Mapping Large Loops with a Single Hand-Held Camera

Laura A. Clemente
Instituto de Investigación
en Ingeniería de Aragón
Universidad de Zaragoza, Spain
laura.clemente@unizar.es

Andrew J. Davison
Dept. of Computing
Imperial College London
ajd@doc.ic.ac.uk

Ian D. Reid
Dept. of Eng. Science
University of Oxford
ian@robots.ox.ac.uk

José Neira and Juan D. Tardós
Instituto de Investigación
en Ingeniería de Aragón
Universidad de Zaragoza, Spain
jneira,tardos@unizar.es

***Abstract*— This paper presents a method for Simultaneous Localization and Mapping (SLAM), relying on a monocular camera as the only sensor, which is able to build outdoor, closed-loop maps much larger than previously achieved with such input. Our system, based on the Hierarchical Map approach [1], builds independent local maps in real-time using the EKF-SLAM technique and the inverse depth representation proposed in [2]. The main novelty in the local mapping process is the use of a data association technique that greatly improves its robustness in dynamic and complex environments. A new visual map matching algorithm stitches these maps together and is able to detect large loops automatically, taking into account the unobservability of scale intrinsic to pure monocular SLAM. The loop closing constraint is applied at the upper level of the Hierarchical Map in near real-time.**

We present experimental results demonstrating monocular SLAM as a human carries a camera over long walked trajectories in outdoor areas with people and other clutter, even in the more difficult case of forward-looking camera, and show the closing of loops of several hundred meters.

I. Introduction

Simultaneous Localization And Mapping (SLAM) is one of the most active research fields in robotics, with excellent results obtained during recent years, but until recently mainly restricted to the use of laser range-finder sensors and predominantly building 2D maps (see [3] [4] for a recent review). Under these conditions, robust large-scale indoor and outdoor mapping has now been demonstrated by several groups around the world.

It is more challenging to attempt SLAM with standard cameras as the main sensory input, since the essential geometry of the world does not 'pop-out' of images in the same way as it does from laser data. Nevertheless, the combination of detailed 3D geometric and photometric information available from cameras means that they have great promise for SLAM applications of all types, and recent progress has been very encouraging. In particular, recent robotic SLAM systems which use odometry and single cameras [5], [6], stereo rigs [7], omnidirectional cameras [8] or inertial sensors [9] have all demonstrated reliable and accurate vision-based localisation and mapping, often in real-time and on increasingly large scales. Also impressive have been stereo vision-based 'visual odometry' approaches [10], [11] which match large numbers of visual features in real-time over sequences and obtain highly accurate local motion estimates, but do not necessarily aim to build globally consistent maps.

In this paper, we consider the extreme case where the only sensory input to SLAM is a single low-cost 'webcam', with no odometry, inertial sensing or stereo capability for direct depth perception – a camera carried by a walking person, for example. Under these conditions, successful real-time SLAM approaches have been limited to indoor systems [12]–[14] which can build maps on the scale of a room. Such work on estimating motion and maps from a single moving camera must also be compared with the wealth of work in visual structure from motion. (e.g. [15]) where high quality reconstructions from image sequences are now routinely obtained, but requiring significant off-line optimisation processing.

Now we show that an approach which builds and joins local SLAM maps, previously proven in laser-based SLAM, can be used to obtain much larger outdoor maps than previously built with single camera only visual input and works in near real-time. The keys are the efficient and accurate building of local submaps, and robust matching of these maps despite high localisation uncertainty. Other approaches to vision-based closing of large loops in SLAM have used appearance-based methods separated from the main mapping representation [8], [16]. While these methods are certainly valuable, here we show that under the conditions of the experiments in our paper we are able to directly match up local maps by photometric and geometric correspondences of their member features.

One of the main difficulties of monocular visual SLAM is landmark initialization, because feature depths cannot be initialized from a single observation. In this work we have adopted the inverse-depth representation proposed by Montiel et al. [2], which performs undelayed initialization of point features in EKF-SLAM from the first instant they are detected. In that work, data association was performed by predicting the feature locations in the next image and matching them by correlation. In this paper we demonstrate that adding a Joint Compatibility test [17] makes the method robust enough to perform for the first time real-time monocular SLAM walking with a hand-held camera in urban areas. In our experiments, the inverse depth representation allows SLAM to benefit from features which are far away from the camera, which are revealed to be essential to maintaining good angular accuracy

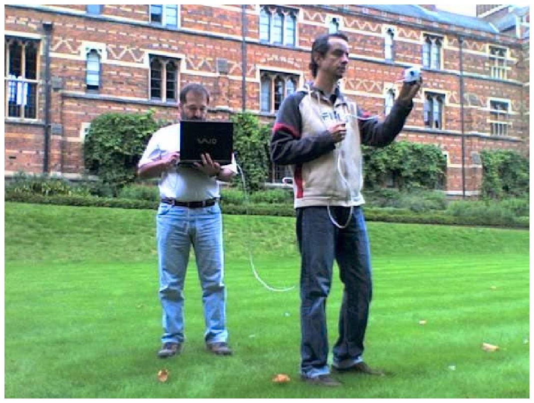


Fig. 1. Experimental setup: a hand-held camera, a firewire cable and a laptop.

in open areas. The joint compatibility technique is able to successfully reject incorrect data associations which jeopardize the operation of SLAM in repetitive or dynamic environments.

