Image-based mapping and navigation with heterogenous robots

Gorkem Erinc, Stefano Carpin

Abstract— **We present a system composed by multiple heterogenous mobile robots that build and share an appearance based map appropriate for indoor navigation using exclusively monocular vision. Robots incrementally create online an appearance based map based on SIFT descriptors. The spatial model is enriched with additional information so that the map can be used for navigation also by robots different from those that built it. Once the map is available, navigation is performed using an approach based on epipolar geometry. The control mechanism builds upon the unicycle kinematic model, and assumes robots are equipped with a servoed camera. The validity of the proposed approach is substantiated both in simulation and on an heterogeneous multirobot system.**

I. Motivation and contribution

This paper presents our first steps towards the implementation of a an heterogeneous multi-robot system operating in indoor environment relying only on visual sensors. We show how a team of heterogenous robots can build and take advantage of a spatial model for an unknown environment based exclusively on images taken from monocular cameras. The model is then used to localize and safely navigate to a target location specified as a desired robot view. Notably, and differently from most formerly developed similar approaches, the map is built incrementally and does not require a preliminary data acquisition stage followed by an off-line lengthy map generation process. Our eventual goal is to equip these robots with mapping and navigation abilities comparable to those displayed by more sophisticated systems using laser range finders. While obviously the spatial model will be different, we strive to reach the same level of autonomy and safety in navigation. We stick to the use of monocular images because monocular cameras are cheap and represent a ready to use tool to exchange high-level information between hand-held devices and robot systems. Therefore, this appears to be a natural way to exchange information between users and robots, or to specify interesting locations for the robot to go. Our work builds upon different contributions made in the past in the fields of visual servoing, mapping, and computer vision, and achieves a new level of competence, namely heterogeneous visual based navigation. The system described in this paper builds from scratch an appearance based map capturing salient visual features detected in the environment explored by the robot. Features inserted into the map are not tied to a specific robot morphology, but are, so to speak, disembodied, inasmuch as they can be interpreted and reused also by robots with a morphology different from the the one that produced the map. The map built can then be used to localize a robot and also for navigation towards a desired target image. In Section II we shortly describe related literature in the field of spatial modeling using vision. Next, in Section III we present a method that allows a robot to move so that its perceived image matches a desired target view. The method used to organize data extracted from images into an appearance based map is presented in Section IV. Section V presents an experimental validation using simulations confirming the goodness of the navigation technique, and also an implementation of the system on an heterogenous couple of robots. Conclusions and future work are presented in Section VI.

G. Erinc and S. Carpin are with the School of Engineering, University of California, Merced (USA). E-mail: {gerinc,scarpin}@ucmerced.edu. Gorkem Erinc is supported by a CITRIS *Seed* grant entitled *Mobile Sensor Networks for Independent Living and Safety at Home.*

II. Related Work

Robot mapping using range sensors has been deeply investigated and the reader is referred to [1] for an in depth discussion of metric maps. Much closer to our study are topological maps, as originally proposed by Kuipers [2]. Often times topological maps have been extracted from metric maps using geometric concepts like generalized Voronoi diagrams [3], [4]. Less attention has been devoted to the problem of building topological maps with the only help of cameras. The concept of appearance based map has gained importance, as this kind of maps are well suited for extracting a topological structure. One of the few systems where vision is used in the context of metric and topological mapping is described in a few recent papers by Kŕose and colleagues [5], [6], where a system composed of a single robot relies on an omnidirectional camera. Omnidirectional cameras greatly simplify the problem, since omnidirectional images can be associated with the position where they have been acquired disregarding the robot heading due to their rotational invariance. For what concerns navigation, there is a vast literature on visual servoing, i.e. the use of computer vision data to control the motion of a robot. The reader is referred to [7] for a recent introduction on the topic, since a thorough discussion is not doable herein due to space constraints. However, we single out a couple of papers by Mariottini and colleagues that solve the the navigation problem for a non-holonomic robot based on *epipolar geometry* [8], [9]. As described later on, our approach extends the ideas presented therein. Their work focuses on a single robot approach and assumes the robot navigates using a data structure based upon formerly collected data. Instead, we support multiple heterogeneous robots and we acquire the data structure incrementally. A recent paper by Santosh and colleagues [10] presents an autonomous image-based navigation algorithm that utilizes an adaptive color

based frontier exploration strategy. Nevertheless, their visual control algorithm computes rototranslation between image pairs using the odometry which may not be available for all robotic systems. Finally, there has been substantial research devoted to the identification and use of visual features to ease robotic tasks. Lowe's SIFT features are probably the most used method in robotics [11]. Koseka and colleagues have investigated the problem of robot localization using SIFT features [12], but they assumed the map was given and static. Fraundorfer et. al. [13] utilized a content based image retrieval system for topological localization and mapping using a single monocular camera. The approach however relies on the bag of words scheme which implies an offline processing for vocabulary construction. A similar approach is proposed by Cummins and Newman in their recent paper [14] tackling the problem of modeling the environment using a bag of words model. Particularly relevant is the solution to the problem of detecting when a new place is visited, but it requires a timely offline processing of the training data.

