Hierarchical localization by matching vertical lines in omnidirectional images

Abstract— **In this paper we propose a new method for robot localization using an omnidirectional camera. It efficiently combines in a hierarchical algorithm the possibility of obtaining appearance-based (topological) localization and metric one. Given the current uncalibrated image seen by the robot, we obtain its localization based on a visual memory of reference images with known locations. First aim is to find the room where the current image is taken, in order to get it we look for the most similar image in the visual memory: First, a global color-based filter is applied to get a limited number of candidate images. Next, a more accurate similarity measure is obtained with a pyramidal matching algorithm. To this end, several histograms are built characterizing the image, using descriptors of radial line support regions. The last step in the method is to get the metric localization. The lines are matched to estimate the 1D radial trifocal tensor, which provides robust matches and the relative location between views. Compared to other vision-based localization methods, the one proposed here has the advantage that it gets accurate metric localization based on a minimal reference image set. Moreover, thanks to the linear wide baseline features, the method is insensitive to illumination changes and occlusions, while keeping the computational load small. We show experiments with real images from two omnidirectional image data-sets. We evaluate the performance of the method to localize the robot, both finding the current room and the metric localization.**

Index Terms: - topological + metric localization, hierarchical methods, radial line matching, omnidirectional images

I. Introduction

For some robotic tasks, such as navigation or obstacle avoidance, the robot needs to continuously localize itself to correct the trajectory errors due to odometry inaccuracy, slipping,... At this point it is clear we need accurate metric localization information. However, the same accuracy in localization is not always needed. Sometimes topological one is enough (e.g. saying in which room we are) and indeed this is also a more intuitive information to communicate with humans. In this paper we present a hierarchical method for robot localization, combining topological and metric information, using a Visual Memory (VM). It consists of a database of sorted omnidirectional reference images (we know *a priori* the room they belong to and their relative positions).

Omnidirectional vision has become widespread in the last years, and has many well-known advantages as well as extra difficulties compared to conventional images. There are many works using all kind of omnidirectional images, e.g. a map-based navigation with images from conic mirrors [1] or localization based on panoramic cylindric images composed from mosaics of conventional ones [2].

We focus our work on hierarchical localization, a quite interesting subject because of the possibility of combining different kinds of localization and also because its higher efficiency allows to handle better big databases of reference images. The localization at different levels of accuracy with omnidirectional images was previously proposed by Gaspar *et al.* [3]. Their navigation method consists of two steps: a fast but less accurate topological step, useful in long-term navigation, is followed by a more accurate tracking-based step for short distance navigation. Also in [4], a hierarchical localization with omnidirectional images is performed. This time is based on the Fourier signature and gives an area where the robot is located around reference images from one environment. It is computationally efficient but it seems to require a very dense database to achieve precise localization. Both works take advantage of a hierarchical approach, but they do not go up to metric localization. One of the advantages of the method proposed here is that it gets this, combining topological and metric localization.

Our method performs a hierarchical localization in three steps, each one more accurate than the previous but also computationally more expensive. From the two first steps we get the topological localization (the room where we are), with a method inspired by the work of Grauman *et al.* [5] for object recognition using *pyramid matching kernels*. In the final and third step, we continue to get a metric localization relative to the VM. For that we use local feature matches in three views to robustly estimate a 1D trifocal tensor. The relative location between views can be obtained with an algorithm based on the 1D trifocal tensor for omnidirectional images as shown in [6]. The features used are scene vertical lines. They are projected in the omni-images as radial ones, supposing vertical camera axis, and allow us to estimate the center of projection in the image. The 1D projective cameras have been previously used e.g. for self calibration [7] and the 1D trifocal tensor, that explains the geometry for three of those camera views, was presented in [8]. Correcting the radial distortion is another recent application of the 1D tensor for omnidirectional images [9].

The line features and the different steps of the localization method are detailed respectively in sections II and III, and in section IV we can see several experiments proving the effectiveness of the different steps of our method with real images.

II. Lines and their Descriptors

The number of features proposed for matching has increased largely over the last few years, where SIFT [10] has become very popular. Yet, in this work, we have chosen vertical lines with their support regions as features. These lines show several advantages when working with omnidirectional images (robustness against partial occlusion, easy and fast extraction, more invariant to affine geometrical deformations,...). Each line is described by a set of descriptors that characterize it in the most discriminant way possible, although it is necessary to find a balance between invariance and discriminative power, as the more invariant the descriptors are, the less discriminant they become. In this section, we first explain lines extraction process (section II-A), whereafter we shed light on the kind of descriptors used to characterize them (section II-B).