To attack the problem of mapping large areas, the technique is applied to build several independent local maps that are integrated into the Hierarchical Map approach proposed by Estrada et al. [1]. Two of the main factors that fundamentally limit EKF-based SLAM algorithms are (i) the processing time associated with the EKF update which is $O(n^2)$ in the number of map features; and (ii) cumulative linearisation errors in the EKF that ultimately contribute to biased and overconfident state estimates which eventually break the filter, usually via poor data association. Hierachical SLAM addresses both of these issues. First, by segmenting the problem into smaller chunks of bounded size, the computational time of the filter is bounded (i.e. $O(1)$). Second, since each local map effectively resets the base frame, linearisation errors only accumulate *within* a local map and not between maps. The main difficulty appearing here is that the scale in pure monocular vision is not observable, so the scale of the different local maps is not consistent. We propose a novel scale invariant map matching technique in the spirit of [18], able to detect loop closures, that are imposed in the upper level of the Hierarchical Map, obtaining a sub-optimal SLAM solution in near real time.

The rest of the paper is structured as follows. Section II describes in detail the local map building technique proposed and presents some experiments showing its robustness in real environments. Section III presents the map matching algorithm and the loop optimization method used at the global level of the Hierarchical Map. Section IV demonstrates the technique by mapping a courtyard by walking with the camera in hand (see fig. 1) along a loop of several hundred meters. The conclusions and future lines of research are drawn in section V.

II. Building Monocular Local Maps

A. EKF SLAM with inverse depth representation

To achieve scalability to large environments we have adopted the Hierarchical Map method proposed in [1]. This technique builds a sequence of local maps of limited size using the EKF-SLAM approach [19]. We have observed that, in the case of monocular SLAM, the use of the Iterated Extended Kalman Filter (IEKF) [20] improves the accuracy of the map and the camera trajectory, at the price of a small increase in the computational cost. In any case, by limiting the maximum size of the local maps, the computation time required per step during the local map building is bounded by a constant.

The state vector of each local map $\mathbf{M}_i$ comprises the final camera location $\mathbf{x}_v^i$ and the 3D location of all features $(\mathbf{y}_1^i \dots \mathbf{y}_n^i)$, using as base reference B, the camera location at the beginning of the local map. We also store the complete camera trajectory inside each local map, that is used only for displaying results. For the state representation inside each local map, we use the inverse-depth model proposed by Montiel *et al.* [2]:

$$\mathbf{x}^T = (\mathbf{x}_v^T, \mathbf{y}_1^T, \mathbf{y}_2^T, \dots, \mathbf{y}_n^T) \tag{1}$$

where:

$$\mathbf{x}_v = \begin{pmatrix} \mathbf{r}^{BC} \\ \mathbf{q}^{BC} \\ \mathbf{v}^B \\ \mathbf{w}^C \end{pmatrix} \tag{2}$$

$$\mathbf{y}_i = (x_i \, y_i \, z_i \, \theta_i \, \phi_i \, \rho_i)^T \tag{3}$$

This feature representation codes the feature state as the camera optical center location $(x_i \, y_i \, z_i)$ when the feature point was first observed, and the azimuth and elevation $(\theta_i \, \phi_i)$ of the ray from the camera to the feature point. Finally, the depth d_i along this ray is represented by its inverse $\rho_i = 1/d_i$. The main advantage of the inverse-depth parametrization is that it allows consistent undelayed initialization of the 3D point features, regardless of their distance to the camera. In fact, distant points, or even points at *infinity* are modelled and processed in the same way. This is in contrast with most current techniques that delay the use of a feature until the baseline is big enough to compute its depth [8], [12].

The camera state $\mathbf{x}_v$ is composed of the camera position $\mathbf{r}^{BC}$ and orientation quaternion $\mathbf{q}^{BC}$ and its linear and angular velocities $\mathbf{v}^B$ and $\mathbf{w}^C$. The process model used for the camera motion is a constant velocity model with white Gaussian noise in the linear and angular accelerations. Using pure monocular vision, without any kind of odometry, the scale of the map is not observable. However, by choosing appropriate values for the initial velocities and the covariance of the process noise, the EKF-SLAM is able to ”guess” an approximate scale for each local map, as will be shown in the experimental results.

B. Feature extraction and matching

Now we focus on the features selection which make up the local maps. Our goal is to be able to recognize the same features repeatedly during local map building and also for loop closing detection and optimization. So what we need are persistent and realiable features that ensure us with high probability a quality tracking process. For this very purpose we have followed the approach of Davison *et al.* [12], [21], who showed that selecting salient image patches (11 x 11 pixels) is useful for performing long-term tracking.

