Indoor autonomous navigation using visual memory and pattern tracking

O. Ait Aider, T. Chateau and J. T. Lapresté
aitaider,chateau,lapreste@lasmea.univ-bpclermont.fr

Abstract

The paper deals with autonomous environment mapping, localisation and navigation using exclusively monocular vision and multiple 2D pattern tracking. The environment map is a mosaic of 2D patterns detected on the ceiling plane and used as natural landmarks. The robot is able to reproduce learned trajectories defined by key images representing the visual memory. The pattern tracker is based on particle filetring. It uses both image contours and gray scale level variations to track efficiently 2D patterns on cluttered background. An original observation model used for filter state updating is presented.

1 Introduction

For an indoor mobile robot system, the ability to autonomously map its environment and localize itself relatively to this map is a highly desired property. Using natural rather than artificial landmarks is another important requirement. Several works using monocular vision for mapping and self localization exist. The most important difficulty is to achieve the generation of a sufficient number of landmarks which can be robustly recognized during navigation session with a near real time rate. Interest points [13], straight lines [14] and rectangular patterns [5] where used. The first approaches focused on producing efficient algorithms to match a set of observed patterns with a subset of the map primitives [14]. Generally, fusion of multiple sensor data (odometry) was used to achieve real time computing and ambiguity eliminating. More recently, the success of real-time tracking algorithms simplified the matching process and allowed to use structure from motion technics to compute the 3D coordinates of the observed features [13].
Among the large variety of existing tracking methods, model-based approaches provide robust results. These methods can use 3D models or 2D templates such as appearance models [7, 3] or Geometric primitives as contour curves and CAD description [11, 10, 12]. For a realistic robotics application there is a need for algorithms enabling not only pattern tracking but also automatic generation and recognition. Object recognition algorithms based on segmented contours (straight lines, ellipses, corners,...) have reached today a high level of robustness and efficiency. Thus, it seems judicious to investigates trackers which use contour based pattern models. Note that tracking in cluttered background and with partial occlusions is challenging because representing patterns with only contour information may produce ambigeous measures. Extensive studies of either active or rigid contour tracking have been presented in litterature [6, 15, 4, 2, 8]. The used methods are usually defined in the Bayesian filtering framework assuming that the evolution of the contour curve state follows a Markov process (evolution model) and that a noisy

observation (measurement model) is available. The contour state is tracked using a probabilistic recursive prediction and update strategy [1]. More recently, Particle filtering was introduced in contour tracking to handle non Gaussianity of noise and non linearity of evolution models [6].
In this paper, we describe a method for autonomous environment mapping, localisation and navigation for an indoor mobile robot using monocular vision and multiple 2D pattern tracking. The environment map is a mosaic of 2D patterns detected on the ceiling plane and used as natural landmarks. The presented algorithm enables the mobile robot to reproduce learned trajectories defined by key images representing the visual memory. The pattern tracker is a contour model-based one and takes into account the image gray scale level variations. It uses the condensation algorithm [6] to track efficiently 2D patterns on cluttered background. An original observation model is used to update the particle filter state. The paper is organized as follows: in section 2 the autonomous localisation and navigation using key images is presented. The pattern tracker is then defined in section 3. An experimental evaluation is finally presented in section 4.

2 Navigating using visual memory

2.1 Problem Formalization

The mobile robot system is composed of three frames as shown in Figure 1: the world frame $\mathscr{F}_W$, the robot frame $\mathscr{F}_R$ and the camera frame $\mathscr{F}_C$. To localize the robot in its environnement one have to estimate the transformation ${}^W\mathbf{T}_R$ between the world frame and the robot frame. Assuming that the transformation ${}^C\mathbf{T}_R$ between the camera frame and the robot frame is known thanks to a camera-robot calibrating method, the image is corrected so that it corresponds to what would be observed if the camera frame fits the robot frame i.e. ${}^C\mathbf{T}_R = \mathbf{I}$. As the visual landmarks are on the ceilling plane, the correction to apply on the observed data is a homography ${}^C\mathbf{H}_R$. Knowing ${}^C\mathbf{T}_R$, ${}^C\mathbf{H}_R$ can be expressed as follows:

$$ {}^C\mathbf{H}_R = \mathbf{K}\left(\mathbf{R} - \mathbf{t}\mathbf{n}^T/d\right)\mathbf{K}^{-1} \tag{1} $$

