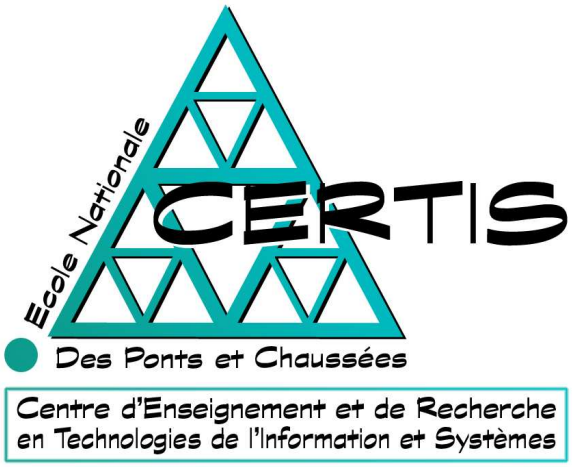
Radon/Hough space for pose estimation

Patrick Etyngier
Nikos Paragios
Jean-Yves Audibert
Renaud Keriven

Research Report 06-22
November 2005



CERTIS, ENPC,
77455 Marne la Vallee, France,

Radon/Hough space for pose estimation

Estimation de la position/orientation d'une caméra l'aide l'espace de Radon/Hough

Patrick Etyngier [2]
Nikos Paragios[2]
Jean-Yves Audibert[2]
Renaud Keriven [2]

[2]CERTIS, ENPC, 77455 Marne la Vallee, France, http://www.enpc.fr/certis/

Abstract

In this document, we present methods to camera pose estimation from one single images in a known environment. The framework of such methods comprises two stages, a learning step and an inference stage where given a new image we recover the exact camera position. This research work focus on achieving such a task with the help of lines and the Radon/Hough transform. The question to be answered in this study is *what can be learnt from lines in order to compute a camera pose estimation*.

Firstly, we tried to point up a relationship between the Hough parameters of a set of lines (ρ, θ) and the camera pose in $SE(3)$ -the space of rigid transformations- based on KCCA method. Such a relationship could be used to predict pose estimation from line configurations.

In a second approach, lines that are recovered in the radon space consist of our feature space. Such features are associated with [AdaBoost] learners that capture the wide image feature spectrum of a given 3D line. Such a framework is used through inference for pose estimation. Given a new image, we extract features which are consistent with the ones learnt, and we associate such features with a number of lines in the 3D plane that are pruned through the use of geometric constraints. Once correspondence between lines has been established, pose estimation is done in a straightforward fashion. Encouraging experimental results based on a real case are presented in this document.

Résumé

Les problèmes de calibrations consistent à retrouver la position et l'orientation d'un observateur (appareil photo, caméra, casque de réalité vistuel etc ...). Ils sont omniprésents dans les domaine de la vision par ordinateur et ont été largement explorés ces dernières années. Cependant, les méthodes par apprentissage sont relativement peu présentes dans la littérature. Nous proposons dans ce document des nouvelles approches de calibration par apprentissage de l'environnement.

La méthode se décompose en deux étapes : d'abord une étape d'apprentissage où un environnement (une pièce par exemple) est appris, et ensuite une étape de déduction où la position et orientation de la caméra est retrouvée. Les travaux présentés dans ce document repose sur la detection de droite dans les images à l'aide de la transformée de Hough. La question qui se pose est :*Que peut-on apprendre des droites afin d'estimer la position d'une camera*. Deux approchesont été explorées :

Nous avons tout d'abord essayé de trouver une relation de corrélation (à l'aide d'un noyau, KCCA) entre les paramètres de Hough (ρ, θ) d'un ensemble de droites, et la position de la camera dans $SE(3)$ -l'espace des transformations rigides-. Une telle relation pourrait être utilisée pour prédire la position à partir d'une configuration de droites.