To detect salient image regions we use the Shi and Tomasi operator [22] with some modifications which result in more salient and better trackable features. The first modification is the application of a gaussian weighted window to the Hessian matrix (4) which makes the response of the detector isotropic and results in patches better centered around the corner or salient point.

$$H = \begin{pmatrix} G_\sigma * (I_x I_x) & G_\sigma * (I_x I_y) \\ G_\sigma * (I_x I_y) & G_\sigma * (I_y I_y) \end{pmatrix} \quad (4)$$

Apart from using the Shi and Tomasi response:

$$\lambda_{min} > \lambda_{threshold} \quad (5)$$

where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of the Hessian image matrix (4) respectively we only accept as good feature points those whose two eigenvalues have similar magnitude:

$$\lambda_{max} / \lambda_{min} < ratio_{threshold} \quad (6)$$

This avoids selecting regions with unidirectional patterns that cannot be tracked reliably. Instead of using all the features that passed both tests, we have implemented a simple selection algorithm that forces a good distribution of features on the image. The features that pass the tests are stored in a 2D-spatial structure. When the number of tracked features falls bellow a threshold, the spatial structure is used to find the best feature (with higher Shi-Tomasi response) from the image region with less visible features.

For tracking the features on the sequence of images we use the active search approach of Davison and Murray [21]. The stochastic map is used to predict the location of each feature in the next image and compute its uncertainty ellipse. The features are searched by correlation inside the uncertainty ellipse using normalised sum-of-squared-differences. This gives enough robustness with respect to light condition changes and also to small viewpoint changes.

C. Joint compatibility

It is well known that data association is one of the most critical parts in EKF-SLAM, since a few association errors may ruin the quality of an otherwise good map. The active search strategy presented gives good feature matchings *most* of the time. However, since we are building maps of large outdoor dynamic environments we have to deal with two well differentiated problems. The first problem is that moving objects produce valid matches – in that they correspond to the same point on the object – which nevertheless violate the basic assumption of static features made by most SLAM techniques. The second problem arises in the presence of ambiguous matches caused, for example, by repeated texture in the environment. Such ambiguous matches are more likely with "young" features whose 3D locations (especially depth) are still uncertain, leading to large search regions.

Algorithm 1 Simplified Joint Compatibility:
$\mathcal{H}$ = simplified_JCBB ()

```
$\mathcal{H} \Leftarrow$ [true]$^m$
if not joint_compatibility($\mathcal{H}$) then
    Best $\Leftarrow$ []
    JCBB([], 1)
    $\mathcal{H} \Leftarrow$ Best
end if
```

Algorithm 2 Recursive Joint Compatibility:
JCBB ($\mathcal{H}$, i) : *find pairings for observation E_i*

```
if i = m then {Leaf node}
    if num_pairings($\mathcal{H}$) $>$ num_pairings(Best) then
        Best $\Leftarrow \mathcal{H}$
    else if num_pairings($\mathcal{H}$) $=$ num_pairings(Best) then
        if $D^2(\mathcal{H}) < D^2$(Best) then
            Best $\Leftarrow \mathcal{H}$
        end if
    end if
else {Not leaf node}
    if joint_compatibility([$\mathcal{H}$ true]) then
        JCBB([$\mathcal{H}$ true], i + 1) {pairing $(E_i, F_j)$ accepted}
    end if
    if num_pairings($\mathcal{H}$) + m - i $\geq$ num_pairings(Best) then
        {Can do better}
        JCBB([$\mathcal{H}$ false], i + 1) {Star node: $E_i$ not paired}
    end if
end if
```

One approach to tackling the problems of repeated patterns is to try to avoid it altogether by selecting features that are highly salient [16]. In this work we take the complementary approach of explicitly detecting and rejecting these matches using the notion of Joint Compatibility, as proposed in [17]. The idea is that the whole set of matchings accepted in one image must be *jointly* consistent.

As the active search gives only one candidate for each matched feature, and the matchings are usually good, we have implemented a simplified version of the Joint Compatibility algorithm that reduces the computational cost of including a backtracking algorithm at each update of the EKF (see Alg. 1). The algorithm tries first the optimistic hypothesis $\mathcal{H}$ that each image feature E_i pairs correctly with its corresponding map feature F_i verifying the *joint compatibility* using the Mahalanobis distance and the Chi-squared distribution:

$$D^2_{\mathcal{H}} = \nu^T_{\mathcal{H}} C^{-1}_{\mathcal{H}} \nu_{\mathcal{H}} < \chi^2_{d,\alpha} \quad (7)$$

where $d = 2\cdot$ num_pairings($\mathcal{H}$), α is the desired confidence level (0.95 by default) and $\nu_{\mathcal{H}}$ and $C_{\mathcal{H}}$ are the joint innovation and its covariance:

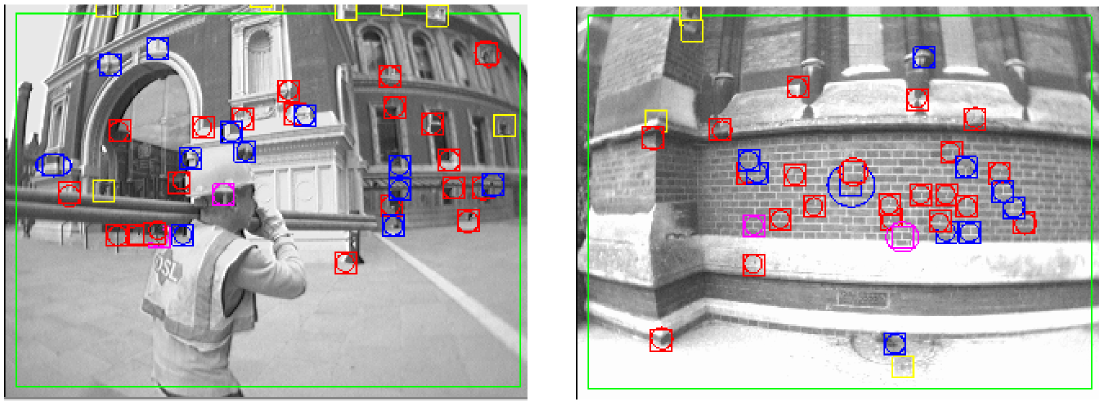
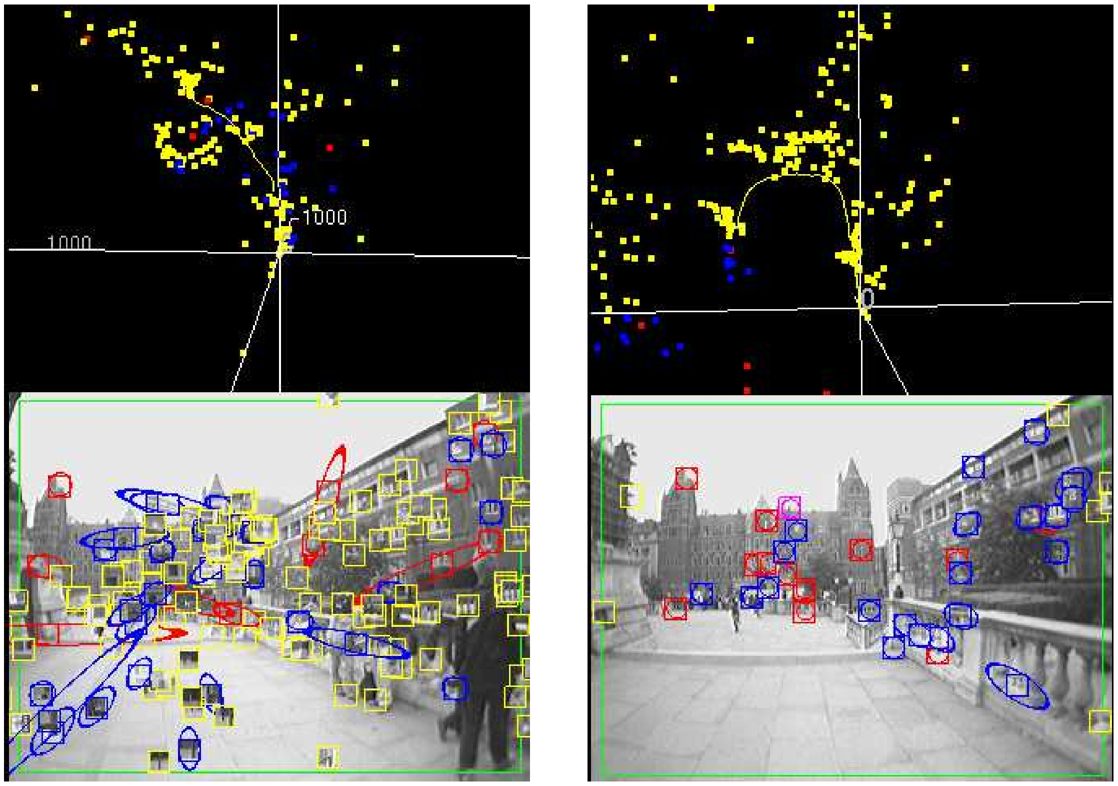


Fig. 2. Incorrect matches successfully rejected by the joint compatibility algorithm (marked in magenta).



(a) Without Joint Compatibility

(b) With Joint Compatibility

Fig. 3. Local map obtained in a dynamic outdoor environment along a U-shaped trajectory. The colored ellipses represent the uncertainty of feature localization and the yellow line on the map corresponds to the computed camera trajectory. Red: features predicted and matched, Blue: features predicted but not found, Yellow: features not predicted.

$$\nu_{\mathcal{H}} = \mathbf{z}_{\mathcal{H}} - \mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}) \quad (8)$$
$$C_{\mathcal{H}} = H_{\mathcal{H}} P H_{\mathcal{H}}^T + R_{\mathcal{H}} \quad (9)$$

This compatibility test only adds the computation of equation (7), because we already need the innovation and its covariance for the update phase of the Kalman filter. Only when the innovation test of the complete hypothesis is not satisfied, our simplified algorithm performs the branch and bound search of the JCBB algorithm to find the largest subset of jointly compatible matchings, as shown in Alg. 2. As the active search has only found one possible pair F_i for each measurement E_i, the solution space consists of a binary tree (accepting or rejecting each possible pair) whose depth is the number of measurements m. It should be noted that the results of JCBB are order-independent. Even when a matching has been accepted, the ”star node” part of the algorithm also analyzes all the hypothesis that do not contain that matching.

An alternative technique that that could be used to detect false matchings is RANSAC (see for example [10]). Our

JCBB technique has two advantages: it allows matchings with features that have been occluded for a while and it is able to reject outliers using all points in the same way, even points with high depth uncertainty or points at infinity.