A. Line Extraction

In this work we use lines and their Line Support Region (LSR) as primitives for the matching. They are extracted using our implementation of the *Burns algorithm* [11]. Firstly, this extraction algorithm computes the image brightness gradient and after it segments the image into line support regions, which consist of adjacent pixels with similar gradient direction and gradient magnitude higher than a threshold. Secondly a line is fitted to each of those regions using a model of brightness variation over the LSR [12]. We can see some extracted LSR in Fig. 1. As mentioned before, we use only vertical lines. We suppose that the optical axis of the camera is perpendicular to the floor and that the motion is done on a plane parallel to the floor. Under these conditions, these lines are the only ones that keep their straightness, being always projected as radial lines in the image. Therefore, they are quite easy to find and we can automatically obtain from them the center of projection, an important parameter to calibrate this kind of images. We estimate it with a simple *ransac*-based algorithm that checks where the radial lines are pointing (Fig. 1). The real center of projection is not coincident with the center of the image, and the more accurate we find its real coordinates the better we can estimate later the multi-view geometry and therefore, the robot localization from it.

B. Line Descriptors

After extracting the image features, next step is to compute a set of descriptors that characterize them as well as possible. Most descriptors will be computed separately over each of the sides in which the LSR is divided (Fig.1).

1) Geometric Descriptors: We obtain two geometric parameters from the vertical lines. Firstly, its direction δ, a boolean indicating if the line is pointing to the center or not. This direction is established depending on which side of the line is the darkest region. The second parameter is the line orientation in the image $(\theta \in [0, 2\pi])$.

2) Color Descriptors.: We have worked with three of the color invariants suggested in [13], based on combinations of the generalized color moments,

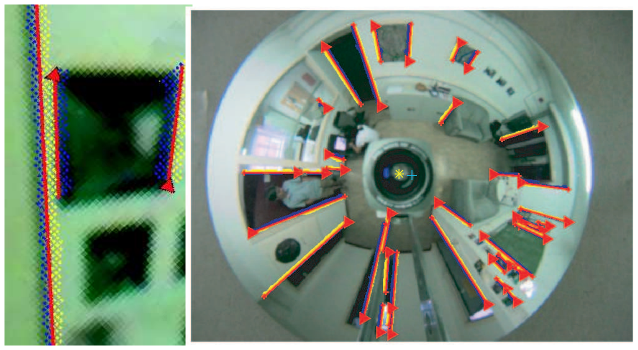


Fig. 1. Left: detail of some LSRs, divided by their line (the darkest side on the right in blue and the lighter on the left in yellow). A red triangle shows the lines direction. Right: all the radial lines with their LSR extracted in one image. The estimated center of projection is pointed with a yellow star (∗) and the center of the image with a blue cross (+).

$$M_{pq}^{abc} = \int\int_{LSR} x^p y^q [R(x,y)]^a [G(x,y)]^b [B(x,y)]^c dxdy. \quad (1)$$

being M_{pq}^{abc} a generalized color moment of order $p+q$ and degree $a+b+c$ and $R(x,y), G(x,y)$ and $B(x,y)$ the intensities of the pixel (x,y) in each RGB color band centralized around its mean. These invariants are grouped in several classes, depending on the schema chosen to model the photometric transformations. The ones we found more suitable for this application are *Type S* (scale invariant), and *Type SO* (scale and offset invariant). From all of them, we analyzed which ones were more representative using a Principal Component Analysis (PCA). Keeping a compromise between complexity and discriminative power we select these 3 descriptors:

$$DS^{RG} = \frac{M_{00}^{110} M_{00}^{000}}{M_{00}^{100} M_{00}^{010}} \quad DS^{RB} = \frac{M_{00}^{101} M_{00}^{000}}{M_{00}^{100} M_{00}^{001}} \quad DS^{GB} = \frac{M_{00}^{011} M_{00}^{000}}{M_{00}^{010} M_{00}^{001}} \quad (2)$$

3) Intensity Frequency Descriptors.: Finally we also use as descriptors the first seven coefficients of the Discrete Cosine Transform, DCT, over the intensity signal (I) of the LSR of each line. The DCT is a well known transform in the areas of signal processing [14] and widely used for image compression.It is possible to estimate the number of coefficients necessary to describe a certain percentage of the image content. For example with our test images, seven coefficients are necessary on average to represent 99% of the intensity signal over a LSR.

