DIIS - I3A
C/ María de Luna num. 1
E-50018 Zaragoza
Spain

Internal Report: 2006-V11

Hierarchical localization by matching vertical lines in omnidirectional images

A.C. Murillo, C. Sagüés, J.J. Guerrero, T. Goedemé, T. Tuytelaars and L. Van Gool

If you want to cite this report, please use the following reference instead:
Hierarchical localization by matching vertical lines in omnidirectional images, A.C. Murillo, C. Sagüés, J.J. Guerrero, T. Goedemé, T. Tuytelaars and L. Van Gool, *Workshop "From Sensors to Human Spatial Concepts"* pages 13-19, IEEE/RSJ Int. Conference on Intelligent Robots and Systems) , Beijing - China, October 2006.

This work was supported by project MCYT/FEDER - DPI2003 07986.

Hierarchical localization by matching vertical lines in omnidirectional images

A.C.Murillo, C.Sagüés, J.J.Guerrero
DIIS - I3A, University of Zaragoza, Spain
{acm, csagues, jguerrer}@unizar.es

T. Goedemé, T. Tuytelaars, L. Van Gool
PSI-VISICS, University of Leuven, Belgium
{tgoedeme, tuytelaa, vangool}@esat.kuleuven.be

***Abstract*— In this paper we propose a new vision based method for robot localization using an omnidirectional camera. The method has three steps efficiently combined to deal with big reference image sets, each step evaluates less images than the previous but is more complex and accurate. Given the current uncalibrated image seen by the robot, the hierarchical algorithm gives the possibility of obtaining appearance-based (topological) and metric localization. Compared to other similar vision-based localization methods, the one proposed here has the advantage that it gets accurate metric localization based on a minimal reference image set, using the 1D three view geometry. Moreover, thanks to the linear wide baseline features used, the method is insensitive to illumination changes and occlusions, while keeping the computational load small. The simplicity of the radial line feature used speeds up the process while it keeps acceptable accuracy. We show experiments with two omnidirectional image data-sets to evaluate performance of the method.**

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

I. INTRODUCTION

Usually the robots have at their disposal a map where they have to act (given or because if has been automatically acquired). For some robotic tasks, such as navigation or obstacle avoidance, the robot needs to localize itself to correct the trajectory errors due to, for instance, 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 the robot needs just topological localization (e.g. identifying in which room we are, because the robot is asked for some information about a certain room) and indeed this is a more intuitive information to communicate with humans.

When working with computer vision, these maps usually consist of a set of more or less organized reference images. In this paper we present a vision based method for hierarchical robot localization, combining topological and metric information, using a Visual Memory (VM). This VM 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 of 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. One of the main novelties of our proposal is that it gets till metric localization using multiple view geometry constraints. Most of the previous vision-based localization algorithms give as final localization the location of one of the images from the reference set, that implies less accuracy or very high density in the stored reference set. For example in [4], a hierarchical localization with omnidirectional images is performed. It 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. With the metric localization we propose, the stored images density is only restricted by the wide-baseline matching that the local features used are able to deal with.

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), using 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 [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 radial distortion is another recent application of the 1D tensor for omnidirectional images [9].

This work was supported by project MCYT/FEDER - DPI2003 07986.

The line features are detailed in section II and all the steps of the localization method are explained in section III. 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 (e.g. robustness against partial occlusion, easy and fast extraction or 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 always their straightness, being 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. As it can be seen in 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.

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 ($\theta \in [0, 2\pi]$).