III. Visual based navigation

This section describes how it is possible to implement an image based visual servoing strategy that steers a robot so that its currently perceived image matches a given target view. Even though this step is conceptually the last one, i.e. it is executed after a set of images is collected and organized into an appearance map, its discussion is presented first since it introduces some concepts related to epipolar geometry that will be used useful later on.

A. System model

It is assumed each robot in the system can be modeled as a unicycle moving on a plane and that each robot is equipped with an actuated camera that can be turned to a desired direction. Hence, the state of the robot is a vector in $\mathbb{R}^4$ and is defined as $[x\ y\ \theta\ \phi]^T$, where x and y are the Cartesian coordinates of the center of the robot, θ is the orientation of the robot with respect to x axis of the world coordinate frame, and ϕ is the orientation of the camera with respect to the robot's heading. The kinematic model is the following:

$$\dot{x} = u_1 \cos\theta \qquad \dot{y} = u_1 \sin\theta$$
$$\dot{\theta} = u_2 \qquad\qquad \dot{\phi} = u_3.$$

The input vector for the system is $U = [u_1\ u_2\ u_3]^T \in \mathbb{R}^3$ specifying forward speed (u_1), rotational speed (u_2), and the rotational speed for the camera (u_3). We assume the camera is tilted by an arbitrary known angle, and that it is possibly displaced with respect to the robot's frame.

B. Navigation between two images

Our approach to navigation exploits some results coming from epipolar geometry [15]. With reference to Fig. 1, let us consider two cameras whose relative displacement is given by a rototranslation (R, T), where $R \in \mathrm{SO}(3)$ is a rotation matrix, and $T \in \mathbb{R}^3$ a translation vector. Let K be the common intrinsic camera parameter matrix. The first camera position will be indicated as the *actual* camera position, while the second is the *desired* camera position (hence the subscripts used in the following).

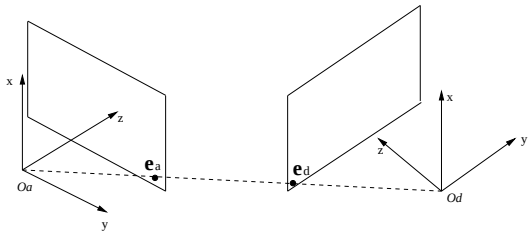

Fig. 1. The frame on the left is associated with the actual camera position, while the frame on the right indicates the desired camera position. The dotted line connecting O_a and O_d is the principal line. Its intersection with the two image planes determines the two epipoles $\mathbf{e_a}$ and $\mathbf{e_d}$.

The line connecting the centers of the reference frames associated with the two cameras is called *principal line*, and let $\mathbf{e_a}$ and $\mathbf{e_d}$ be the intersections of the principal line with the image planes of the first and second camera, respectively. Points $\mathbf{e_a}$ and $\mathbf{e_d}$ are called *epipoles*. The horizontal coordinate of $\mathbf{e_a}$ in the actual image plane will be indicated as e_{au}, and a similar notation is used for $\mathbf{e_d}$. The following 3×3 matrix is called *fundamental matrix*:

$$F = K^{-T} \widehat{T} R K^{-1} \tag{1}$$

where $\widehat{T}$ is the skew-symmetric matrix associated with the translation vector T. Under these hypothesis the following relationships hold, showing that the epipoles are in the left and right null spaces of the fundamental matrix F.

$$\mathbf{e}_d^T F = 0 \quad F\mathbf{e}_a = 0$$

Then, given F the epipoles can be computed. It is however immediate to realize that knowledge of F, or of the epipoles, does not imply the transformation (R, T) can be retrieved. As evident from Fig. 1, epipoles are invariant for translations along the principal line. The fundamental matrix F can be computed using various algorithms known in literature and it requires knowledge of a certain number of matching features between two images taken by the cameras. In our system we use the eight points algorithm, that requires eight correspondences. Corresponding features are obtained by the SIFT features inserted into a database, as described in the next sections, so from now onwards we assume that the fundamental matrix F and the epipoles can be computed when needed. In [9] Mariottini and colleagues proposed a two-stage method that controls a non-holonomic robot equipped with a fixed camera. Given the actual camera position, and a desired camera position specified as a target view, their control schema moves the robot so that the actual image eventually matches the desired image. Due to the non-holonomic constraints, the produced path significantly deviates from the shortest one between the current and the goal positions. Given that the system model we formerly assumed allows for rotations in place and includes an additional degree of freedom for the camera, we extend

their approach into a four-stage control schema which is more articulated but produces the shortest possible motion. The four-step strategy is described in the following. To shorten the discussion it is assumed that simple proportional controller laws are used to zero errors, while in practice one could use more sophisticated approaches. The k_i factors are then the proportional gains used in the various steps. The algorithm proceeds to next step when the error defined for the current step falls below a preset value. The reader is referred to Fig. 2 for a graphical illustration of the four steps.