We have chosen 22 descriptors, but to deal with big amounts of images, it would be good to decrease this number. Also using PCA analysis we reduced it to a 12 line descriptors vector, that we will use for the Pyramidal matching: *3 color descriptors* $(DS^{RG}, DS^{RB}, DS^{GB})$ and *2 frequency descriptors* (DCT_1, DCT_2), computed on each side of the LSR, and *2 geometric properties* (orientation θ and direction δ). However, for the individual line matching we propose to use also the 5 following coefficients of the DCT in both sides of the LSRs, 22 descriptors per line, because there this size is not that critical as we will deal only with 3 images.

III. Hierarchical Localization Method

As said before, our proposal consists of a hierarchical robot localization in three steps. With this kind of localization, we can deal with large databases of images in an acceptable time. This is possible because we avoid evaluating all the images in the entire data set in the most computationally expensive steps. Note also that the VM stores already the extracted image features and their descriptors. The matches between adjacent images can be also pre-computed to save time during navigation. The three steps of our method, schematized in Fig. 2, are as follows:

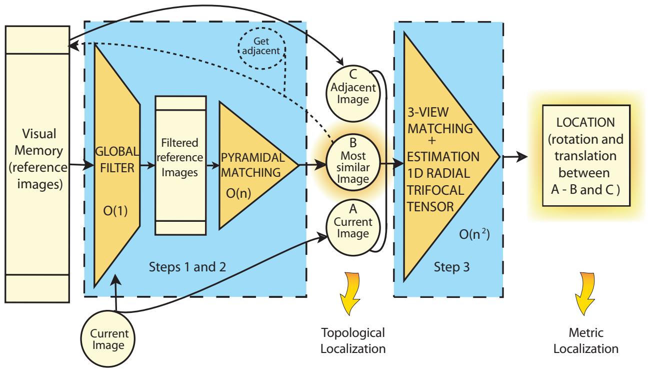


Fig. 2. Diagram of the three steps of the hierarchical localization.

A. Global Descriptor Filter

In a first step, we compute three color invariant descriptors over all the pixels in the image. The descriptor used is DS, described previously in section II-B.2. We compare all the images in the VM with the current one and we discard the images with more than 60% difference in those descriptors previously normalized.

B. Pyramidal Matching

This step finds which is the most similar image to the current one. For this purpose, the set of descriptors of each line is used to implement a *pyramid matching kernel* [5]. The idea consists of building for each image several multi-dimensional histograms (each dimension corresponds to one descriptor), where each line feature occupies one of the histogram bins. The value of each line descriptor is rounded to the histogram resolution, which gives a set of coordinates that indicates the bin corresponding to that line.

Several levels of histograms are defined. In each level, the size of the bins is increased by powers of two until all the features fall into one bin. The histograms of each image are stored in a vector (pyramid) ψ with different levels of resolution.

The similarity between two images, the current (c) and one of the visual memory (v) is obtained by finding the intersection of the two pyramids of histograms

$$S(\psi(c),\psi(v)) = \sum_{i=0}^{L} w_i N_i \ , \tag{3}$$

with N_i the number of matches (LSRs that fall in the same bin of the histograms) in level i, and w_i the weight for the matches in that level, that is the inverse of the current bin size (2^i). This distance is divided by a factor determined by the self-similarity score of each image, in order to avoid giving advantage to images with bigger sets of features, so the distance obtained is

$$S_{cv} = \frac{S(\psi(c),\psi(v))}{\sqrt{S(\psi(c),\psi(c))\ S(\psi(v),\psi(v))}} \ . \tag{4}$$

Once the similarity measure between our actual image and the VM images is obtained, we choose the one with highest S_{cv} as the most similar. Based on the annotations of this chosen image, we know in which room the robot is currently.

C. Line Matching and Metric Localization

For some applications, a more accurate localization information than the room is needed. In this case, we continue with the third step of the hierarchical method, where we need to obtain individual matches to compute the localization through a trifocal tensor.

1) Line matching algorithm.: Once we have the most similar image, first we find two-view line matches between the current and the most similar, and between that most similar and its adjacent in the VM. The two-view line matching algorithm we propose works as follows:

• First decide which lines are compatible with each other, checking certain constraints in the direction, the relative rotation and that at least the individual descriptor distances in one side of the LSR are under an established threshold.