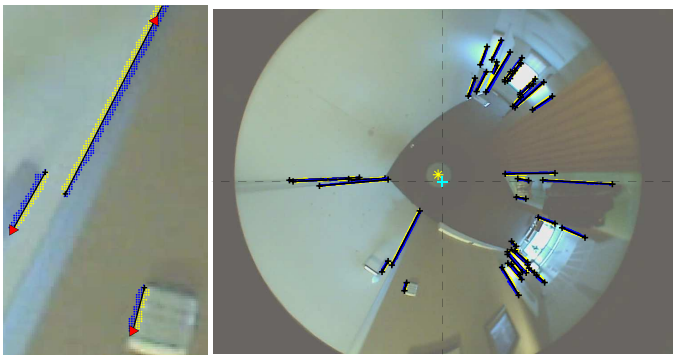


Fig. 1. Left: detail of some LSRs, divided by their line (the darkest side on the right in blue and the lighter on the left in yellow). A red triangle shows the lines direction. Right: all the radial lines with their LSR extracted in one image. The estimated center of projection is pointed with a yellow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

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

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

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. After studying the work where they are defined, and trying to keep a compromise between complexity and discriminative power, we chose as most suitable for us those invariants defined for scale photometric transformations using relations between couples of color bands. The definitions of the 3 descriptors chosen are as follows:

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

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. As known, using Principal Component Analysis (PCA) we get several linear combinations of the descriptors we study, each of them with a certain discriminative level indicated. If we have a look at the weights of each descriptor in the linear combinations which turn to be more distinctive for our feature sets, we can get an idea of which descriptors seem more representative. Then we reduced our line descriptors vector to 12 elements, what 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. Even the matches between adjacent images can be also pre-computed. The three steps of our method, schematized in Fig. 2, are as follows:

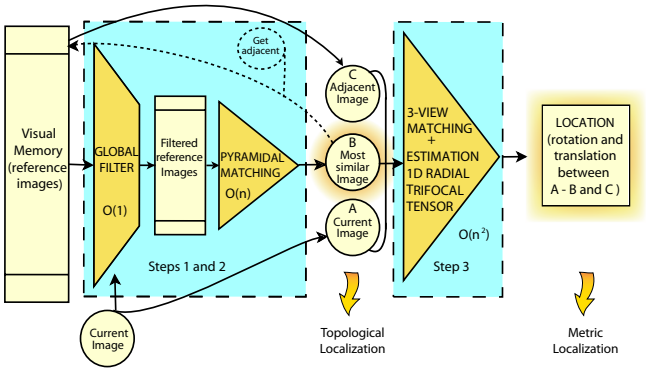


Fig. 2. Diagram of the three steps of the hierarchical localization.

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 an established 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

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

with N_i the number of matches (LSRs that fall in the same bin of the histograms, see Fig. 3) between images c and v in level i of the pyramid. w_i is 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

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

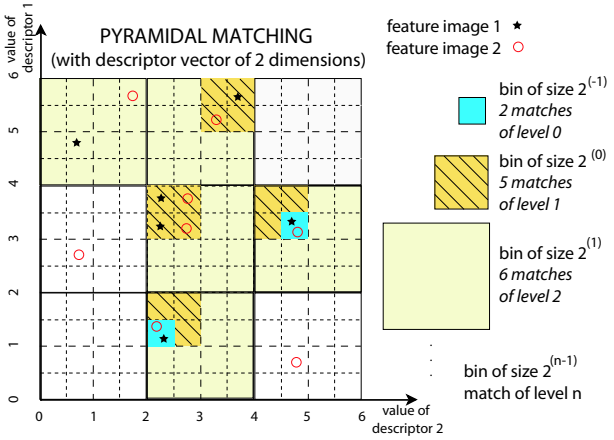


Fig. 3. Example of Pyramidal Matching, with correspondences in level 0, 1 and 2. For graphic simplification, with a descriptor of 2 dimensions.

Notice that the matches found here are not always individual feature-feature matches, as we just count how many fall in the same bin. The more levels we check in the pyramid, the bigger are the bins, so the easier it is to get multiple coincidences in the same bin.

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. From the previous step, the pyramidal matching we could think to get some matches. However, as we show from several tests, it is not so convenient because many times the matches are multiple and it is not possible to distinguish which one goes with which one, due to the discrete size of the bins in the pyramid.

1) Line matching algorithm.: Once we have the most similar image, 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.

They must have the same direction (every line, has a direction assigned when extracted, depending in which side of the line is the darkest region).