To verify the robustness of this technique we conducted a mapping experiment in an outdoor populated area. Most of the time, all matchings found by correlation were correct, and the branch and bound algorithm was not executed. Figure 2 shows two typical cases where the Joint Compatibility successfully rejected wrong matchings on dynamic and repetitive parts of the environment. Even in this cases, the computational cost added is negligible. The key question is: if the number of bad matchings is so small, how bad can they be for the SLAM process? The answer is given in figure 3. We followed a long U-shaped trajectory walking with a hand-held camera looking forward. Figure 3(a) shows the dramatic effect of the moving people and the repetitive environment patterns. The estimated trajectory is completely wrong and the monocular SLAM algorithm is trying to find in the image features that are actually well behind the camera (drawn in yellow). Running on the same dataset, the inverse-depth SLAM algorithm with the Joint Compatibility test gives the excellent map of figure 3(b). To our knowledge this is the first demonstration of a real-time SLAM algorithm walking with a camera in hand in a large outdoor environment.

III. Hierarchical SLAM

A. Building sequences of local maps

To achieve scalability to large environments, the technique described in the previous section is used to build a sequence of independent local maps of limited size that are later combined using the Hierarchical Map technique [1]. In this way, the computational cost of the EKF-SLAM iteration is constrained to real-time operation. Once the current map reaches the maximum number of features, it is freezed and a new local map is initialized, using as base reference the current camera location. To maintain the statistical independence between local maps, no information is transferred from the previous map to the new one.

When a new map is started, the set of features currently visible in the old map are inserted into the new map as *new* inverse-depth features, using their image locations as the *only* knowledge of their 3D locations. This is important since, though it may seem to be throwing away the prior knowledge of their locations from the previous map, it is only through doing so that the local maps remain independent, yielding the desired O(1) update. It is, however important that there is a group of features which are represented in adjacent maps, since only through these common features can loop-closing and trajectory refinement be effected. These common features are used to estimate the change on the scale factor that may exists between consecutive maps.

At this point, we have a series of local maps ($M_1, \ldots, M_n$) containing the state vector defined in eq. (1) and its covariance matrix. The final camera location $\mathbf{x}_C^i$ in map i corresponds to the base reference of map $i+1$. The transformations between the successive local maps and their covariances constitute the global level of the Hierarchical Map:

$$hmap.\hat{\mathbf{x}} = \begin{pmatrix} \mathcal{T}_1^W \\ \mathcal{T}_2^1 \\ \vdots \\ \mathcal{T}_n^{n-1} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{x}}_C^1 \\ \hat{\mathbf{x}}_C^2 \\ \vdots \\ \hat{\mathbf{x}}_C^n \end{pmatrix} \tag{10}$$

$$hmap.P = \begin{pmatrix} P_1 & 0 & \ldots & 0 \\ 0 & P_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & P_n \end{pmatrix} \tag{11}$$

B. Scale-invariant map matching

By composing the camera state locations $\mathcal{T}_i^{i-1}$ we are able to compute the current camera location and hypothesize loop closures. To verify the loop we have developed a map matching algorithm (see Algorithms 3 and 4) able to deal with the presence of an unknown scale factor between the overlapping maps. First, the method uses normalized correlation to find features in both maps that are compatible (unary constraints). Then, a specialized version of the Geometric Constraints Branch and Bound (GCBB) algorithm [23] is used to find the maximal subset of geometrically compatible matchings, by comparing the relative distances between feature points is space (binary constraints). Although the results of the algorithm is also order independent, its running time may benefit from a good ordering (most promising matchings first). Then, the homogeneous transformation $\mathcal{T}_j^{i-1}$ and the scale change between both maps is estimated from the subset of matched features. An example of loop detection using this technique is shown in figure 4. This technique is able to find loops when the view points in both maps are similar. To detect loops with arbitrary viewpoints, invariant features such as SIFT would be needed.

C. Loop optimization

We have local independent maps scaled to the same reference and we also know the relation between two overlapping maps that close a loop. Then, the Iterated Extended Kalman Filter [20] is used for re-estimating the transformations between the maps that from the loop, as proposed in [1]. The measurement $\mathbf{z}$ corresponds to the transformation estimated by the map matching algorithm, and the measurement function is given by the compositions of all local map states that form the hierarchical map:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) = \mathcal{T}_i^{i-1} \oplus \mathcal{T}_{i+1}^{i} \oplus \ldots \oplus \mathcal{T}_j^{j-1} \tag{12}$$

IV. Experimental results

To validate the proposed SLAM method we have conducted an experiment in a large and dynamic outdoor environment. The experimental setup consists of a low cost Unibrain IEEE1394 camera with a 90 degree field of view, acquiring monochrome image sequences of 320x240 resolution at 30