• Compute a unique distance, from all the descriptors ones, between each pair of compatible lines. Mahalanobis distances are the most commonly used, however a lot of training is needed to compute the required covariance matrix with satisfying accuracy. Alternatively, we have tried to normalize those distances dividing the descriptors by the maximum value of each one. Theoretically, this may not be as correct as Mahalanobis distances, but in practice it works well and it is more adaptable to different queries. So it is enough to normalize the values and to have the distances of the different kinds of descriptors in similar scales, so we can sum them. We also apply a correction or penalty to increase the distance with the ratio of descriptors (N_d) whose differences were over their corresponding threshold in the compatibility test. So the final distance we consider between two lines, i and j, will be

$$d_{ij} = (d_{RGB}^{Min} + d_{DCT}^{Min})(1 + N_d), \tag{5}$$

where d_{RGB}^{Min} and d_{DCT}^{Min} are the smallest distances (in color and intensity descriptors respectively) of the two LSR sides.

• Perform a nearest neighbour matching between compatible lines.

• Apply a topological filter to the matches that helps to reject non-consistent matches. It is an adaptation for radial lines in omnidirectional images of the proposal in [15]. This improves the robustness of this initial matching.

• Run a re-matching step, that takes into account the fact that the neighbours to a certain matched line should rotate in the image in a similar way from 1 view to the other.

2) 1D Radial Trifocal Tensor and Metric Localization.: It is well known, that to solve the structure and motion problem from lines at least three views are necessary. Then, after computing two-view matches between the two couples of images explained before, we extend the matches to three views intersecting both sets of matches and checking a consistent behaviour.

There is a scheme in Fig. 3 showing a vertical line projected in the three views used and the location parameters we want to estimate $(\alpha_{21}, t_{x21}, t_{y21}, \alpha_{31}, t_{x31}, t_{y31})$.

We apply a robust method (*ransac* in our case) to the three-view matches to simultaneously reject outliers and estimate a 1D radial trifocal tensor. The orientation of the lines in the image is expressed as 1D homogeneous coordinates ($\mathbf{r} = [\cos\theta, \sin\theta]$) to estimate the tensor. The trilinear constraint, imposed by a trifocal tensor in the coordinates of the same line $\mathbf{v}$ projected in three views $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$, is as follows,

$$\sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{2} T_{ijk} \, \mathbf{r_1}_{(i)} \, \mathbf{r_2}_{(j)} \mathbf{r_3}_{(k)} = 0 \qquad (6)$$

where T_{ijk} $(i, j, k = 1, 2)$ are the eight elements of the $2 \times 2 \times 2$ trifocal tensor and subindex $_{(\cdot)}$ are the components of vectors $\mathbf{r}$.

With five matches and two additional constraints defined for the calibrated situation, we have enough to compute a 1D trifocal tensor [8]. In our case we have a radial 1D trifocal tensor, and it is known than from it we can get a robust set of matches, the camera motion and the structure of the scene [6]. In general, the translation between views is estimated up to a scale, but with the *a priori* knowledge that we have (we know the relative position of the two images in the visual memory) we can solve it exactly.

IV. Experiments and Results

In this section we show the performance of the algorithms from previous sections for detecting the current room (IV-A), for line matching (IV-B) and for metric localization (IV-C).

We worked with two different data-sets, *Almere* and our own (named *data-set 2*), because from this second one was available some extra ground truth data, that was convenient to measure the errors in the localization.

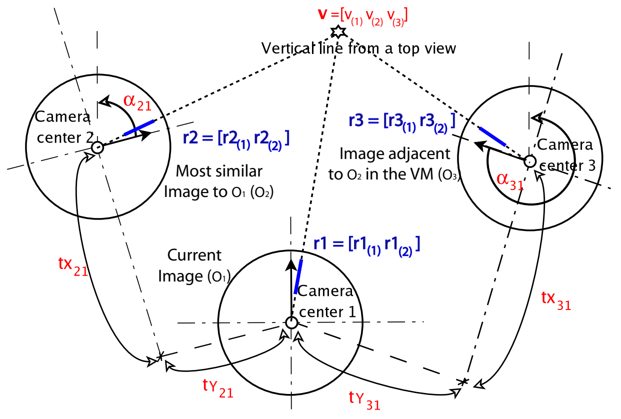


Fig. 3. Motion between images and landmark parameters

From the *Almere* data-set, we have extracted the frames from the low quality videos from the rounds 1 and 4 (2000 frames extracted in the first, and 2040 in the second). We kept just every 5 frames, selecting half for the visual memory (every 10 frames starting from number 10) and the other half for testing (every 10 frames starting from number 5).

The other visual memory has 70 omnidirectional images (640x480 pixels). 37 of them are sorted, classified in different rooms, with between 6 and 15 images of each one (depending on the size of the room). The rest corresponds to unclassified ones from other rooms, buildings or outdoors.