Where $\mathbf{K}$ is the camera intrinsic parameters matrix, $\mathbf{R}$ and $\mathbf{t}$ are respectively the rotation matrix and the translation vector in ${}^C\mathbf{T}_R$, $\mathbf{n}$ is the vector normal to the ceilling plane and d is the distance from the robot frame to this plane. After applying the computed correction to image data, one can assume for the clarity of the presentation and without loss of generality that ${}^C\mathbf{T}_R$ is set to the identity matrix.

2.2 On-line robot pose computing

Let us denote $\mathscr{F}_M$ the 2D frame defined by the x and y-axis of $\mathscr{F}_W$ and related to the ceiling plane. We assume that a set of 2D landmarks $\mathbf{m}_i$ detected on this plane are modeled and grouped in a mosaic represented by a set $\mathscr{M} = \left\{\left(\mathbf{m}_i, {}^M\mathbf{T}_i\right), i = 1, ..., n\right\}$ where the planar transformation matrix ${}^M\mathbf{T}_i$ between $\mathscr{F}_M$ and a frame $\mathscr{F}_i$ related to $\mathbf{m}_i$ defines the pose of $\mathbf{m}_i$ in the mosaic (Figure 2). We have ${}^M\mathbf{T}_i = \begin{bmatrix} {}^M\mathbf{R}_i & {}^M\mathbf{P}_i \\ \mathbf{0} & 1 \end{bmatrix}$ where

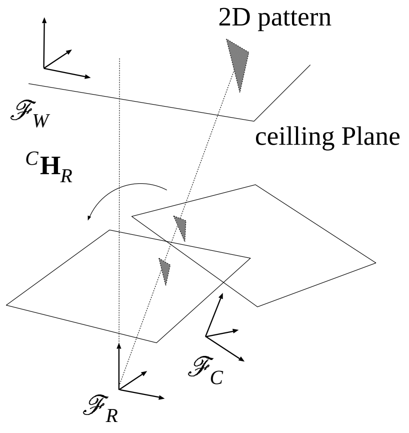


Figure 1: The camera-robot system
.

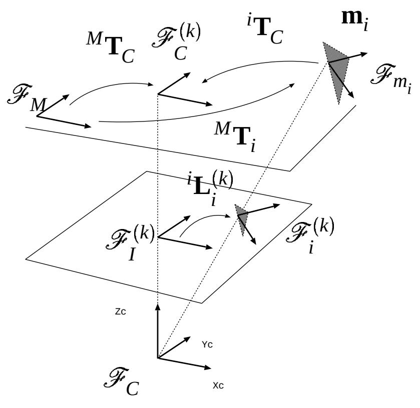


Figure 2: Robot pose computing using visual data of a 2D model
.

${}^M\mathbf{R}_i = \begin{bmatrix} C\theta_i & -S\theta_i \\ S\theta_i & C\theta_i \end{bmatrix}$ expresses the rotation of the model and ${}^M\mathbf{P}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ its position. Localizing the robot at an instant k consists in computing ${}^M\mathbf{T}_C^{(k)}$ which defines the homogenous transformation between the projection $\mathscr{F}_C^{(k)}$ of the camera frame $\mathscr{F}_C$ on the mosaic plane as shown in Figure 2. At the instant k, the robot grabs an image I_k of the ceiling. Let $\mathscr{F}_I^{(k)}$ be a 2D frame lied to the image plane. The pose of the projection of an observed model $\mathbf{m}_i$ on the image plane is defined by the transformation ${}^I\mathbf{L}_i^{(k)} = \begin{bmatrix} {}^\mathbf{I}\mathbf{r}_\mathbf{i}^{(\mathbf{k})} & {}^\mathbf{I}\mathbf{p}_\mathbf{i}^{(\mathbf{k})} \\ \mathbf{0} & 1 \end{bmatrix}$ between $F_I^{(k)}$ and a frame $F_i^{(k)}$ lied to the projection of $\mathbf{m}_i$ where ${}^M\mathbf{r}_i = \begin{bmatrix} C\theta_i^{(k)} & -S\theta_i^{(k)} \\ S\theta_i^{(k)} & C\theta_i^{(k)} \end{bmatrix}$ express the rotation of the model and ${}^M\mathbf{p}_i = \begin{bmatrix} u_i^{(k)} \\ v_i^{(k)} \end{bmatrix}$ its position in the image. Considering the inverse of the perspective projection, we obtain the transformation between the projections $F_C^{(k)}$ and F_i of the camera frame and the model frame respectively on the mosaic plane (Figure 1):

