match them with the features stored in the reference model $\mathcal{M}$. For each feature, a match is found by minimizing the Euclidean distance among descriptors: if $f_i \in F$, its best match is $f_j^{\mathcal{M}} \in \mathcal{M}$, where

$$j = \underset{l}{\text{argmin}} \, \|f_i.\text{Descriptor} - f_l^{\mathcal{M}}.\text{Descriptor}\|. \quad (1)$$

Equation (1) implies a linear scan of all the descriptors in $\mathcal{M}$ and thus could violate the real-time requirements in case of long reference sequences. In this case, an approximate nearest-neighbor match can be found by using the hierarchical *k*-means tree as proposed in [21]. In this case, the search for the best match is done by visiting the entire tree rather than performing a linear search. However, the visiting order is defined by a priority queue, which forces the nodes with cluster centers closest to the processed descriptor to be visited first. This way, there is an high probability that the best match is found in the early stages of scan, and approximation is obtained by stopping the tree visit after a given amount of leaf nodes have been visited. With this approach, the matching time can be drastically improved, at the price of the loss in accuracy due to approximation.

Let now assume that k features in $\mathcal{M}$ have been matched. They are associated to k sets of frame ranges $\{r_i\}_{i=1}^k$, where r_i contains the first and last frame number in which feature i has been tracked. Ideally, the best-matching reference frame

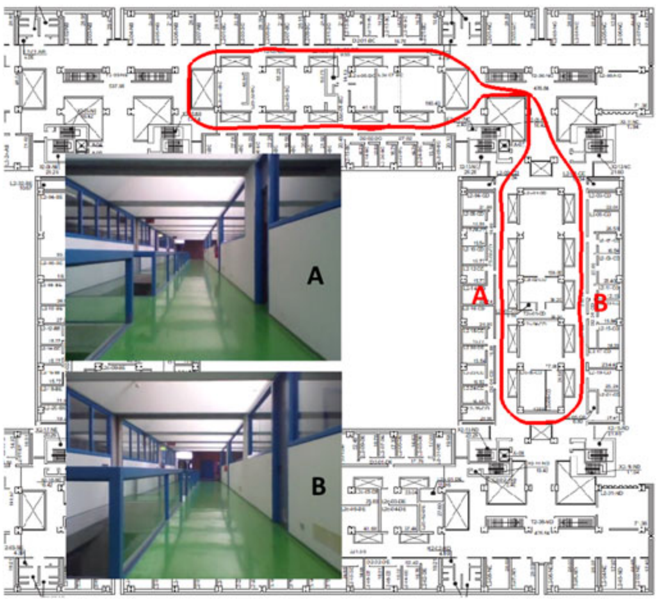

Fig. 2. Path of the camera while acquiring the reference video sequence for environment 1. The depicted area is approximately 125 m × 110 m. Two frames are acquired at points A and B, respectively.

is the one that contains all the matched features, and thus can be defined as $b = \bigcap_{i=1}^k r_i$, as shown in Fig. 1. In practice, however, the intersection is often empty due to tracking errors, thus we use an accumulator to achieve more robust results. The accumulator "**a**" is an initially empty $N \times 1$ vector, where N is the number of frames in the reference sequence. For each matched feature descriptor $f_i^{\mathcal{M}}$ and its corresponding range r_i, the accumulator is properly incremented:

$$\mathbf{a}[r_i.\text{First} \ldots r_i.\text{Last}] = \mathbf{a}[r_i.\text{First} \ldots r_i.\text{Last}] + 1. \quad (2)$$

Once all the matched ranges have been added in the accumulator, the best matching reference frame b is the one that maximizes "**a**":

$$b = \underset{j}{\text{argmax}} \, \mathbf{a}[j]. \quad (3)$$

If several matches are found, they are considered equivalent and the first one is arbitrarily selected. The entire procedure is shown in Algorithm 2.