In Fig. 4 we can see an scheme of both databases, the second one with details of the relative displacements between them. All images have been acquired with an omnidirectional vision sensor with hyperbolic mirror.

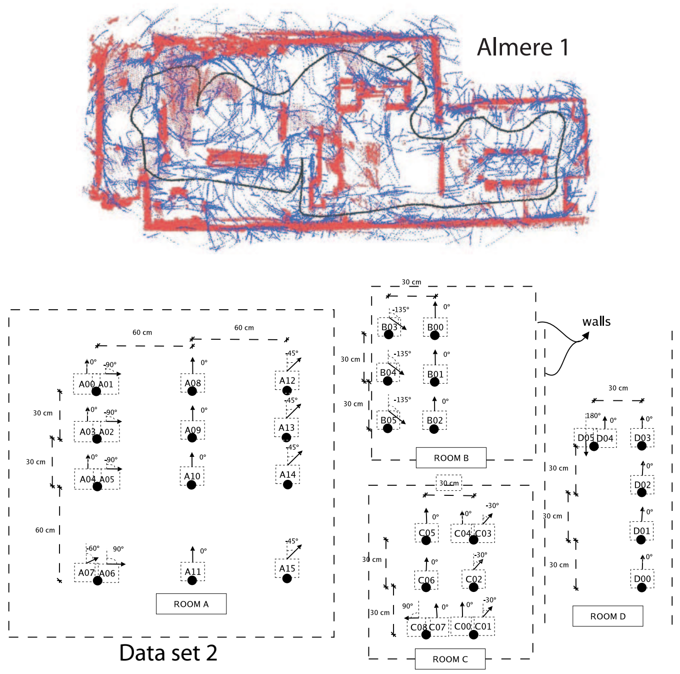


Fig. 4. Grids of images in rooms used in the experiments. Top: *Almere* data-set. Bottom: *Data-set 2*

The present implementation of our method is not optimized for speed, so timing information would be irrelevant. However, we have analyzed the time complexity in each of the three steps of the method.

- The first step (Section III-A) has constant complexity with the number of features in an image.
- The second step algorithm (Section III-B) is linear with the number of features.
- The third step process (Section III-C) has quadratic complexity with the number of features.

Indeed, in each step we have a computational cost linear with the number of images remaining on it. That is why is so important to keep the minimal number of images for the last and most expensive step.

A. Localizing the current room

In this experiment we ran the two first steps of the method. We evaluated the topological localization for all possible tests in *Data-set 2* (removing 1 test-image and comparing against all the rest). For the experiments with the *Almere* data-set we used the frames left for testing from the round 1, and also the same frame-number from the round 4.

First step: Before computing a similarity measure to the full data-set, we applied the global filter. It was expected to reject a big amount of very different images, leaving more or less the ones corresponding to the same room and a relative small percentage (half or less) of *outliers* (images from other rooms). The efficiency of this filter is quite good. We observed in the experiments with the *Data-set 2* that an average of 70% of the images were rejected, from where only an average of 1.4% of the images were rejected being correct (false negatives). Using the *Almere* images the results were a little worse (around 80% rejected from where 20% were false negatives). This decreasing in the performance become more accentuated when classifying images from the round 2, with many occlusions, where sometimes we got too many false negatives. In the worst cases all good solutions were rejected already in this first step, making impossible for the next steps to find a correct classification.

In those average values, indeed there are cases with few rejected images in this step, so the Pyramidal matching will have to deal with many candidates, while other times very few candidates passed (in 16% of the cases less than 5), making easier the task of the Pyramidal matching.

Second step: The goal of this step is to choose the most similar image to the current one, that for us will be correct if it is in the proper room.

Firstly, for building the pyramid of histograms, indeed it is necessary to perform a suitable scaling of the descriptors, as they have values in very different ranges. What we did is to scale the descriptors considered more important (color and geometric ones) between 0 and 100, and the rest between 0 and 50, so we achieve that the accuracy of the most important descriptors decreases in a slower way than the accuracy of the others, in each level of the pyramid. Here, from the *Almere* set chosen for the VM, we will use only a fifth (every 50 frames: 50-100-150-...), as the images were still quite close and to localize the room it is not necessary so much accuracy. The *data-set 2* was already thought for this experiment so the images are already more separated, so we kept them all for the VM. In *Data-set 2,* the VM images were more equally distributed in all the rooms and without the conflicts in the *Almere* one, e.g. when the robot is between two rooms. Due to this fact, and that the VM from Almere trajectory was built automatically from the video sequence, without any post-processing such as clustering nodes, we can observe a lower performance with the *Almere* data-set.