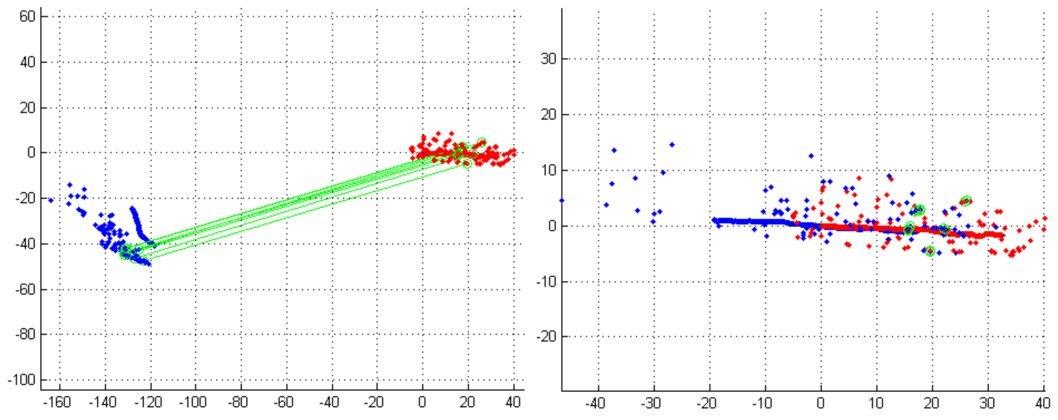


Fig. 4. Loop closure detection. Matchings found between two maps (left) and aligned maps (right).

Algorithm 3 Map Matching GCBB with variable scale:
$\mathcal{H}$ = map_matching_GCBB (observations, features)

```
unary ⇐ compute_unary_constraints(features, observations)
binary.O.distances ⇐ estimate_distances(observations)
binary.F.distances ⇐ estimate_distances(features)
Best.H ⇐ []
Best.scale ⇐ -1.0
variable_scale_GCBB([], 0)
$\mathcal{H}$ ⇐ Best
```

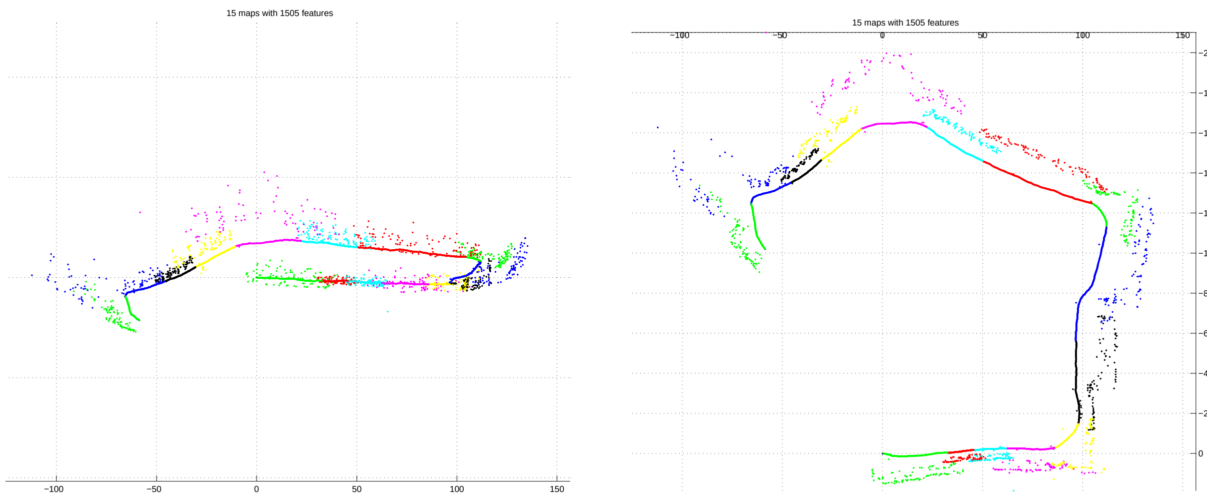
fps, a firewire cable and a laptop (see Fig. 1). We acquired a real image sequence of 6300 frames walking in a courtyard along a loop trajectory of around 250 meters, with the camera in hand, looking to one side. The sequence was processed with the proposed algorithms on a desktop computer with an Intel Core 2 processor at 2,4GHz. Figure 5(a) shows the sequence of independent local maps obtained with the inverse-depth EKF-SLAM using joint compatibility. As it can be seen in the figure, the algorithm ”guesses” an approximate scale that is different for each local map. When a map is finished, it is matched with the previous map and the relative change on the scale is corrected, as shown in figure 5(b). When a loop is hypothesized, the map matching algorithm is executed to find the loop closure, and the loop constraint is applied at the upper level of the Hierarchical Map, giving the result of figure 5(c).

The local map building process has been tested to run in real-time (at 30Hz) with maps up to 60 point features. During the experiments, the joint compatibility algorithm consumed $200\mu s$ at every step and, when occasionally the complete search is executed, the computation cost increases only up to $2ms$, which is an acceptable cost for the great increase in robustness and precision obtained.