Dans une deuxième approche, les droites sont caractérisées par des patches centrés autour des maxima locaux de l'espace de Radon. Les droites mise en correspondance dans plusieurs images de points de vue différents permettent à des algorithmes d'apprentissage AdaBoost de capturer un large spectres des caractéristiques d'une droite données.
Etant donnée une nouvelle image, on extrait les caractéristiques consistantes avec celles apprises. Le problème est relaxé par l'ajout de contraintes géométriques qui permettent d'élaguer les résultats obtenus. Lorsque les correspondances entre les droites 3d (reconstruites à partir de la séquence d'apprentissage) et les droites de la nouvelles images sont retrouvées, l'estimation de la position de la caméra est calculée directement. Des résultats expérimentaux sont montrés dans ce document.

Contents

1 Introduction

Pose estimation has been extensively studied in the past years. Nevertheless, it is still an open problem particularly in the context of real time vision. Robot navigation, autonomous systems and self-localization are some of the domains in computational vision where pose estimation is important. One can also cite a number of application in augmented and mixed reality where a solution to this problem is critical. In prior literature pose estimation methods are either feature-driven [30] or geometry-driven [2, 13, 27, 7].

The solution proposed aims to combine feature-based methods and geometry-driven approaches. To this end, we consider geometric elements such as lines to be the most appropriate feature space. Such a selection is motivated from a number of reasons. Lines are simple geometric structures that refer to a compact representation of the scene, while at the same time one can determine angles and orientations that relate their relative positions. Parallel to that, in the image projection space appropriate feature spaces (Hough [11, 34], Radon [34]) and methods exist for fast extraction and tracking [9] of such geometric elements with important precision.

The geometry of line configuration [in the Radon space] can be related with the space of rigid transformation through KCCA. The kernel-correlation between both spaces could help us to infer pose estimation from bunches of examples. We achieved some works in this direction but results does not seems to be promising compared to the feature-and-geometry based method.

Hence, the most promising solution is both feature-and-geometry driven. Lines are caracterized by their projection in the Radon space, forming a feature space. In addition, the geometry of 3d-line configuration can be easily recovered through a 3d reconstruction of the scene. The scheme of our method is thus to reconstruct line while their geometry and features are learnt. Once this is done, a simple line detector coupled with the information previously learnt can be implemented in order to infer the pose estimation from a single view. The domain pointed out is of course real-time application suchlike augmented reality based on a head mounted device or robot navigation.

The reminder of the document is oorganized in the following fashion. In section 2, we present basics of line detection based on Hough and Radon transform. A matching and tracking process are also presented in this section 2.3. The correlation between the line configuration of a static environment and the camera pose in part of section 3. In section section 4, we give a second approach to the problem where the feature space is based on the Radon space. Experimental results and discussion are finally presented in the last section.

2 Feature detection, matching & tracking

The detection of primitives in images is a recurrent problem in computer vision, particularly for points and lines. We are going to be only interested in line extraction in the remaining of this document. Feature detection is a key point of the problem. In this section, we present one of the most powerful tools for robust lines detection in images: the *Hough transform*. Nervertheless, the voting space of the Hough transform has some discretization defects that might be unsatisfactory, in particular when neighborhood of local maxima are to be used further. The Radon transform may be used with the edge map in such cases. Obviously, Hough transform and Radon transform are presented in this section.

2.1 The Hough transform

The Hough Transform is a method able to find parametrized shapes in a data set and has been the purpose of a lot of research since the 60'. The idea of this transform is to express a mapping between an image space and a parameter space which constitute a dual space. Obviously, the parameter space depends on the shape of the primitive we work on. In the first forms, the Hough transform [19, 29] was designed only for 2-lines. Hough[19] chose the slope and the intercept as parameters of the line which can be a complication because both parameters are not bounded. The method is very simple:
Let be $I \subseteq \mathbb{R}^2$ the image space, $P \subseteq \mathbb{R}^2$ the parameter space and $l_0^I = \{(x, y), y = -a_0x - b_0\}$ a line in the image space. The superscripts ^{I} and ^{P} are used to specify if we consider a subset of points in the image space or in the parameter space. Now, for any point $p_0 \in I$ we can compute all the lines $l_i^P = \{(a_i, b_i), y_0 = -a_ix_0 - b_i\}$ going through it using the equation. Since this last equation is linear, we clearly see that a point in image space is mapped to a line in parameter space and vice versa. The same reasoning can be done for a point in parameter space mapped to a line in image space and vice versa. Then all the colinear points (which belongs to a same line) are going to be mapped to as many lines that intersect at the same point in the parameter space. In practice, an accumalator array of the size of the parameter space is set up to zero and each point p in the image space votes for the cells correponding the lines going throught p. The line detection is finally achieved by putting a ceiling on the accumulator array.
The previous description is actually a particular case of the principle of duality in projective geometry where the same equation $l^T p = 0$ can be seen alternatively as *the point equation of the line* and *the line equation of the point* [16]. More recently, A. S. Aguado, E. Montiel and M. S. Nixon [1, 6] have formalized and generalized not only to projective geometry the relationship between the principle of duality and the Hough transform.