The results when choosing the most similar image are shown in Table I for all experiments, the one with *Data-set 2* and the two using *Almere* data (Almere$1\propto1$ indicates we are classifying images from round 1 against the VM of other images from round 1, while Almere$4\propto1$ means that we are classifying images from round 4 against the VM made from round 1). The image chosen as most similar ($1\ Ok$ in the table) was around 85% correct in the *Almere*$1\propto1$ test and always correct with the *Data-set 2*. In the test *Almere*$4\propto1$ the performance was lower, because the first step was working worse, making sometimes that even all possible correct images were rejected by it. This can be due to big occlusions, that made the global appearance based step to fail more often. This proofs the importance of using local features to deal with occlusions. If we skip the first step that is based in global appearance, the performance will increase again but also the computations will grow quite a lot. We must find a compromise between complexity and correctness.

Almere$1\propto1$		Almere$4\propto1$		*Data-set 2*	
1 Ok	3 Ok	**1 Ok**	3 Ok	**1 Ok**	3 Ok
85%	53%	55%	34%	100%	94 %

TABLE I

FINDING THE MOST SIMILAR IMAGE IN THE VM. $1Ok$: most similar found correct. $3Ok$: 3 first images sorted with the similarity score correct.

1) Pyramidal vs. Individual Matching to get a similarity score.: The results of the algorithm we develop for matching lines in section III-C could also be used for searching the most similar image. For this purpose, we can compute a similarity score that depends on the matches found (n) between the pair of images, weighted by the distance (d) between the lines matched. It has to take also into account the number of features not matched in each image (F_1 and F_2 respectively) weighted by the probability of occlusion of the features (P_o). The defined dissimilarity (DIS) measure is

$$DIS = n\ d + F_1(1 - P_o) + F_2(1 - P_o)\ . \quad (7)$$

We have compared this way and the Pyramidal matching method to select the most similar image in the previous experiments of this section and we noticed the pyramidal method seems more suitable. The correctness in the results was quite similar for both methods. Using the *Data-set 2* the individual matchings similarity measure the percentage of good choices decreased from 100% to 92%, and also the percentage when checking the 3 first images chosen was worse, from 94% to 83%. With the *Almere* data the correct results were 85% or 90% for selecting the most similar and 53% or 52% when checking the 3 first classified ones. However, the main advantage of the pyramidal method is that it is a linear algorithm, so it is much faster (in the experiments half of the time than the algorithm based on the individual matches). Therefore, we decided to use the pyramidal matching to select the most similar image.

B. Line Matching Results

Here we show the performance of the two-view line matching algorithm developed for the last step of the process. In these experiments we obtained individual matches between each image and the most similar selected by the Pyramidal matching (using the results from the experiments of previous section IV-A, when a current image was compared against the rest of the VM). We got between 6 and 35 matches for the different tests, depending on the rooms, from which only an average of 10% were wrong.

The better performance we get with wide-baseline images, the less density needed in the visual memory of each room. So this step to get wide-baseline matches is the *key-point* to be able to work with a minimal database. In Fig.5 we can see an example of the line matches using images more separated, not with the image that was selected as most similar but with some further one.

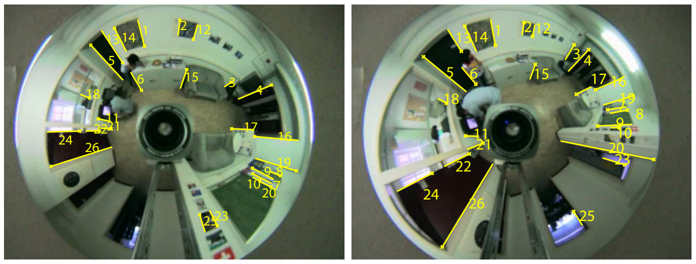


Fig. 5. Matches with *more separated images* than the couples obtained in the Pyramidal matching - Room A (A01-A07)- 26 matches/ 7 wrong.