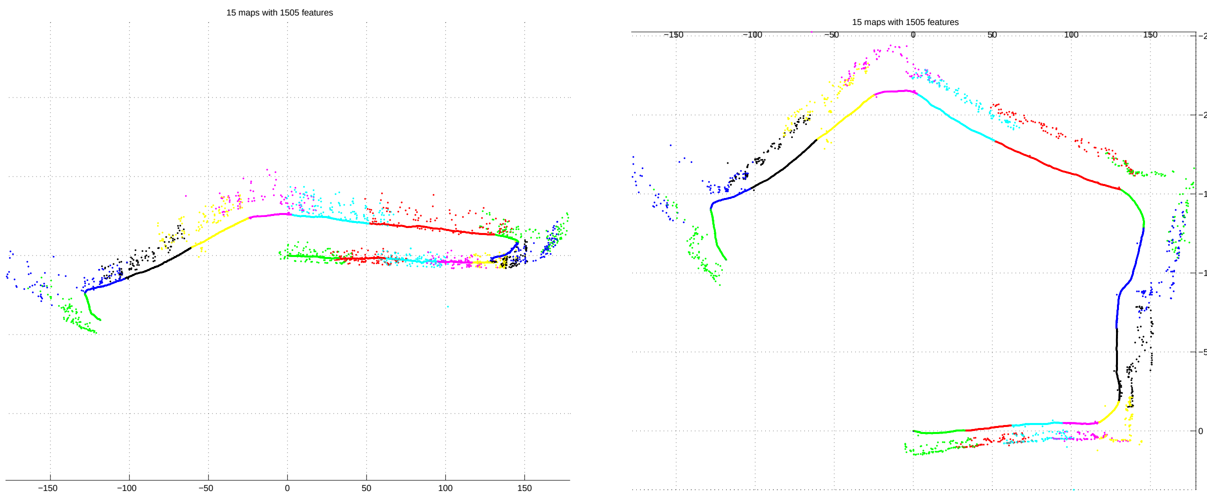
The map matching, scale adjustment and loop optimization phases have been implemented in Matlab. The scale factor estimation between two adjacent maps takes about $120ms$ and the loop optimization using the IEKF takes $800ms$ when performing 6 iterations. The most expensive part is the

Algorithm 4 Recursive Modified Geometric Constraints:
variable_scale_GCBB ($\mathcal{H}$, i) : *find pairings for observation* E_i

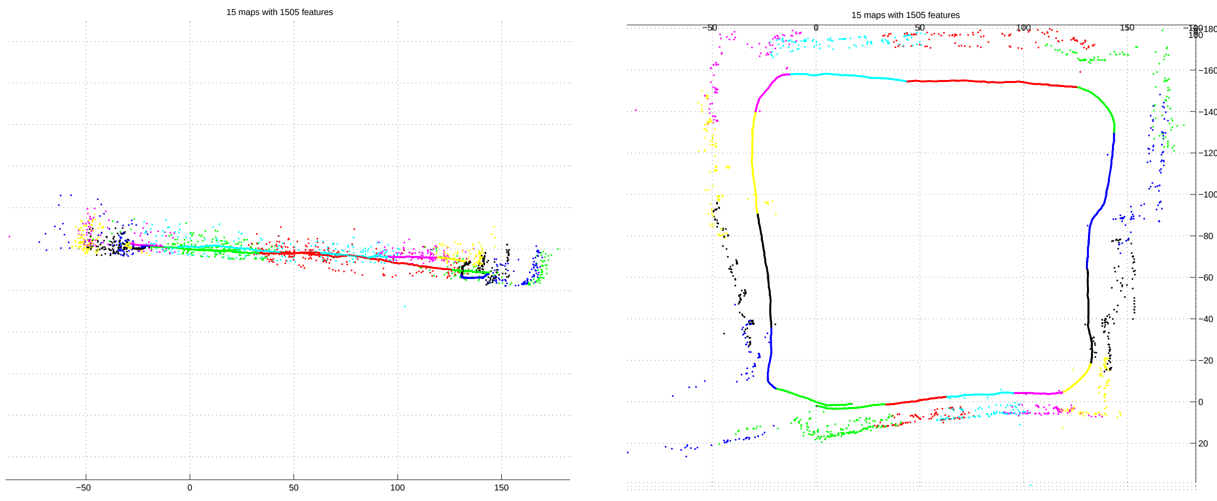
```
if i > m then {Leaf node}
   if num_pairings(H) > num_pairings( Best.H) then
      Best.H ⇐ $\mathcal{H}$
      Best.scale ⇐ binary.scale
   end if
else {Not leaf node}
   if num_pairings($\mathcal{H}$) == 0 then {This is the first pair}
      for ∀j | (unary(i,j) == true) do
         variable_scale_GCBB([H j], i+1)
      end for
   else if num_pairings($\mathcal{H}$) == 1 then {This is the 2nd pair}
      k ⇐ { K | H(K) $\neq$ 0 }
      distance_obs ⇐ binary.O.distances(i,k)
      for ∀j | (unary(i,j) == true) do
         distance_feat ⇐ binary.F.distances(j,H(k))
         if distance_feat $\neq$ 0 then
            binary.scale ⇐ distance_obs ÷ distance_feat
            binary.satisfies ⇐ binary_constraints(binary.scale)
            variable_scale_GCBB([H j], i+1)
         end if
      end for
   else {Normal recursion with binary constraints calculated}
      for ∀j | ( (unary(i,j) == true) AND ({∀k | H(k) $\neq$ 0 }
               AND binary.satisfies(i, k, j, H(k))}) ) do
         variable_scale_GCBB([H j], i+1)
      end for
   end if
end if
{Checking if can do better}
if num_pairings(H) + m - i > num_pairings(Best.H) then
   variable_scale_GCBB([H 0], i+1) {Star node: $E_i$ no paired}
end if
```



(a) Local maps obtained with pure monocular SLAM



(b) Hierarchical map auto-scaled, before loop detection



(c) Hierarchical map after loop closing

Fig. 5. Results obtained mapping a loop of several hundred meters with a camera in hand: side view (left) and top view (right).

scale-invariant map matching algorithm that takes around one minute in our current Matlab implementation. We expect that an optimized C++ implementation running on background will provide close to real time loop detection.