1) From the initial configuration, applying the control law $U = [0, -k_1 e_{au}, 0]^T$ the actual epipole's horizontal coordinate e_{au} is brought to 0. The desired behavior is that the robot rotates around its z axis towards the target configuration. At the end of this step, the robot's heading will be aligned with the principal line.
2) Keeping the robot still, the camera will be rotated to align the actual camera coordinate frame with the desired camera configuration. The control law brings the horizontal coordinate of the actual epipole e_{au} to the horizontal coordinate of the desired epipole e_{du}: $U = [0, 0, -k_2 (e_{au} - e_{du})]^T$.
3) Since in the first two steps the epipoles are locked to appropriate values to produce the desired behaviors, in this step an error based on feature matches is used for feedback. The error Err_{pix} is defined as the maximum error between the horizontal image coordinates of matched features. The control law for this step is the following: $U = [-k_3 Err_{pix}, 0, 0]^T$. As the error goes to zero, the robot approaches the desired robot position while moving along the baseline. During this motion the actual camera heading is kept aligned with the desired camera configuration. At the end of this step the robot will be at the desired location and the camera will have the same view.
4) If it is desired to match the robot's heading with the camera heading, then in the last step the robot should be rotated to zero the angle between its heading and the camera orientation. Since the camera is attached to the robot, its rotation also affects the camera. Thus, the camera also rotates in order to compensate the rotation of the robot. The resulting control law is defined as $U = [0, -k_4 Err_{pix}, -k_5 \phi]^T$.

While describing the above strategy we implicitly assumed there was some overlapping between the initial actual image and the desired image. This assumption can always be enforced, and it will be briefly resumed at the end of the next section where the supporting spatial model is presented.

C. Enabling heterogenous navigation

The proposed navigation algorithm provides a viable control strategy even if the desired and actual images are taken with different camera configurations. In other words, a target image captured by a certain robot can be used by a different mobile platform for navigation purposes. This ability greatly enhances the utility of the system. For the moment we assume the robots are equipped with same cameras, i.e. same

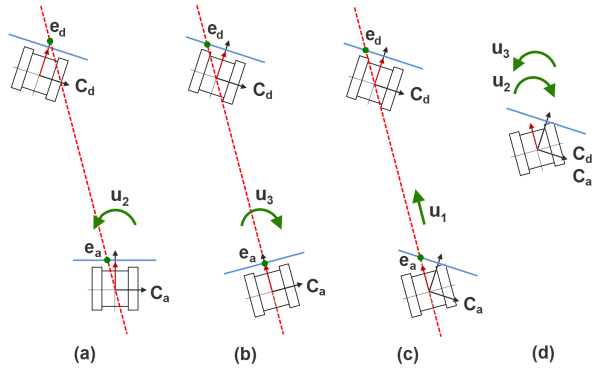

Fig. 2. This picture describes the four steps needed to bring the robot to the desired position specified by a target view.

intrinsic camera parameters, however, we plan to extend the framework to achieve full heterogeneity. Changing the elevation of the camera only shifts the projection of the observed scene along the vertical axis of the image plane and does not have any effect on horizontal coordinates of the image pixels. Since the navigation algorithm is designed based on the u-coordinates of the epipoles, the elevation of cameras does not affect the movement of e_{au} and e_{du}. Thus, the navigation algorithm can be used for heterogeneous teams of robots having cameras at different heights. Another factor that plays an important role on how the scenery is perceived is the tilt angle of the camera. In a heterogeneous team of robots some robots may have cameras tilted to a different angle than the rest of the team, or a single robot may capture images by tilting its camera to get better angles of perception during the course of its navigation. However, as the tilt angle between two cameras differs, the locations of the epipoles change. Hence, we extend the control strategy to make navigation possible between images taken by cameras with different tilt angles. The intuition behind this extension is that we want to know how the image would look like if the actual camera position had the same tilt angle as the desired camera position (or viceversa). In that case, that image could be used as the target image and the navigation algorithm could be applied without much change since the tilt angles of the cameras used to capture both images would be the same. In order to create this image we back-project each image into normalized image coordinates from pixel coordinates, virtually rotate the desired camera to the same tilt angle of the actual camera, and project the image again to pixel coordinates as observed from the rotated camera. The idea is explained below in more detail as a 3-step procedure. Since we are only interested in the extracted features, we slightly abuse the terminology and refer to the union of the extracted features as the image. Let I_t be the target image defined as $I_t = \bigcup p_i$ where p_i is an extracted feature from the image with coordinates $[u_i \ v_i]^T$ in the image plane. The intrinsic parameter matrix K can be decomposed as $K = K_s K_f$ where the two matrices, K_s and K_f, are shown below. The parameter f represents the focal length of the camera, s_x and s_y are the skew coefficients, and Pc is the principal center

of the camera

$$K_s = \begin{bmatrix} s_x & 0 & Pc_x \\ 0 & s_y & Pc_y \\ 0 & 0 & 1 \end{bmatrix}, K_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For each feature p_i in the target image we apply the following transformations:

1) The feature $p_i = [u_i\ v_i]^T$ is first transformed into $p_{pix} = [u_i\ v_i\ 1]^T$ defined in homogeneous pixel coordinates. Then, we back-project it to normalized image coordinates by using K_s

$$p_{img} = K_s^{-1}\, p_{pix}. \tag{2}$$

After the feature is transformed into normalized image plane, the z coordinate is set to the focal length of the camera

$$p'_{img} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & f \end{bmatrix} p_{img}. \tag{3}$$

2) Next, in order to simulate the perception from the virtually rotated camera, we rotate the point around the camera center with R_ψ where R is the rotation matrix defined around the camera's x axis and ψ is the tilt angle between the two cameras

$$p_{rot} = R_\psi\, p'_{img}. \tag{4}$$

At the end of the this step we achieve the normalized image coordinates of the feature as it would be perceived from the target camera if it had the same tilt angle of the actual camera.

3) In the last step we project the feature onto the CCD of our virtual camera and get the pixel coordinates we look for. The projected feature should be normalized in order to get the correct pixel values

$$p'_{pix} = K_s\, K_f\, p_{rot}. \tag{5}$$

The tilt-correction process is summarized in Fig. 3. Once all features are transformed into their new pixel coordinates, the resulting image can be used as the target image and the navigation algorithm can be applied.

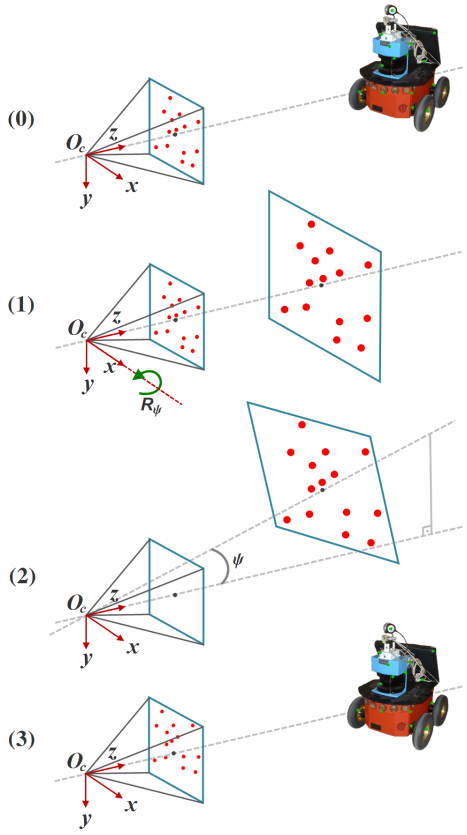

Fig. 3. The first row represents the feature extraction process. In the first step feature points are back-projected and p_{img} is computed. Then, points are rotated by ψ around x axis of the camera as illustrated in the second step and projected back to achieve the desired effect.

IV. Appearance Based Mapping and Localization

In this section we describe the spatial model used, how the map is incrementally built, and how it can be used for localization and navigation.

A. Mapping

The proposed method is built on the concept of *appearance graph*. An appearance graph is an undirected weighted graph $G = (V, E)$. Each vertex $v \in V$ represents an image captured by a monocular camera at a certain position in the workspace. For the case of monocular cameras each image is intrinsically related to the position and orientation of the camera capturing the image. However, it is important to note that this information is not encoded in the graph structure. An edge $e \in E$ connects two vertices $v_i, v_j, i \neq j$ whenever the associated images are sufficiently similar. The weight w_{ij} associated with an edge measures the similarity between images in vertices v_i and v_j. A metric to measure similarity between images will be discussed in the next subsection. A sample graph is shown in Fig. 4. It is impractical to store raw images to determine similarities between vertices and then build the graph. Therefore, a set of robust local image features characterizing the scene perceived is extracted from each image. For this purpose we have chosen SIFT descriptors. Extracted SIFT features are inserted into a feature database, *DB*, and linked to the vertex they belong to. The database stores each feature in a structure holding its 128 dimensional descriptor and the list of its appearances. An appearance object contains the feature's x and y pixel coordinates in the image plane, and other properties like its scale and direction along with the pointers to the vertices the feature belongs to. In order to account for possibly different tilt angles, all features are transformed into zero tilt angle using the procedure described above before being stored in the database. To the best of our knowledge all successful applications of appearance-based visual localization methods presented in Section II embrace a static approach for database generation which requires a training session. Striving instead to create systems that can autonomously and incrementally build their own spatial models, we opt for a dynamic database generation method where the database *DB* and the graph G