C. Metric Localization Results

As explained before, we need three views to get the localization. Therefore, once we have two-view matches, between current image and the most similar selected, and between this one and one of its neighbours in the VM, we get three-view matches from the intersection of those sets. We make these three-view matches a robust set simultaneously to the 1D radial tensor estimation. In Fig. 6, there are some typical results obtained after applying the whole method in some examples from *Almere* data-sets. In *Almere 1* $\propto$*1* the query image is from round 1, while in *Almere 4* $\propto$*1* is from round 4. The localization parameters seem stable and consistent, however we did not have ground truth to evaluate them. We show in TABLE II the metric localization results with a similar test done with images from *Data-set 2*, where we had ground truth. Notice the small errors (around 1^o), specially if we take into account the uncertainty of our ground-truth, that was obtained with a metric tape and goniometer. We can find some other experiments performed with more separated images to check the reach of the method to get the metric localization from three omnidirectional views in [6].

Location Param.	Rotation α_{21}	Rotation α_{31}	Transl. direction t_{21}	Transl. direction t_{31}
reference	0^o	0^o	0^o	-26.5^o
manual	-1.34^o	-0.90^o	0.65^o	-26.35^o
automatic	-1.38^o	-0.94^o	0.68^o	-26.44^o
(std)	(0.07)	(0.07)	(0.13)	(0.17)

TABLE II
METRIC LOCALIZATION FROM A *Data-set 2* TEST. *reference*: reference data *manual*:results with manual matches; *automatic*: mean and standard deviation for results from automatic matches (50 executions per test).

V. CONCLUSIONS

In this work, we have proposed a hierarchical mobile robot localization method. Our three-step approach uses omnidirectional images and combines topological (which room) and metric localization information. The localization task computes the position of the robot relative to a grid of training images in different rooms. After a rough selection of a number of candidate images based on a global color descriptor, the best resembling image is chosen based on a pyramidal matching with wide baseline local line features. This enables the grid of training images to be less dense. A third step involving the computation of the omnidirectional trifocal tensor yields the metrical coordinates of the unknown robot position. Our experiments, with two different data-sets of omnidirectional images, show that this approach is fast and gives a good accuracy.

REFERENCES

[1] Y. Nishizawa Y. Yagi and M. Yachida. Map-based navigation for a mobile robot with omnidirectional image sensor copis. In *IEEE Trans. Robotics and Automation*, pages 634–648, October 1995.

[2] Chu-Song Chen, Wen-Teng Hsieh, and Jiun-Hung Chen. Panoramic appearance-based recognition of video contents using matching graphs. *IEEE Trans. on Systems Man and Cybernetics, Part B*, 34(1):179–199, 2004.

[3] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Trans. on Robotics and Automation*, 16(6):890–898, 2000.

[4] E. Menegatti, A. Pretto, and E. Pagello. Testing omnidirectional vision-based monte-carlo localization under occlusion. In *Proc. of Int. Conf. on Intelligent Robots and Systems, IROS 2004*, 2004.

[5] K. Grauman and T. Darrell. The pyramid match kernels: Discriminative classification with sets of image features. In *Proc. of the IEEE Int. Conf. on Computer Vision*, 2005.

[6] C. Sagüés, A.C. Murillo, J.J. Guerrero, T. Goedemé, T. Tuytelaars, and L. Van Gool. Localization with omnidirectional images using the radial trifocal tensor. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2006.

[7] Olivier Faugeras, Long Quan, and Peter Sturm. Self-calibration of a 1d projective camera and its application to the self-calibration of a 2d projective camera. In *Proc. of the 5th European Conference on Computer Vision, Freiburg, Germany*, pages 36–52, June 1998.

[8] K. Åström and M. Oskarsson. Solutions and ambiguities of the structure and motion problem for 1d retinal vision. *Journal of Mathematical Imaging and Vision*, 12:121–135, 2000.

[9] S. Thirthala and M. Pollefeys. The radial trifocal tensor: A tool for calibrating the radial distortion of wide-angle cameras. In *Proc. of Computer Vision Pattern Recognition*, 2005.

[10] David G. Lowe. Distinctive image features from scale-invariant key-points. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