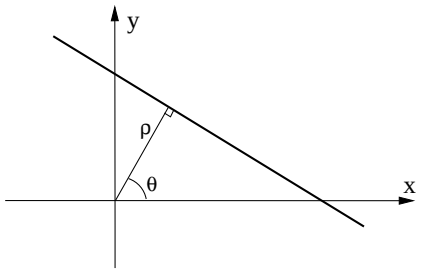


Figure 1: Most used parametrization in the Hough transform

As previously evocated, the line slope parametrization is not always optimal because both parameter are not bounded. The parametrization of line mostly used by the image processing community is the one proposed by Richard O. Duda and Peter E. Hart [12]. The author wrote the line in the following way:

$$l^I = \{(x, y), x\cos(\theta) + y\sin(\theta) - \rho\} \tag{1}$$

where the two parameters θ and ρ are respectively the angle of its normal and the distance to the origin as represented in figure 1. If we choose to restrict $theta$ to $[0, \pi]$, ρ is an algebraic distance otherwise, $\theta \in [0, 2\pi]$ and ρ is an absolute distance. It is clear that this parametrization is unique. In this parametrization, a point in image space does not map anymore to a line but obviously to a sinusoid. Figure 2 shows an example of the Hough transform on a very simple example.

The Hough transform as described so far is from now on written SHT (Standart Hough Transform) and belongs to a classification called *one to many* $(1 \rightarrow m)$. Each point produces indeed a bench of points in the parameter space. The other main classification of the Hough transform is called *many to one* $(m \rightarrow 1)$, but we are going to be back about it in a few lines.
Although the Hough transform is a very robust way to find lines in data set, it is very highly costing from a computational point of view, particularly when the data set of point in the image space is large. In order to improve the computation time, N. Kiryati and Y. Eldar and A. M. Bruckstein [22] have proposed the *Probabilistic Hough Transform* (PHT) that selects a poll of sample in the image space instead of using it entirely. They could thus speed up the process using probabilities by doing a kind of "coarse to fine" Hough transform. The idea has been extended in [26].
As previsouly said, the other main classication of Hough Transform is the *many-to-one* one introduced by the *Randomized Hough Transform* (RHT) [36]. Rather than taking a single point in the image space, Lei Xu and Erkki Oja prefered to

Figure 2: Example of Hough Transform: Image space on the left, parameter space on the right. The three highest values of the parameter space represented by an accumulator give the 3 lines in the image space

compute only one point in the parameter space by taking randomly several points in the image space. For the case of a line, two random points define a line and so, vote for one point in the parameter space. As a threshold is reach in the parameter space, the correponding line is detected and masked out of the image space. The algorithm start again until it does not find any line after a certain number of polls. The PHT and RHT have been unified later by H. Kalviainen, N. Kiryati and S. Alaoutinen [21]. The reader can refer also to [20, 33]for more details.
The Hough transform has been widely extended to other shapes than lines, even in higher dimensions. Nevertheless, we are mainly interested in lines in the remaining of this document.

Nevertheless, the standard Hough transformation space has unfortunately discretization defect as shown in figure 3 in the stripe between the two red lines. Since our goal is to work in such space, we chose instead to use Radon transform which do not suffer of such a defect and can be efficiently implemented thanks to FFT.[[]] In fact, both transformations are derived from the same concept and the output spaces are the same when the Radon space is computed on the edge map. We recall quickly the mathematical writing of the Radon transform.