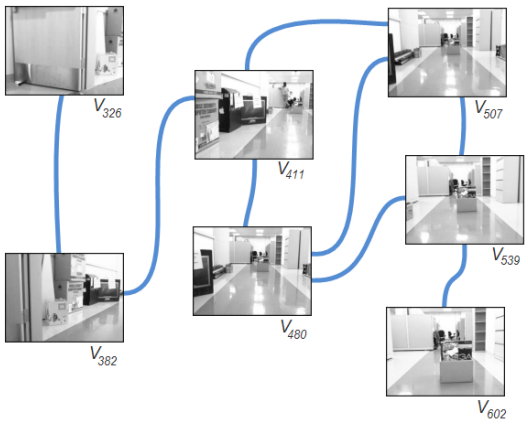

Fig. 4. The figure shows an appearance based map with 7 vertices. Edges are added between sufficiently similar images.

are created from scratch and grown with features, vertices, and edges as the robot progresses through its mission.

B. Localization

The proposed approach provides solutions to both localization and mapping problems. Given a graph and its associated feature database, the robot captures an image and extracts its SIFT features. The graph is then searched for a similar image. If a vertex with the same image already exists in the graph, that vertex is assigned to be the location of the robot. Otherwise, new features are pushed into the database, and a new vertex containing these features is created and inserted into the graph after generating necessary edges towards similar vertices. The added vertex is then returned as robot's current location. The strength of this method becomes apparent with the fact that this is the same procedure that describes mapping and all kinds of localization (tracking, global localization and kidnapped robot). As described above, the algorithm incrementally builds the graph while at each step it also provides the current location. It is obviously not practical to insert every captured image into the appearance graph. Therefore, to reduce the growing computational costs only the images that provide additional valuable information should be added. On the other hand, the more images are in the map, the similar the localization result will look to a random query image. Commonly, image selection is implemented by sampling image data uniformly over time or using a position estimate. In the literature there are more elaborate solutions like the one proposed by Booij et al. [16]. In this implementation we only add images that are similar to the last image inserted into the graph but have no more than a preset number of features in common.

1) Matching images to vertices: Given an image and its associated set of features B_i, the algorithm searches *DB* for a match to each feature $f_j \in B_i$, according to the matching method described in the next subsection. Each feature match (f_j, f_k) votes for evert vertex pointed by that feature. At the end of the voting process the vertex v_m with most votes is selected as the strongest candidate. In order to eliminate possible outliers, a robust estimation of the multi-view geometry that links the images encoded by these vertices is computed utilizing a RANSAC algorithm as described in [17]. The set of matches that voted for v_m is considered and the fundamental matrix F is computed based on a number of randomly selected tentative matches. Next, matches supporting the computed fundamental matrix are determined by checking all the tentative matches. A match (f_a, f_b) is said to be supportive of F if $dist\left(\overline{\mathbf{x}}_a, F\overline{\mathbf{x}}_b\right)^2 + dist\left(\overline{\mathbf{x}}_b, F^T\overline{\mathbf{x}}_a\right)^2 \leq \varepsilon$ where $dist(.)$ is the distance between a point and a line, ε is a predetermined constant, and $\overline{\mathbf{x}}$ is the normalized image coordinates of a feature. These steps are repeated m times and the fundamental matrix with the most support is chosen as the best possible fundamental matrix between these two vertices. A match between the image and v_m is defined if the number of matches supporting this matrix is higher than some threshold. The same process is used to decide if two vertices are sufficiently similar, and then to insert edges in the graph. The weight of each edge encoding the similarity measure is set to the number of matches supporting the fundamental matrix between two connected vertices. The majority voting schema proposed to find the closest vertex to the current image is compared with the common image-image matching method [18] and the results are shown in Fig. 5. The other method extracts features from the current image, compares them with features from each image stored in the database and the image scoring the greatest similarity measure is returned as the current location. In order to reduce the computational cost of image matching, a kd-tree is constructed from the features of each image in the database. In this performance comparison analysis, graphs with different number of images are generated and the resulting graph is used to localize a random query image. The plots in the figure correspond to the overall time required to construct the kd-trees and compute 500 localizations. The incremental construction of the kd-tree consisting of all the features in the database in our approach takes much more time comparing it to the construction process of several small kd-trees. However, as can be seen in the figure the difference between storing the kd-trees in advance or reconstructing kd-trees for each localization takes only a small percentage of the overall time required. In other words, the performance of the other approach suffers more from the localization step realized by image-image comparisons. On the other hand, the time spent to build the feature database pays off since in the majority voting schema the localization step becomes computationally very cheap. Consequently, in scenarios where number of localization calls exceeds the number of image insertions into the graph, the majority voting schema outperforms image-image comparison approach.