$$ {}^C\mathbf{T}_i^{(k)} = \begin{bmatrix} {}^C\mathbf{R}_i & {}^C\mathbf{P}_i \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} {}^I\mathbf{r}_i^{(k)} & \frac{Z}{f}{}^I\mathbf{p}_i^{(k)} \\ \mathbf{0} & 1 \end{bmatrix} $$

where Z is the distance from the origin of the camera frame to the ceiling and f the focal length. We can thus express the robot pose relatively to $\mathscr{M}$ with respect to the parameters of a seen model $\mathbf{m}_i$ by

$$ {}^M\mathbf{T}_C^{(k)} = {}^M\mathbf{T}_i \left({}^C\mathbf{T}_i^{(k)}\right)^{-1} \qquad (2) $$

Note that the robot position is computed up to a scale factor $\frac{Z}{f}$. The absolute position can be retrieved if the camera is calibrated. Otherwise, the computed pose is sufficient to achieve navigation using visual servoing as we will see in section 2.

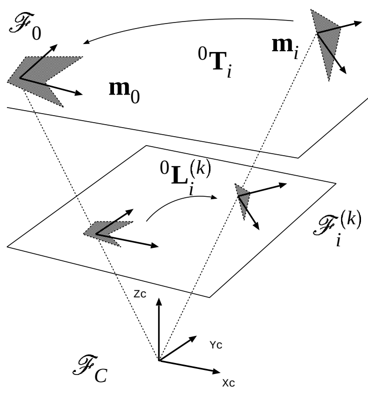


Figure 3: Computing the pose of a new model in the mosaic

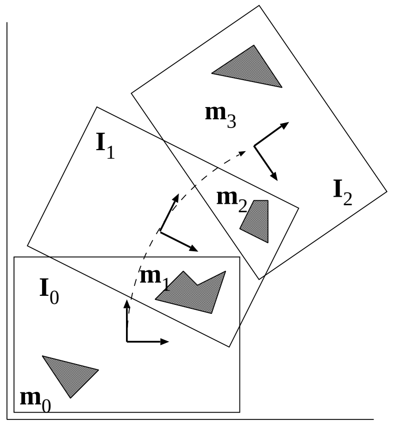


Figure 4: An example of key images forming a trajectory

2.3 Environment mapping

We will now explain the process of building the mosaic of 2D landmarks. At time $k = 0$, the robot grabs an image I_0, generates a first 2D model $\mathbf{m}_0$ and associates to it a frame $\mathscr{F}_0$. In fact, this first model will serve as a reference to the mosaic. Thus, we have $\mathscr{F}_0 = \mathscr{F}_M$ and $^M\mathbf{T}_0^{(k)} = \mathbf{I}$ ($\mathbf{I}$ is the identity matrix). In the image $\mathbf{I}_0$, a tracker τ_0 is initialized with a state $^I\mathbf{L}_i^{(0)}$. As the robot moves, the state of τ_0 evoluates with respect to k. The system generates other models. At each generation of a new model $\mathbf{m}_i$ at the instant k, a new tracker τ_k is initialized with a state $^I\mathbf{L}_i^{(k)}$. Due to the mosaic rigidity, the transformation between the two model frames is time independent and equal to $^0\mathbf{L}_i =^0 \mathbf{L}_I^{(k)I}\mathbf{L}_i^{(k)}$ (Figure3). Noting $^0\mathbf{L}_i = \begin{bmatrix} ^0\mathbf{r}_i & ^0\mathbf{p}_i \\ \mathbf{0} & 1 \end{bmatrix}$ and projecting this transformation onto the mosaic plane we obtain the pose of the new model m_i in $\mathscr{M}$:

$$^M\mathbf{T}_i = \begin{bmatrix} ^0\mathbf{r}_i & \frac{Z}{f}{}^\mathbf{0}\mathbf{p_i} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