V. Conclusion

In this paper we have demonstrated for the first time that monocular vision-only SLAM – with a single hand-held camera providing the *only* data input – can achieve large-scale outdoor closed-loop mapping in near real-time. Achieving these results which such basic hardware opens up new application areas for vision-based SLAM, both in flexible (possibly low-cost) robotic systems and related areas such as wearable computing. The success of our system lies in the careful combination of the following elements:

- An inverse depth representation of 3D points. It allows the use of partial information, inherent to monocular vision, in a simple and stable way. All features, even those far from the camera, immediately contribute valuable information.
- A branch and bound joint compatibility algorithm that allows the rejection of measurements coming from moving objects that otherwise plague and corrupt the map. Although texture gives a powerful signature for matching points in images, the spatial consistency that this algorithm enforces is essential here.
- A Hierarchical SLAM paradigm in which sequences of local maps of limited size are managed, allowing the system to work with bounded complexity on local maps during normal operation. By running the map matching algorithm in the background, the system can attain real time execution.
- A new map matching algorithm to detect large loops which takes into account the unobservability of the scale intrinsic to pure monocular SLAM. This algorithm allows us to detect loop closures even when the maps involved have been computed with different scales.

Future work includes improving the map matching algorithm to reach real time performance, possibly using invariant feature descriptors. A current limitation of Hierarchical SLAM is the fact that it does not make use of matchings between neighboring maps. We plan to investigate new large mapping techniques that can overcome this limitation, obtaining maps closer to the optimal solution.

Acknowledgments

This research has been funded in part by the Royal Society International Joint Project 2005/R2, DGA(CONSI+D)-CAI under Programa Europa, Dirección General de Investigación of Spain grant DPI2006-13578, the Engineering and Physical Sciences Research Council (EPSRC) grant GR/T24685/01 and an EPSRC Advanced Research Fellowship to AJD.

Thanks to José M. M. Montiel and Javier Civera for providing their code for inverse-depth visual SLAM and the fruitful discussions maintained and to Paul Smith for his help with the initial experiments.

References

[1] C. Estrada, J. Neira, and J. D. Tardós, “Hierarchical SLAM: real-time accurate mapping of large environments,” *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, August 2005.

[2] J. M. M. Montiel, J. Civera, and A. J. Davison, “Unified inverse depth parametrization for monocular SLAM,” in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[3] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[4] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): part II,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[5] N. Karlsson, E. D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, “The vSLAM algorithm for robust localization and mapping,” in *IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 24–29.

[6] J. Folkesson, P. Jensfelt, and H. Christensen, “Vision SLAM in the Measurement Subspace,” *IEEE Int. Conf. Robotics and Automation*, pp. 30–35, 2005.

[7] R. Sim and J. J. Little, “Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters,” in *IEEE/RSJ Conf. on Intelligent Robots and Systems*, 2006.

[8] T. Lemaire and S. Lacroix, “SLAM with panoramic vision,” *Journal of Field Robotics*, vol. 24, no. 1-2, pp. 91–111, 2007.

[9] R. M. Eustice, H. Singh, J. J. Leonard, M. Walter, and R. Ballard, “Visually navigating the RMS titanic with SLAM information filters,” in *Proceedings of Robotics: Science and Systems*, 2005.

[10] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[11] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. P. Gerkey, “Outdoor mapping and navigation using stereo vision,” in *Proc. of the Intl. Symp. on Experimental Robotics (ISER)*, July 2006.

[12] A. J. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *9th International Conference on Computer Vision, Nice*, 2003.

[13] E. Eade and T. Drummond, “Scalable monocular SLAM,” in *IEEE Conf. on Computer Vision and Pattern Recognition, New York*, 2006.

[14] D. Chekhlov, M. Pupilli, W. W. Mayol, and A. Calway, “Real-time and robust monocular SLAM using predictive multi-resolution descriptors,” in *2nd International Symposium on Visual Computing*, 2006.

[15] “2d3 web based literature,” URL http://www.2d3.com/, 2005.

[16] P. Newman and K. Ho, “SLAM-Loop Closing with Visually Salient Features,” *IEEE Int. Conf. on Robotics and Automation*, pp. 635–642, 2005.

[17] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, 2001.

[18] W. Grimson, “Recognition of object families using parameterized models,” *Proceedings of the First International Conference on Computer Vision*, pp. 93–100, 1987.

[19] R. Smith, M. Self, and P. Cheeseman, “A stochastic map for uncertain spatial relationships,” in *Robotics Research, The Fourth Int. Symposium*, O. Faugeras and G. Giralt, Eds. The MIT Press, 1988, pp. 467–474.

[20] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York: John Willey and Sons, 2001.

[21] A. J. Davison and D. W. Murray, “Simultaneous localization and map-building using active vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, 2002.

[22] J. Shi and C. Tomasi, “Good features to track,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, Jun 1994, pp. 593–600.

[23] J. Neira, J. D. Tardós, and J. A. Castellanos, “Linear time vehicle relocation in SLAM,” in *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, September 2003, pp. 427–433.