2) Matching features: Given a feature f_j to match against *DB*, we preliminary determine the distance to the nearest neighbor and to the second nearest neighbor, according to the L_2 norm. Lowe [19] empirically showed that any query point returning a distance ratio below 0.8, eliminates 90% of the false matches while discarding 5% of correct matches. Therefore we adopt the same criterion and pick the same

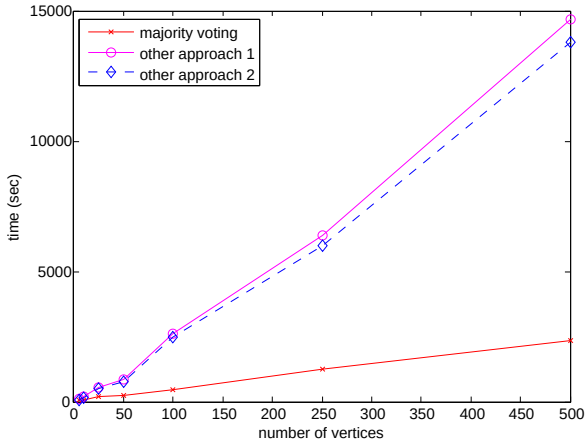

Fig. 5. Appearance graphs are constructed for different number of images, and 500 localizations to random query images are performed. The graph shows the overall time spent by the proposed majority voting schema and two image-image comparison methods. In *Other method 1* for each localization instance kd-trees for images in the database are constructed from scratch. In *Other method 2* kd-trees are computed in advance and used throughout all localization calls.

threshold. Even though this is probably the method mostly used at the moment, it exhibits some problems. The main one is that it is non-symmetric, i.e. it is possible that f_j matches a certain feature f_i, but f_i does not match f_j. This is due to the *local* nature of the metric, i.e. matching is influenced not only by the two features being considered, but also by the surrounding ones. Consequently, as the database *DB* is updated over time, formerly determined matches may become invalid due to newly added features altering the local distribution. For this reason, only features which could not be matched with any other feature in *DB* are inserted into the database, because otherwise each time a matched feature is inserted the next feature to be matched against the same feature has to be much closer than the current match. Since features are elements in $\mathbb{R}^{128}$, nearest neighbor search is a computationally expensive process. Kd-trees provide no speedup over exhaustive search for spaces with 10 or more dimensions [19]. Moreover, it is known that for a Kd-tree to be effective the search space should be well populated, i.e. $N >> 2^d$ where N is the number of points to be searched and d the dimensions of the search space. Therefore, we instead use MPNN [20], a version of Approximate Nearest Neighbor (ANN) [21] that handles fast queries with an ε approximation bound and allows the incremental growth of the Kd-tree. This last possibility is essential to implement the dynamic extension of the database of features.

C. Navigation

The appearance graph representation coupled with the presented navigation method provides the ability to autonomously navigate towards interesting places even though no metric information is stored. Given a goal image the algorithm first searches it in the appearance graph. If it fails in locating the goal image in the graph, the image is inserted and the vertex is set as the target location. If the initial position is not known the robot then localizes itself in the map by searching for the vertex with highest similarity to the actual view as described in Section IV-B. The optimal path connecting the initial and goal images is determined using Dijsktra's algorithm. Since Dijsktra's algorithm tries to minimize the cost of the path, edges in the graph are labeled with a distance measure. The distance between vertices is defined as dissimilarity between images associated with them, where non existing edges indicate an infinite dissimilarity. The distance is calculated as $\frac{1}{w_{ij}}$. The computed path is not necessarily the shortest path in Euclidean space since the planning happens in the appearance space. Therefore, the computed path favors navigation through images with a high number of features matches and avoid places where features change rapidly. Hence the robot has better recognition during the course of its navigation. Moreover, this strategy ensures sufficient overlap between successive images in the path, as requested by the navigation schema formerly illustrated.

V. Experimental results

A. Simulation

The described four step navigation algorithm is first simulated in *MATLAB*. The environment is modeled by random 3-D points posing as actual correspondences of visual features. The desired camera configuration is also randomly chosen with the constraint that enforces a minimum number of visible features. Among the places from which the desired camera configuration is in the field of view, the actual camera configuration is randomly selected in a way that at least 50% of the features are seen by both cameras. Two camera configurations can vary in terms of 3-D coordinates and pan and tilt angles. The 3-D scene is projected into virtual image planes and the feature correspondence problem is assumed to be solved. Image coordinates of the features are projected to zero tilt angle configuration for both cameras. The resulting feature sets are used to compute the fundamental matrix and the epipoles. The necessary inputs are calculated by the algorithm described in Section III. The overall behavior of the errors of a sample run are shown in Fig. 6 and it shows that even using cameras with different tilt angles it is possible to zero out the defined errors.