The relative rotation between each line and one compatible in the other view has to be consistent. It must be similar to the global rotation between both images obtained with the average gradient of all image pixels. This global rotation is not accurate at all, but if it is for instance 90 degrees, the real rotation between matches should not be zero.

As explained, the rest of descriptors are computed in regions around the line in both sides. To classify two lines as compatible, at least the descriptor distances in one side of the LSR should be under an established threshold.

• Compute a unique distance, using all the descriptors individual distances, 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

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

where d^{Min}_{RGB} and d^{Min}_{DCT} 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], that is based on the probability that two lines will keep their relative position in both views. It improves the robustness of this initial matching (although it can reject some good matches, those can be recovered afterwards in next step, now it is more important to have robustness).

• 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. 4 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 by 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,

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

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 (internal parameters of the camera are known), we have enough to compute a 1D trifocal tensor [8]. In our case with omnidirectional images we have a radial 1D trifocal tensor. From this tensor, even without camera calibration, 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 data-sets, *Almere* and our own (named *data-set 2*). From this second one ground truth data was available, convenient to measure the errors in the localization.

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

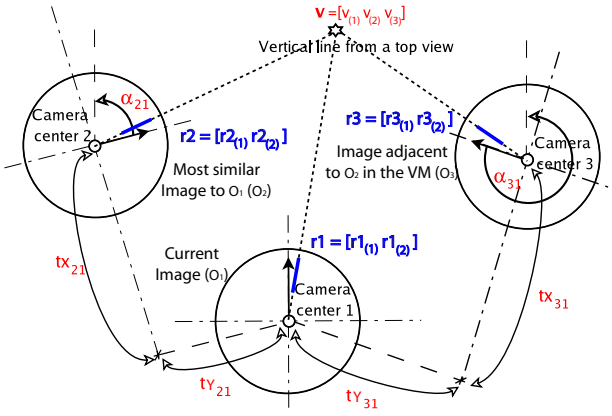


Fig. 4. Motion between images and landmark parameters

(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. 5 we can see a 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.

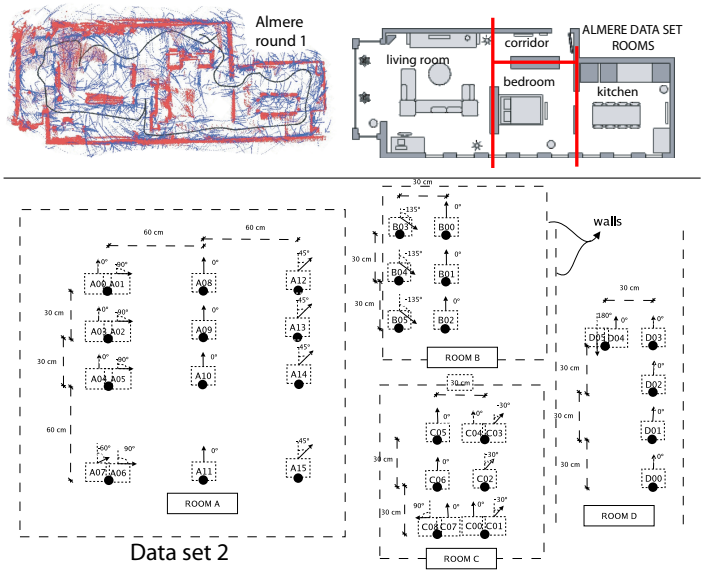


Fig. 5. Grids of images in rooms used in the experiments. Top: *Almere* data-set. Bottom: *Data-set 2*

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 the three steps.

- The first step (Section III-A) has constant complexity with the number of features in the image. Here all the images in the VM are evaluated.
- The second step algorithm (Section III-B) is linear with the number of features. Here only the images that passed the previous step are evaluated, using 12-descriptors/feature.
- The third step process (Section III-C) has quadratic complexity with the number of features, and a 22-descriptors vector per feature is used. However here we evaluate only the query image and two from the VM.