Of course, as the robot moves, $\mathbf{m}_0$ may eventually disappear at an instant k when creating a new model $\mathbf{m}_i$. In fact, it is sufficient that at least one model $\mathbf{m}_j$, already defined in the mosaic, is seen at the instant k. To compute the pose $^M\mathbf{T}_i$, we first calculate $^j\mathbf{L}_i$ and project it to obtain $^\mathbf{j}\mathbf{T_i}$. The model $\mathbf{m}_i$ is then added to $\mathscr{M}$ with the pose $^M\mathbf{T}_i =^0 \mathbf{T}_j^j\mathbf{T}_i$.

2.4 Path learning and visual navigation

The idea is to enable the robot to learn and reproduce some pathes relating important places in the environment. Let us consider a trajectory executed during the learning phase and relating a point A to a point B. A set of so called key images is chosen among the sequence of video images acquired during navigation following the trajectory. A key image I_k is defined by a set of mosaic models and their poses in the image (Figure 4): $I_k^{(AB)} = \left\{ \left(\mathbf{m}_i, {}^I L_i^{(k)} \right), i = 1, 2, ... \right\}$. A trajectory relating A and B is then noticed $\Phi_{AB} = \left\{ I_k^{(AB)}, k = 1, 2, ... \right\}$ Key images are chosen so that the combination of the elementary trajectories between each couple of successive key images $I_k^{(AB)}$ and $I_{k+1}^{(AB)}$ forms a global trajectory which approximatively fits the learned trajectory. Some conditions have to be satisfied when creating Φ_{AB}: i) two successive key images must contain at least one common model of the mosaic, ii) the variation of the orientation of a model between two successive key images is smaller than a defined threshold. The first condition is necessary to visual servoing. The second is motivated by the fact that several different pathes relating two poses exist. It is thus necessary to insert additional key images to reduce the variation of orientation between two successive images (Figure 4).Visual servoing is used to carry the robot from a key image to another.

3 Pattern tracking

3.1 The general tracking problem formulation

The tracking problem can be expressed as the estimation of the state of a dynamical system basing on noisy measurements done at discret times. Let us consider that, at time k, the state of a system is defined by a vector $\mathbf{X}_k$ and the measurements by a vector $\mathbf{Z}_k$. Based on a Bayesian approach, the tracking consists in iteratively predicting and updating the pdf of the system state using respectively the dynamical and the observation models. The posterior pdf $p\left(\mathbf{X}_k | \mathbf{Z}_{\mathbf{1:k}}\right)$ is thus estimated as the vector $\mathbf{Z}_{1:k} = (\mathbf{Z}_i, i = 1, ..., k)$ containing the latest measurements becames available online.
Particle filtering is an elegant solution in case of non-linearity of the evolution and the measurement functions and non-Gaussianity of noise. The key idea is to use a Monte Carlo method to represent the pdf of $\mathbf{X}_k$ by a set of samples (particles) $\mathbf{S}_k^{(i)}$. A weight $w_k^{(i)}$ is associated to each particle. It corresponds to the probability of realisation of the particle. The set of particles $\left\{ \mathbf{S}_k^{(i)}, i = 1, ..., N_p \right\}$ (where N_p is the number of particles) is resampled at each iteration according to the weights. The most probable samples are retained. If the number of particles is sufficiently high, the state estimated from the paticles converges toward the real value. This method necessitates drawing samples from the pdf $p\left(\mathbf{S}_k^{(i)} | \mathbf{Z}_{1:k}\right)$. This is not straightforward for any density function. It is then convenient to use a so called proposal density $\pi\left(\mathbf{S}_k^{(i)} | \mathbf{Z}_{1:k}\right)$ similar to $p\left(\mathbf{S}_k^{(i)} | \mathbf{Z}_{1:k}\right)$ and from which samples can be generated.
In this paper we use the Condensation algorithm which is a particle filter version where the proposal density is equal to the prior one $\pi\left(\mathbf{S}_k^{(i)} | \mathbf{Z}_{1:k}\right) = p\left(\mathbf{S}_k^{(i)} | \mathbf{S}_{k-1}^{(i)}\right)$ [6].