B. Implementation on a multirobot system

The proposed system has been implemented on a multi robot system consisting of two platorms. The first one is the P3AT platform produced by Activemedia. The other is an in-house autonomous robot developed atop the iRobot Create robotic platform. Both robots are equipped with a Philips webcam operating at a resolution of 320×240 pixels mounted on a Phidget servo providing the needed additional degree of freedom for rotating the camera. The robustness of the proposed approach has been demonstrated with the implementation of multi-robot navigation in the appearance graph. Initially each robot is driven through an arbitrary trajectory, and a sequence of images are automatically captured. For each image, SIFT features are extracted and processed as described in Section IV, thus building

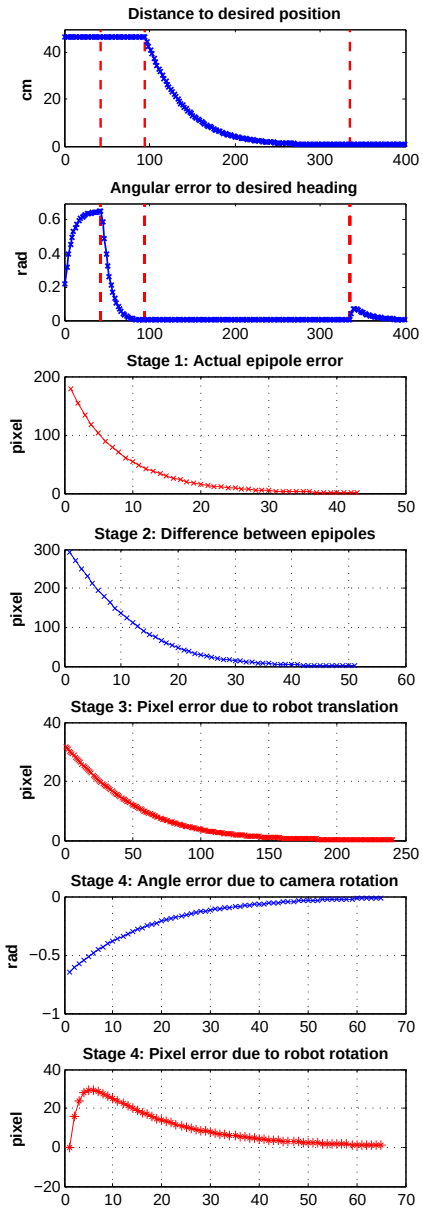

Fig. 6. Simulation results of a random run of the navigation algorithm shows the error profiles for each stage in addition to the translational and angular errors to the target configuration which differs from the actual camera configuration by 30° tilt and 12.5° pan angle. Each dotted line in the first two columns corresponds to a transitions to the next stage.

the appearance graph and the features database *DB*. Each robot is then placed on an arbitrary location close to the explored part of the environment and forced to use the graph generated by the other robot. Robots are assigned a given desired view. After an preliminary localization step each robot computes the optimum path using Dijsktra's algorithm. The resulting path is followed by navigating from one image to the next as in waypoint navigation. Due to the noise in feature coordinate estimations and outliers in feature matches, errors regarding the epipoles are obviously not as smooth as observed in simulation. Thus, by applying the action computed based on the noisy data the robot might get lost, i.e. the number of common features may fall below the minimum number required to robustly estimate the epipoles due to loss of overlap in their observed scenes. Whenever the number of matches between the actual image and the intermediate desired image falls below a threshold, the robot first assumes the failure is due to the delay in communication between the controller and the actuators and swipes its local environment for re-localization by rotating its camera. In the trials with Create, where this type of failure is commonly observed, this method successfully re-localized the robot in the graph. On the contrary, the P3AT never lost its sight in our trials. Only in one occasion where sufficient overlap between the captured and the desired image could not be recovered by the camera rotation, the Create declared itself lost and started a global localization procedure. The robot successfully localized itself in the graph by creating an edge to the next waypoint in the path and reached the target image by navigating through the remaining waypoints.

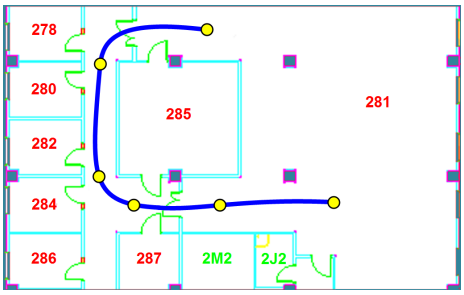

Fig. 7. The thick path illustrates the followed trajectory during the map building process. Circles indicate some of the vertices created in the appearance graph and the associated images images are presented in Fig. 8. Robots start from the bottom vertex on the right and navigate to the top vertex in the middle along the depicted path.