Indeed, in each step we have a computational cost linear with the number of images remaining on it. Then, it is very important to keep the minimal set of images for the last 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 images, corresponding to very different ones than the query one, leaving more or less the ones of the same room and a small percentage of *outliers* (images from other rooms). Taking into account amount of images rejected and false negatives obtained, the best performing value for this filter was rejecting the images with more than 80% difference in the global descriptors. With the *Data-set 2* around 70% of the images were rejected, from where only an average of 4% of the images were rejected being correct (false negatives). Using the *Almere* images the results were a little worse (55% rejected and 9% false negatives). This decreasing in the performance become more accentuated when classifying images from the round 4, with many occlusions, where sometimes we got too many false negatives (around 60% rejected with 14% false negatives). In the worst cases all good solutions were rejected already in this first step, making to find a correct classification impossible for the next steps.

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 to have so much accuracy. In the *data-set 2* the images are already more separated, as they were taken every certain distance and not during a whole robot trajectory, 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 observe 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 (Almere1$\propto$1 indicates we are classifying images from round 1 against the VM of other images from round 1, while Almere4$\propto$1 means that we are classifying images from round 4 against the VM made from round 1). The image chosen as most similar (1 Ok in the table) was correct in 81% of the cases in *Almere*1$\propto$1 test and 97% in the *Data-set 2*. In the test *Almere*4$\propto$1 the performance was lower., but we should take into account that this round corresponds to a round with many occlusions, while the reference set (*Almere*1) is from an occlusion-clean round. This makes the first step work worse and the global

Almere1$\propto$1		Almere4$\propto$1		*Data-set 2*	
1 Ok	3 Ok	**1 Ok**	3 Ok	**1 Ok**	3 Ok
81%	52%	71%	20%	97%	74 %

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.

Location	Rotation		Transl. direction	
Param.	α_{21}	α_{31}	t_{21}	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).

appearance based descriptors less useful. In some cases, even all possible correct images were rejected by it. This confirms the importance of using local features to deal with occlusions. If we skip the first step that is based on global appearance, the performance increases but also the computations will grow quite a lot. We must find a compromise between complexity and correctness.

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

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

We have compared this approach 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 *Data-set 2* the percentage of good choices was the same with both, but the percentage when checking the 3 first images chosen was worse with individual matching. However with the *Almere* data, the individual matchings similarity measure performed a little better. As the performance seems similar but the pyramidal method has the advantage of having linear complexity with the number of features (in the experiments around 25% less execution time), we decided to use it for this step.

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.6 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.

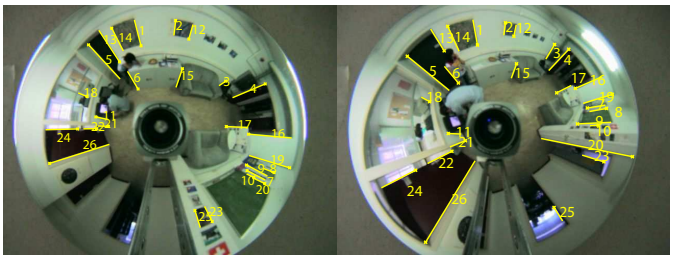


Fig. 6. Matches with *more separated images* than the couples obtained in the Pyramidal matching - Room A (A01-A07)- 26 matches/ 7 wrong.