3.2 Pattern modelling

The pattern model must enable not only near real time tracking but also automatic generation and recognition. It must be complex enough to discard ambiguities due to the presence of objects in the background similar to parts of the model and simple enough to reduce the computational cost of tracking. The structure of a pattern is built in two levels:

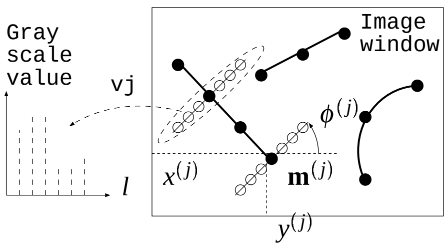
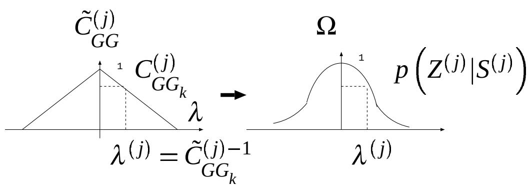


Figure 5: Pattern model: sampled points on segmented contours and corresponding gray scale vectors in the gradient direction



$\tilde{C}^{(j)}_{GG}$
$C^{(j)}_{GG_k}$
λ
$\lambda^{(j)} = \tilde{C}^{(j)-1}_{GG_k}$
Ω
$p\left(Z^{(j)}|S^{(j)}\right)$
$\lambda^{(j)}$

Figure 6: Deriving the probabilty measure from the inter-correlation measure .

the first level is composed of a set of contours polygonally approximated by straight line segments and arcs, the second level is composed of a list of vectors whose elements represent the evolution of the image gray scale value around points sampled along the contours and following normal directions at these points (Figure 5). The contour representation is used to automatic generation and recognition and thus to the tracking initialization. During the tracking, only the gray scale sample vectors are used to estimate the state of the pattern. Let us consider a window of interest in the image with a center (x_c, y_c) and dimensions Δ_x and Δ_y. Each segmented contour is sampled in a set of image points $\left\{\mathbf{m}^{(j)} = \left(u^{(j)}, v^{(j)}\right), j = 1,...,N_m\right\}$ where N_m is the number of points. At each point, we built a vector $\mathbf{V}^{(j)} = \left(g_1^{(j)}, g_2^{(j)}, ..., g_l^{(j)}, ...\right)^T$ composed of N_s gray scale value samples from the image following the gradient direction at the pixel $\mathbf{m}^{(j)}$ and with a fixed step size δ. $g_l^{(j)}$ is a bilinear approximation of the gray scale values of the nearest four pixels. A pattern model can thus be expressed as follows:

$$\mathbf{M} = \left\{\left(\mathbf{U}^{(j)}, \mathbf{V}^{(j)}, \mathbf{W}^{(j)}\right), j = 1,...,N_m\right\} \tag{4}$$

with $\mathbf{U}^{(j)} = \left[x^{(j)}, y^{(j)}, \phi^{(j)}\right]^T$, where $x^{(j)} = \frac{u^{(j)} - x_c}{\Delta_x}$ and $y^{(j)} = \frac{v^{(j)} - y_c}{\Delta_y}$ are the normalized co-ordinates of $\mathbf{m}^{(j)}$ inside the interest window and $\phi^{(j)}$ the gradient orientation at $\mathbf{m}^{(j)}$. The vector $\mathbf{W}^{(j)} = \left(a^{(j)}, b^{(j)}, ...\right)^T$ is composed of a set of parameters defining a function $\tilde{C}^{(j)}_{GG}$ which is an approximation of the one-dimensional discrete normalized auto-correlation function $C^{(j)}_{GG}$ of $G^{(j)}$ where $G^{(j)}(l) = g_l^{(j)}$ for $l = 1,...,N_s$, and $G^{(j)}(l) = 0$ elsewhere. We have $C^{(j)}_{GG}(\lambda) = \left(1/\parallel \mathbf{V}^{(j)} \parallel^2\right) \sum_{l=-N_S}^{N_S} G^{(j)}(l)\, G^{(j)}(l-\lambda)$. The simplest expression of $\tilde{C}^{(j)}_{GG}$ is a straight line equation (Figure6). $\mathbf{W}^{(j)}$ is then one-dimensional and equal to the slope. The interest of saving the auto-correlation function parameters and the way this measure is used in tracking will be discussed in the next subsection.