Fig. 7 presents the path followed in a sample run where the traversed distance is roughly 30 meters. The resulting appearance graph is built online at real-time frame rates and consists of 42 images whereas the feature database contains 7735 features. The overall graph and database construction takes 13 seconds if processed offline. Please note that the path is only for visualization purposes since the proposed method does not use any metric information. Images used to build the appearance graph are shown in the top row of Fig. 8. More precisely those are some of the images associated with the six vertices outlined along the path. The middle row shows the corresponding view obtained by the P3AT robot while navigating to reach in sequence those waypoints. The last row instead shows the same images collected by the iRobot Create. As can be seen in the last image of the sequence, both robots successfully reached the target location.

VI. Conclusion and Future Work

We have presented a system capable of incrementally building an appearance based map from scratch. The map can be used for localization and to plan a path that can be followed by a robot using an image based visual servoing

Fig. 8. With reference to Fig. 7, the top row shows the images collected while building the appearance map. Then middle row shows the corresponding view of the P3AT robot upon reaching those waypoints, while the bottom row shows the corresponding views for the iRobot Create.

approach using epipolar geometry. An interesting feature of the system we propose is that the appearance based map can be used by a set of hetereogenous robots, i.e. it can be shared among the members of an heterogenous multirobot team. The proposed approach has been implemented and validated both in simulation and on a team of two robots, effectively demonstrating the power of the proposed system. We plan to extend the system in various directions. Firstly, we will study how the map can be cooperatively built by multiple robots operating in the same environment. This problem can be solved in two ways. Each robot may continuously contribute its data to a shared model, or each robot may build its own partial map, and then occasionally merge it with some other partial model acquired by a different robot. Both these approaches will be pursued and compared. Secondly, on the practical side we will investigate the tradeoff between speed of execution and precision to evaluate the effects of moving the robots at higher velocities at the cost of occasionally losing localization or missing some of the steps in the navigation strategy. Furthermore, we will study the stability of the proposed 4-step servoing algorithm and explore how to choose the control parameters to guarantee the convergence of designed errors at each step. Finally, from a computational point of view it is evident that better strategies are needed to establish correspondences between SIFT features. The problems we outlined are usually unimportant when the set of feature is acquired upfront and then processed in one batch. Instead, for the case of incremental construction it is evident that more sophisticated methods need to be investigated to overcome the limitations outlined.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006.

[2] B. Kuipers, "Modeling spatial knowledge," *Cognitive science*, vol. 2, pp. 129–153, 1978.

[3] H. Choset and K. Nagatani, "Topological simultanous localization and mapping (SLAM): Toward exact localization without explicit localization," *IEEE Transaction on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, 2001.

[4] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[5] Z. Zivkovic, O. Booij, B. Kröse, E. Topp, and H. Christensen, "From sensors to human spatial concepts: An annotated data set," *IEEE Transaction on Robotics*, vol. 24, no. 2, pp. 501–505, 2008.

[6] Z. Zivkovic, , O. Booij, and B. Kröse, "From images to rooms," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 411–418, 2007.

[7] F. Chaumette and S. Hutchinson, "Visual servoing and visual tracking," in *Handbook of Robotics*. Springer, 2008, ch. 24, pp. 563–583.

[8] G. L. Mariottini and D. Prattichizzo, "Epipole-based visual servoing for nonholonomic mobile robots," in *Proceedinds of the IEEE International Conference Robotics and Automation (ICRA)*, 2004.

[9] G. L. Mariottini, G. Oriolo, and D. Prattichizzo, "Image-based visual servoing for nongolonomic mobile robots using epipolargeometry," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 87–100, 2 2007.

[10] D. Santosh, S. Achar, and C. V. Jawahar, "Autonomous image-based exploration for mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[11] D. Lowe, "Distinctive image features from scale-invariants keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] J. Košecká, F. Li, and X. Yang, "Global localization and relative positioning based on scale-invariant keypoints," *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 27–38, 2005.

[13] F. Fraundorfer, C. Engels, and D. Nister, "Topological mapping, localization and navigation using image collections," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[14] M. Cumming and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[15] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003.

[16] O. Booij, Z. Zivkovic, and B. Krose, "Sampling in image space for vision based slam," in *Proceedings of the Inside Data Association Workshop during the Robotics: Science and Systems Conference (RSS)*, 2008.

[17] L. Maohai, H. Bingrong, and L. Ronghua, "Novel method for monocular vision based mobile robot localization," in *Proceedings of International Conference on Computational Intelligence andSecurity*, 2006.

[18] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose, "Navigation using an appearance based topological map," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

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

[20] A. Yershova and S. LaValle, "Improving motion-planning algorithms by efficient nearest-neighbor searching," *IEEE Transaction on Robotics*, vol. 23, pp. 151–157, 2007.

[21] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu, "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.