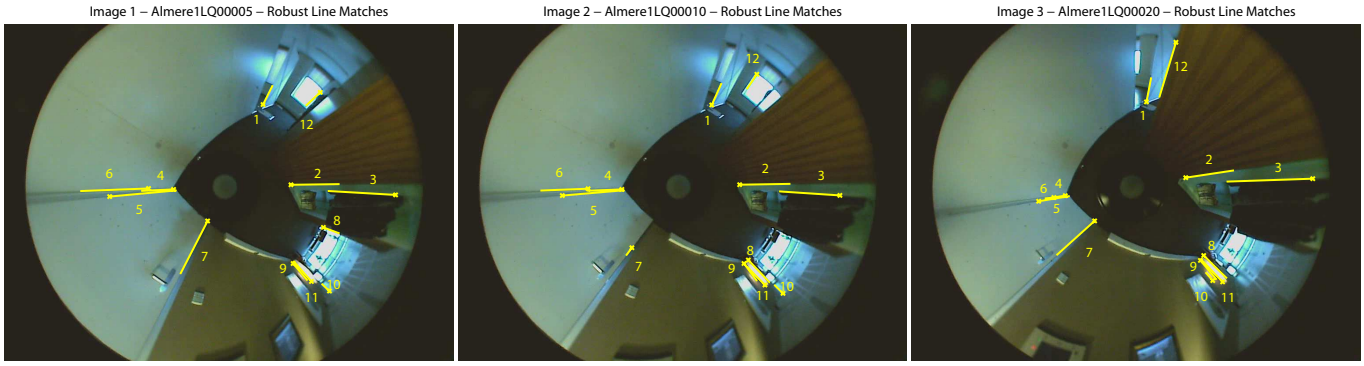
C. Metric Localization Results

As explained before, we need three views to get the localization. Therefore, once we have two-view matches, between the 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 robustly estimate the 1D radial tensor, so we get also a robust set of three-view matches (those who fit the geometry of the tensor). In Fig. 7, 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.

V. CONCLUSIONS

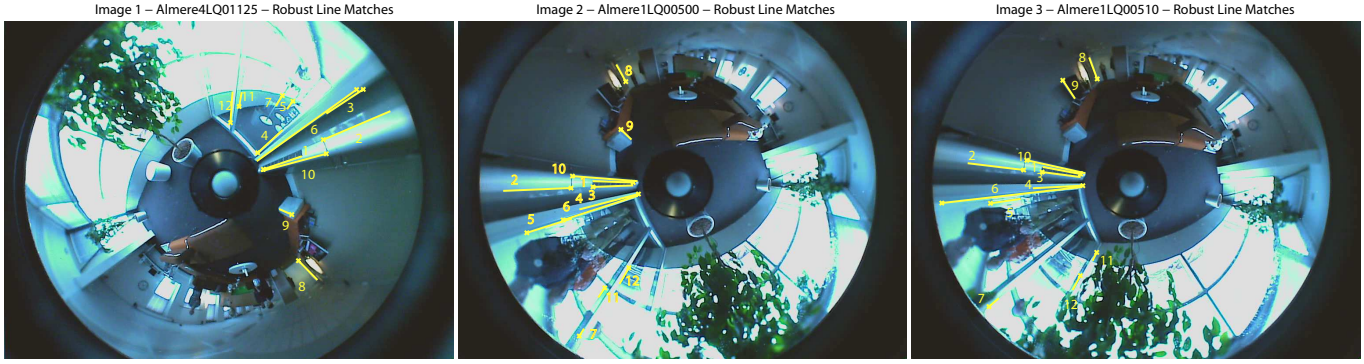
In this work, we have proposed a hierarchical mobile robot localization method. Our three-step approach uses omnidirectional images and combines topological (which room) and metric localization information. The localization task computes the position of the robot relative to a grid of training images 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

QUERY MOST SIMILAR FOUND ADJACENT IN THE VM



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

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



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

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

Fig. 7. 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.

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. This allows accurate localization with a minimal reference dataset, contrary to the approaches that give as current localization the location of one of the reference views (as topological information is correct, but as metric can give quite high error). Our experiments, with two different data-sets of omnidirectional images, show the efficiency and good accuracy of this approach.

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, T. Maeda, and H. Ishiguro. Image-based memory for robot navigation using properties of the omnidirectional images. In *Robotics and Autonomous Systems, Elsevier, Vol. 47, Iss. 4 , pp. 251-267*, 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 keypoints. *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] J.J. Guerrero and C. Sagüés. Camera motion from brightness on lines. combination of features and normal flow. *Pattern Recognition*, 32(2):203–216, 1999.

[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.