3.3 Observation model

State vector definition and dynamical model The state vector $\mathbf{X}_k$ defines the pose of the pattern at time k in the image I_k:

$$\mathbf{X}_k = \left(x_k, y_k, \theta_k, s_k\right)^T \tag{5}$$

x_k, y_k, θ_k are respectively the position of the pattern center and its orientation in the image frame and s_k is the scale factor. The evolution of $\mathbf{X}_k$ is modelled by a noise vector $\mathbf{v}$ from a gaussian distribution of zero mean value and standard deviation $\sigma_{\mathbf{v}} = \left(\sigma_x, \sigma_y, \sigma_\theta, \sigma_s\right)^T$. Thus, the dynamical model function is simply $\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{v}$.

Observation model Let us consider a set $\left\{\mathbf{S}_k^{(i)} = \left(x_k^{(i)}, y_k^{(i)}, \theta_k^{(i)}, s_k^{(i)}\right)^T, i = 1, ..., N_p\right\}$ of predicted state vector particles. A key point in the particle filter design is to define how to compute an observation measurement $\mathbf{Z}_k$ and to estimate $p\left(\mathbf{Z}_k|\mathbf{S}_k^{(i)}\right)$. For each particle, the model is fitted to the image I_k according to $\mathbf{S}_k^{(i)}$. Around each image point coinciding with a model point $\mathbf{m}_j$, an observed vector $\mathbf{V}_k^{(j)} = \left(g_1^{(j)}, g_2^{(j)}, ..., g_l^{(j)}, ...\right)^T$ of gray scale values is built following a direction which is the transformation of the gradient orientation $\phi^{(j)}$ saved in the model and with a step size equal to $\delta s_k^{(i)}$. For each observed vector we compute the normalized inter-correlation measure $C_{GG_k}^{(j)} = \dfrac{\mathbf{V}^{(j)}.\mathbf{V}_k^{(j)}}{\| \mathbf{V}^{(j)} \| \| \mathbf{V}_k^{(j)} \|}$ between the stored vector and the observed vector components. The question is now how to use the inter-correlation measure to estimate $p\left(\mathbf{Z}_k|\mathbf{S}_k^{(i)}\right)$? We first compute the probability $p\left(\mathbf{Z}_k^{(j)}|\mathbf{S}_k^{(i)}\right)$ that each model point $\mathbf{m}^{(j)}$ is placed according to the state vector particle $\mathbf{S}_k^{(i)}$ on the corresponding point in the observed pattern. The maximum of probability is expected at $\mathbf{m}^{(j)}$. The inter-correlation measure $C_{GG^{(k)}}^{(j)}$ can yeld an estimate of the deviation $\lambda^{(j)} = \tilde{C}_{GG}^{(j)-1}\left(C_{GG_k}^{(j)}\right)$ between the observed and the predicted gray scale vectors, $\tilde{C}_{GG}^{(j)-1}$ being the inverse of the auto-correlation function. Assuming that the probability that the observed gray scale vector fits the predicted one follows a Gaussian distribution with respect to $\lambda^{(j)}$ (Figure 6) we can reasonably approximate $p\left(\mathbf{Z}_k^{(j)}|\mathbf{S}_k^{(i)}\right)$ by $\Omega_\sigma(\lambda)$ the one dimensional Gaussian function with standard deviation σ. Assuming that the probabilities $p\left(\mathbf{Z}_k^{(j)}|\mathbf{S}_k^{(i)}\right)$ are mutually independent, it results that

$$p\left(\mathbf{Z}_k|\mathbf{S}_k^{(i)}\right) = \prod_{j=1}^{N_p} p\left(\mathbf{Z}_k^{(j)}|\mathbf{S}_k^{(i)}\right) \tag{6}$$