IV. Experimental Results

In order to test the proposed system, we considered three indoor environments consisting in the hallways of different buildings. In the first environment, the reference sequence is 8144 frames long, acquired at 640 × 480 with the integrated webcam of a laptop placed on a cart. The second and third sequences are 2636 and 2900 frames long, respectively, acquired at full HD with a Motorola Moto G smartphone and then rescaled by a factor 0.5. The smartphone was held by hand, thus the sequences are not stabilized. In some cases, the environments are particularly challenging because of many similar-looking areas (e.g., see Fig. 2). The reference sequences have been manually labeled with positional information by taking note of the frame number when the camera was passing over known reference points roughly placed every 6 m, with all the remaining positions being extrapolated by cubic interpolation. We tried to keep a constant direction and speed between the reference points in order to minimize the errors introduced by interpolation.

As a test set, four sequences have been acquired in Environment 1, and two sequences in each one of the other environments. Test sequences have been manually labeled with positional information for error measuring purposes only. For each sequence frame, we estimated its localization with the algorithms proposed in Section II and measured its Euclidean distance from the manual label. The mean error, however, is not a good quality metric: when the system fails, the falsely-matched frame could be located anywhere on the path, thus leading to unpredictable, and possibly high, error values. We thus propose to count the number of outliers as a more robust error metric. In our tests, a localization result is considered an outlier if the error is above 4 m. The average error is then computed on inlier data only. It is worth noting that the results could be improved by imposing some form of spatial coherence (e.g., with a Kalman filter). However, we chose not to use any kind of filtering and classify each frame independently, in order to give more generic results that are meaningful also for other types of applications (e.g., single-shot localization). The testing platform is a linux PC running MATLAB, with a Xeon E5-2660 CPU and 128 GB RAM.[1]

Algorithm 1 was run on the reference sequences. The threshold on the minimum number of features to track in each frame was set to 100, as the system does not seem to improve its performances with higher values. The hierarchical k-means tree was built with $k = 32$, although the parameter does not seem to significantly influence the error rates. If compared with the naive approach of storing each extracted feature, the proposed method reduced the feature database size to 7.7%, 4.7%, and 2.6% of the original sizes in the three sequences.

Table I(a) shows the localization results of the proposed method with linear scan [see (1)]. We report the number of outliers and the mean error computed only on inliers. The total processing time for each frame is also shown. For each environment, we show the average values obtained on the test sequences. In all the environments, the algorithm performed well, by limiting the number of outliers to a very small fraction of the processed frames. In all the remaining frames, we have a localization error ranging from 0.58 to 0.74 m. Timings are relatively high, from a minimum of 119 ms up to 224 ms to process a single frame. Timings include the SURF feature extraction step (approximately 43 ms in the first environment and 65 ms in the other ones).

The running time can be improved by using approximate search in the hierarchical k-means tree. In this case, we fixed the number of leaves to be visited to 32. The search running time is thus constant, at the price of an approximation in the results. However, the technique performed extremely well in all the test sequences, as it can be seen in Table I(b). Both the number of outliers and the average localization errors are comparable with the linear scan results. On the other side, processing time is significantly reduced, especially if we consider that it is mostly due to the SURF feature extraction.

[1]Video sequences and source code available at `http://avires.dimi.uniud.it/papers/il16/`.

TABLE I
EXPERIMENTAL RESULTS IN THREE ENVIRONMENTS (MEAN ERRORS ARE EXPRESSED IN METERS AND TIMINGS IN MILLISECONDS)

Environment	Outliers	Mean inliers error	Processing time
1	0.05% (4/8730)	0.58	224
2	0.13% (3/2326)	0.74	150
3	0.13% (3/2335)	0.68	119
(a) Proposed method			
1	0.09% (8/8730)	0.60	52
2	0.13% (3/2326)	0.73	74
3	0.17% (4/2335)	0.68	69
(b) Proposed method with approximate match			
1	0.26% (23/8730)	0.67	78
2	0.47% (11/2326)	0.78	88
3	0.56% (13/2335)	0.66	88
(c) Bag of visual words			
1	0.31% (27/8730)	0.67	52
2	0.21% (5/2326)	0.78	76
3	0.38% (9/2335)	0.65	73
(d) BoW with approximate match			

Finally, we compared our method with the popular BoW approach. It reduces the search space by vector quantization, while we exploit temporal coherence of video sequences through feature tracking. For the comparison to be fair, we used the same model size in both approaches. For example, in Environment 1, we considered 63 071 features over 8144 frames. Since each SURF descriptor consists of 64 real numbers, we use roughly 496 values per frame. The BoW approach stores a histogram for each frame, which represents the occurrence rates of all the visual words in each image. We thus fixed the vocabulary size to 496. As it can be seen in Table I(c), the BoW approach performs worse than the proposed method in terms of outliers. Changing the vocabulary size to 100, 250, 1000, 2000, and 5000 did not lead to better results. The source of errors mainly lies in many different physical locations which are visually similar.