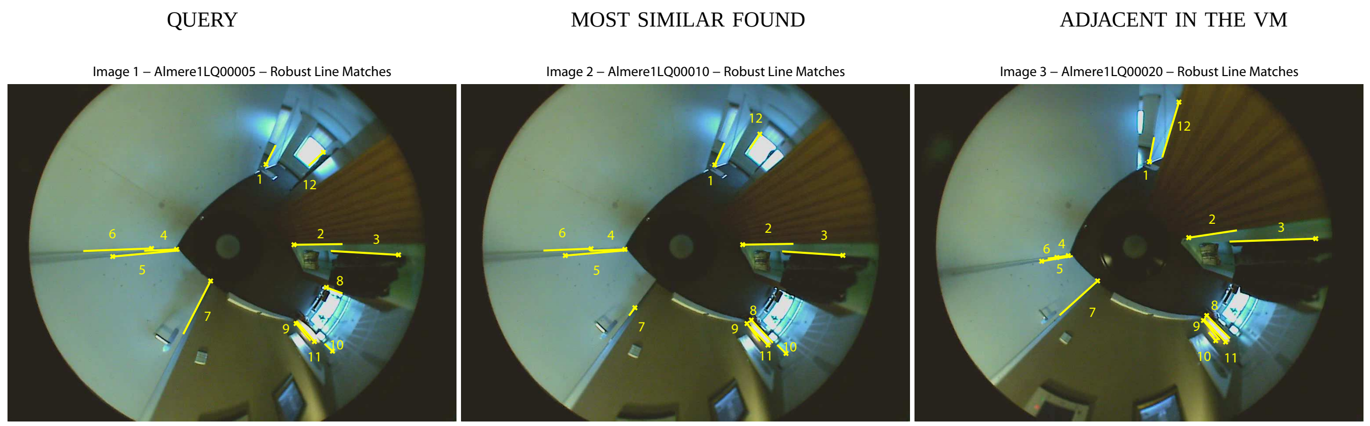
[11] J.B. Burns, A.R. Hanson, and E.M. Riseman. Extracting straight lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(4):425–455, 1986.

[12] Omitted for double blinded version.

[13] F. Mindru, T. Tuytelaars, L. Van Gool, and T. Moons. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94(1-3):3–27, 2004.

[14] K.R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press, Boston, 1990.

[15] H. Bay, V. Ferrari, and L. Van Gool. Wide-baseline stereo matching with line segments. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, June 2005.


QUERY
MOST SIMILAR FOUND
ADJACENT IN THE VM
Image 1 – Almere1LQ00005 – Robust Line Matches
Image 2 – Almere1LQ00010 – Robust Line Matches
Image 3 – Almere1LQ00020 – Robust Line Matches

Almere 1$\propto$1. 19 initial matches with 11 wrong. After robust estimation: 12 robust matches with 4 wrong.

Localization Parameters

α_{21}	α_{31}	t_{21}	t_{31}
4 ^{o}	16 ^{o}	45 ^{o}	44^o

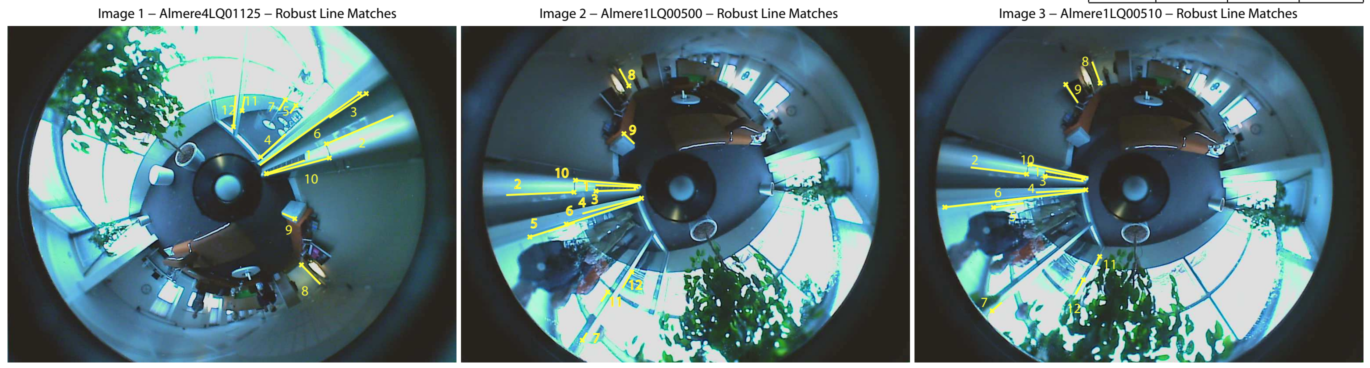


Image 1 – Almere4LQ01125 – Robust Line Matches
Image 2 – Almere1LQ00500 – Robust Line Matches
Image 3 – Almere1LQ00510 – Robust Line Matches

Almere 4 $\propto$1. 15 initial matches with 5 wrong. After robust estimation: 12 robust matches with 3 wrong.

Localization Parameters

α_{21}	α_{31}	t_{21}	t_{31}
161 ^{o}	150^o	114 ^{o}	132^o

Fig. 6. Examples of triple robust matches obtained between the current position (first column image) and 2 reference images (second and third columns), together with the localization parameters obtained.