The characterization of the auto-correlation function by the parameter vector $\mathbf{V}_j$ for each model point improve the precision of the estimate of $p\left(\mathbf{Z}_k|\mathbf{S}_k^{(i)}\right)$. Indeed, The inter-correlation may decrease faster for a highly textured contour point than for a point on a contour defined by two large and homogeneous regions (Figure 7). In addition, this allows to define a criterium to select or reject some contour points during the pattern model building phase. Moreover, note that this measure is robust to illumination changes thanks to normalization.

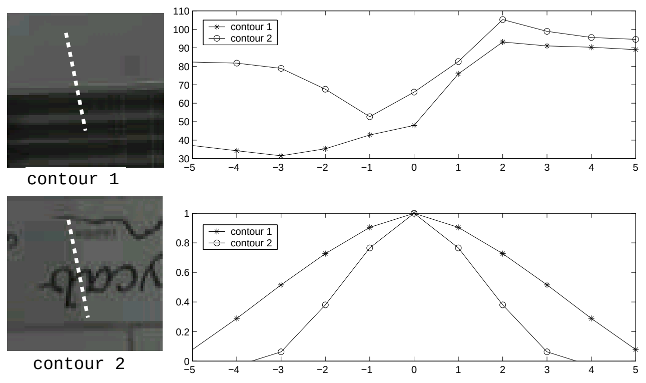


Figure 7: Gray scale sample vectors obtained with two different contours

Handling occlusion and scale changing effects When a gray scale value vector $\mathbf{V}_k^{(j)}$ is observed with a deviation $\lambda^{(j)}$ from the expected pose, the inter-correlation measure decreases slowly from its maximum value $C_{GG_k} = 1$. Contrarily, when some of the vectors $\mathbf{V}_k^{(j)}$ are occluded, a brutal variation of the inter-correlation measure appears. To handle this effect, the algorithm tolerates a fraction of vectors with bad inter-correlation measure ($\lambda^{(j)}$ greater than a defined threshold). These vectors are not included in the probability calculation. Changing scale may cause apparition or disappearing of some details in the image. Consequently, the gray scale value vectors $\mathbf{V}_k^{(j)}$ may be completly different from the original one even without occlusion. It was shown in [9] that the Gaussian function is an efficient scale space Kernel. Before computing the observed vector $\mathbf{V}_k^{(j)}$ according to a prediction $\mathbf{S}_k^{(i)}$, the image region containing $\mathbf{m}^{(j)}$ is convolved with a 2D Gaussian function $\Omega_{s^{(i)}s^{(i)}}$ where $s^{(i)}$ is the predicted scale, if $s^{(i)} > 1$.

4 Experimental evaluation

4.1 Pattern tracker evaluation

The evaluation of the tracker was done by superimposing an image patch (pattern) with known state vector trajectory on a cluttered background image sequence. The goal was to analyse the accuracy of the tracker with respect to the pattern displacement magnitude. The number of particles was set to 200 and the standard deviations of the evolution model parameters were $\sigma_x = 5$ pixels, $\sigma_y = 5$ pixels, $\sigma_\phi = 3$ degrees, $\sigma_s = 0.1$. Figure 8 shows displacements and scale variations estimated by the tracker with respect to the ground truth values. It can be noticed that the tracker remains accurate inside $[-3\sigma, +3\sigma]$.

4.2 Environment mapping and localization

The presented approach was tested on a Pekee robot mounted with a low cost camera. Figure 9 shows an example of mosaic construction. The set of images represents the detected 2D models. The last image is a representation of the constructed mosaic and the robot locations (gray triangles) computed using model tracker.