Regarding the computational time, the BoW approach performs faster than our linear scan method, but slower than the hierarchical k-means tree matching. However, the approximate search can be used to speed up the BoW histogram matching too, and the results are shown in Table I(d). As it can also be seen in this case that the approximation does not have significant impact on the system accuracy. The running times are comparable with the proposed approach.

V. CONCLUSION

We proposed a visual indoor localization system that can run in real time even in presence of a large amount of reference data. Experimental results show that the system can achieve a good localization accuracy, given that proper reference data is available. The system outperformed the popular BoW algorithm in terms of detected outliers, while keeping comparable running times.

References

[1] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.

[2] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki, "On the feasibility of using wireless ethernet for indoor localization," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 555–559, Jun. 2004.

[3] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason, "GSM indoor localization," *Pervasive Mobile Comput.*, vol. 3, no. 6, pp. 698–720, 2007.

[4] L. Pei, R. Chen, J. Liu, H. Kuusniemi, T. Tenhunen, and Y. Chen, "Using inquiry-based Bluetooth RSSI probability distributions for indoor positioning," *J. Global Positioning Syst.*, vol. 9, no. 2, pp. 122–130, 2010.

[5] G.-Y. Jin, X.-Y. Lu, and M.-S. Park, "An indoor localization mechanism using active RFID tag," presented at the *IEEE Int.* Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing, Taichung, Taiwan, 2006.

[6] J. W. Marck, M. Ali, and R. Van Heijster, "Indoor radar SLAM A radar application for vision and GPS denied environments," in *Proc. Eur. Radar Conf.*, 2013, pp. 471–474.

[7] J. Steckel, A. Boen, and H. Peremans, "Broadband 3-D sonar system using a sparse array for indoor navigation," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 161–171, Feb. 2013.

[8] Z. Yang, C. Wu, Z. Zhou, X. Zhang, X. Wang, and Y. Liu, "Mobility increases localizability: A survey on wireless indoor localization using inertial sensors," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 1–34, 2015.

[9] A. Makadia, "Feature tracking for wide-baseline image retrieval," in *Proc. Eur. Conf. Comput. Vis.*, 2010, vol. 6315, pp. 310–323.

[10] M. Lourenco, V. Pedro, and J. P. Barreto, "Localization in indoor environments by querying omnidirectional visual maps using perspective images," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 2189–2195.

[11] J. Kim and H. Jun, "Vision-based location positioning using augmented reality for indoor navigation," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 954–962, Aug. 2008.

[12] M. Werner, M. Kessel, and C. Marouane, "Indoor positioning using smartphone camera," in *Proc. Int. Conf. Indoor Positioning Indoor Navig.*, Sep. 2011, pp. 21–23.

[13] A. Remazeilles and F. Chaumette, "Image-based robot navigation from an image memory," *Robot. Autonom. Syst.*, vol. 55, no. 4, pp. 345–356, 2007.

[14] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *Proc. 12th Eur. Conf. Comput. Vis.*, vol. 7572, 2012, pp. 752–765.

[15] A. Irschara, C. Zach, J. M. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2009, pp. 2599–2606.

[16] L. Svärm, O. Enqvist, M. Oskarsson, and F. Kahl, "Accurate localization and pose estimation for large 3D models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 532–539.

[17] B. Micusik and H. Wildenauer, "Descriptor free visual indoor localization with line segments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3165–3173.

[18] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. Workshop Stat. Learn. Comput. Vis.*, 2004, pp. 1–2.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, 2008.

[20] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Comput. Soc. Conf. Vis. Pattern Recognit.*, 1994, pp. 593–600.

[21] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory. Appl.*, 2009, pp. 1–10.