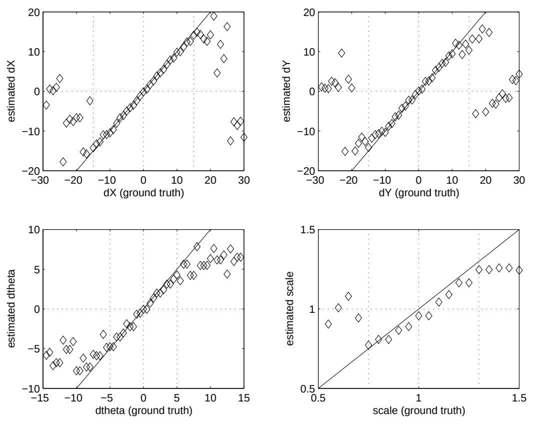


Figure 8: State vector variation estimates (dX and dY in pixels and dtheta in degrees) with respect to real variation values (The straight line represents the ideal curve)

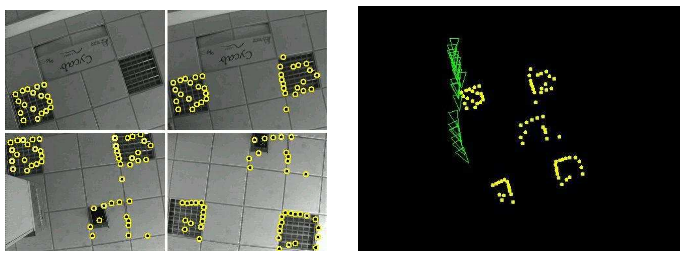


Figure 9: An example of mosaic: a set of key-images with detected 2D patterns (on left), The mosaic and robot locations (on the right)

5 Conclusion and future works

A efficient particle filter tracker with specific observation pobability densities was designed to track planar modelled patterns on cluttered background. Scale changing and partial occlusions were taken into account. The obtained algorithm runs in near real time rate and is accurate and robust in realistic conditions according to experimental validation. The tracker was used for autonomous indoor environment mapping and image-based navigation. Experimental results confirm the validity of the approach. Future works will deal with the adaptation of the tracker to 3D movement tracking using homographies in order to enable autonomous mapping of 3D indoor environment using planar patterns detected on the walls.

Aknowledgement: This work was supported by a grand of the WACIF project by the french government.

References

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 50:174–188, 2002.

[2] B. Bascle, P. Bouthemy, R. Deriche, and F. Meyer. Tracking complex primitives in an image sequence. In *In Proc. of the 12th int. Conf. on Pattern Recognition*, pages 426–431, 1994.

[3] M. Black and A. Jepson. Eigentracking: robust matching and tracking of articulated objects using a view-based representation. In *In Proc. European Conf. on Computer Vision*, pages 329–42, 1996.

[4] A. Blake, R. Curwen, and A. A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *Int. J. Computer Vision*, 11:127–145, 1993.

[5] J.B. Hayet. *Contribution à la navigation d'un robot mobile sur amers visuels texturés dans un environnement structuré*. PhD thesis, Universit Paul Sabatier, Toulouse, janvier 2003.

[6] M. Isard and A. Blake. Condensation-conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29:5–28, 1998.

[7] F. Jurie and M. Dhome. Real-time template matching: an efficient approach. In *In the 12th Int. Conf. on Computer Vision*, Vancouver, 2001.

[8] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contours. *Int. J. Computer Vision*, 1:321–331, 1988.

[9] T. Lindeberg. Scale-space theory: a basic tool for analysing structures at different scales. *Journal of Applyed Statistics*, 21:224–270, 1994.

[10] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *Int. J. Computer Vision*, 8:113–122, 1992.

[11] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2d-3d model-based approach. In *In Proc. IEEE Int. Conf. on Computer Vision, ICCV'99*, pages 262–268, Kerkira, Greece, 1999.

[12] A. E. C. Pece and A. D. Worrall. Tracking with the em contour algorithm. In *In Proc. of the European Conf. on Computer Vision*, pages 3–17, Copenhagen, Danemark, 2002.

[13] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale invariant features. In *Int. Conference on Robotics and Automation (ICRA'01)*, 2001.

[14] R. Talluri and J.K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Trans. on Robotics and Automation*, 12(1):63–77, 1996.

[15] Y. Zhong, A. K. Jain, and M. P. Dubuisson. Object tracking using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:544–549, 2000.