2.2 The Radon transform

Let g be a mapping defining an image over a domain space U such that:

$$\begin{array}{rccc} g: & U & \longmapsto & \mathbb{R} \\ & u & \longrightarrow & g(u) \end{array}$$

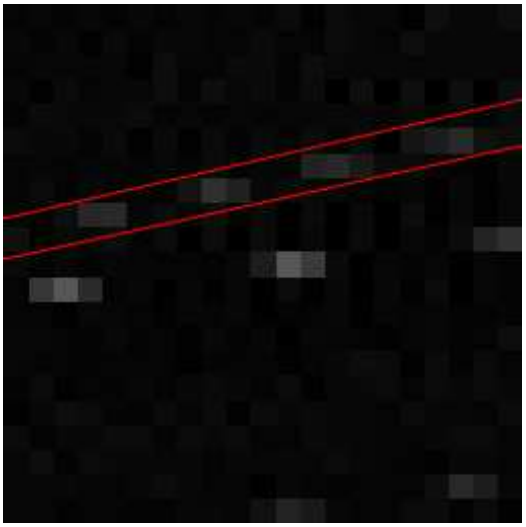


Figure 3: Example of discretization defects using standart Hough transform between the two red lines

and let $f_{\mathbf{p}}(u) = 0$ define a shape described by the vector parameter $\mathbf{p}$. The Radon transform of g regarding to the shape $f_{\mathbf{p}}(u) = 0$ is given by:

$$\mathcal{R}(g)(\mathbf{p}) = \int_U g(x)\delta\left[f_{\mathbf{p}}(x)\right] du \tag{2}$$

where $\delta(.)$ is the Delta-Dirac function. Radon transform in its discrete form is extensively used in tomography image reconstruction but it can also very useful for line detection.
In that particular case, $U = \mathcal{R}^2$ ie $u = (x, y)$ and let $\mathbf{p} = (\rho, \theta)$ such that:

$$f_{\mathbf{p}=(\rho,\theta)}(x, y) = \rho - x\cos(\theta) - y\sin(\theta) \tag{3}$$

and thus, equation 2 can be rewritten:

$$\mathcal{R}(I)(\rho, \theta) = \iint_{\mathbb{R}} I(x, y)\delta\left(\rho - x\cos(\theta) - y\sin(\theta)\right) dxdy \tag{4}$$

where $g = I$ is the image transformed.
Finally, local maxima are thresholded and the median value of neigborood pixel is used to achieved such a thing.

2.3 Tracking / Matching lines in the Radon space

2.3.1 Basic image to image tracking

Local maxima in such a space correspond to lines in the original image and can be extracted in a straightforward fashion. Such a global transformation encodes the entire line structure in a compact fashion, is capable to account for occlusions, local and global changes of the illumination as well as strong presence of noise.

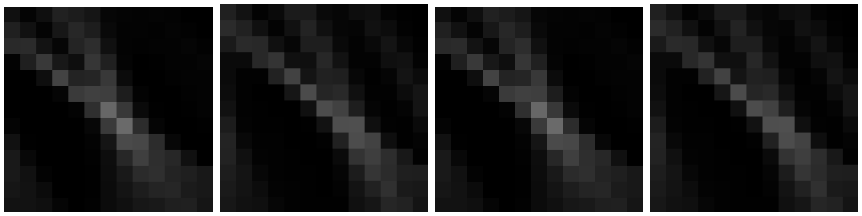


Figure 4: Line signature in the Radon space for a number of consecutive images.

Tracking lines in such a space is a feasible task with simple methods being able to capture the line displacement from one image to the next. Such a problem is simplified due to the constraint that lines corresponds to local maxima in the space and therefore simple comparison between local radon patches could provide explicit correspondences between lines. To this end, we consider simple normalized correlation criterion. We seek to recover the optimal displacement $\mathbf{du} = (dx, dy)$ between two radon images such that the distance between the corresponding patches is minimal. Basically, the algorithm works with the Radon spaces ($\mathcal{R}_1$ & $\mathcal{R}_2$) of two successive images (I_1 & I_2) and for each local maximum detected previously in $\mathcal{R}_1$ - ie a line in I_1 - it searches for the 2d-dimensional shift in $\mathcal{R}_2$ such that an energy is minimized:

$$\min_{\substack{(dx,dy)\in \\ \Omega(X,Y)}} E(dx, dy) \tag{5}$$

where $\Omega(X, Y)$ is the neiborghhood of (X, Y).
The search can be constrained on local maximums of $\mathcal{R}_2$ but experiments did not show an interest of proceeding in such a way. A free shift search is thus prefered in the following. Just as points in images are tracked based on a very slight image to image transformation hypothesis, it is reasonable to make the same assumption in the radon space. Thus, we simply chose to compute a cross normalized sum of

differences in our implementation:

$$E(dx,dy) = \frac{\sum_{u,v} \left[\left(\mathcal{W}_{X,Y}\left\{I_1\right\}(u,v)\right)\left(\mathcal{W}_{X',Y'}\left\{I_2\right\}(u,v)\right)\right]^2}{\sum_{u,v}\left[\mathcal{W}_{X,Y}\left\{I_1\right\}(u,v)\right]^2 \sum_{u,v}\left[\mathcal{W}_{X',Y'}\left\{I_2\right\}(u,v)\right]^2} \tag{6}$$

where $X' = X + dx$, $Y' = Y + dy$, $\mathcal{W}_{X,Y}$ is a designed window centred in (X,Y) such that the values $\mathcal{W}_{X,Y}(u,v)$ are centred (the mean over the windows is substracted).
Obviously, the particular structure of the Radon space which fold up is taken into consideration. We tried other forms of similar energy (correlation ...) but none showed real improvments.

2.3.2 Tracking over a sequence

In the previous line, we presented a simple image to image line tracking. We are however interested in tracking lines over a video sequence. Thus, dying lines - ie lines that are not present anymore in an image- and new line detection should be taken into consideration. Without loss of generality, algorithm 1 outlines the procedure implemented to achieve such a task. It is based on three main functions: image to image line detection, new line detection and outgoing line detection. The former has been described previously. The algorithm tries to keep up to $N_l^{\max}$ during the tracking within the N^{seq} images. New line detection has been already detailled and is used to maintain the number of lines tracked in the current image (up to $N_l^{\max}$). The last function ensure that a line will not be tracked if it not anymore in the current image. In order to decide if a line should be tracked in the following images, the algorithm analyses with the help of variance the patches over N images. Such a way avoids removing and detecting again continuously the same line along the tracking within the video sequence.

2.3.3 Conclusion

We presented an efficient method for line tracking in the Radon space based on correlation patches. Indeed, systems have more and more ressources to make these computations. The correlation patches in the Radon space can also be used in order to improve a manual matching in a sequence of images.

Algorithm 1 Tracking Lines in a video sequence

INPUTS: N, N^{seq},$N_l^{\max}$ be initialized
Initialize $\mathcal{O} = \emptyset$, $\mathcal{N} = \emptyset$, $t \leftarrow 0$
while $t + N <= N^{\text{seq}}$ **do**
  for all *line* $\in \mathcal{O}$ **do**
    Trackline(*line*,$t + N - 2$,$t + N - 1$);
  end for
  $n \leftarrow N_l^{\max} - |\mathcal{O}|$ {number of lines to detect in image}
  $\mathcal{N} \leftarrow$ detect n new lines in image t
  for all *line* $\in \mathcal{N}$ **do**
    Trackline(*line*,$\{t, \ldots, t + N - 1\}$);
  end for
  $\mathcal{O} \leftarrow \mathcal{O} \cap \mathcal{N}$
  for all *line* $\in \mathcal{O}$ **do**
    OutGoing_Detection(*line*,t,$\{t, \ldots, t + N - 1\}$);
  end for
  $t \leftarrow t + 1$
end while

Trackline(*l*,$\{a, \ldots, b\}$) is the basic tracking function of the line *l* between images a and b, in the corresponding Radon/Hough spaces (see 2.3).

OutGoing_Detection(*l*,t,$\{t, \ldots, t + N - 1\}$) is the procedure that detects if the tracking of the line *l* should be stopped or not. (see 2.3).

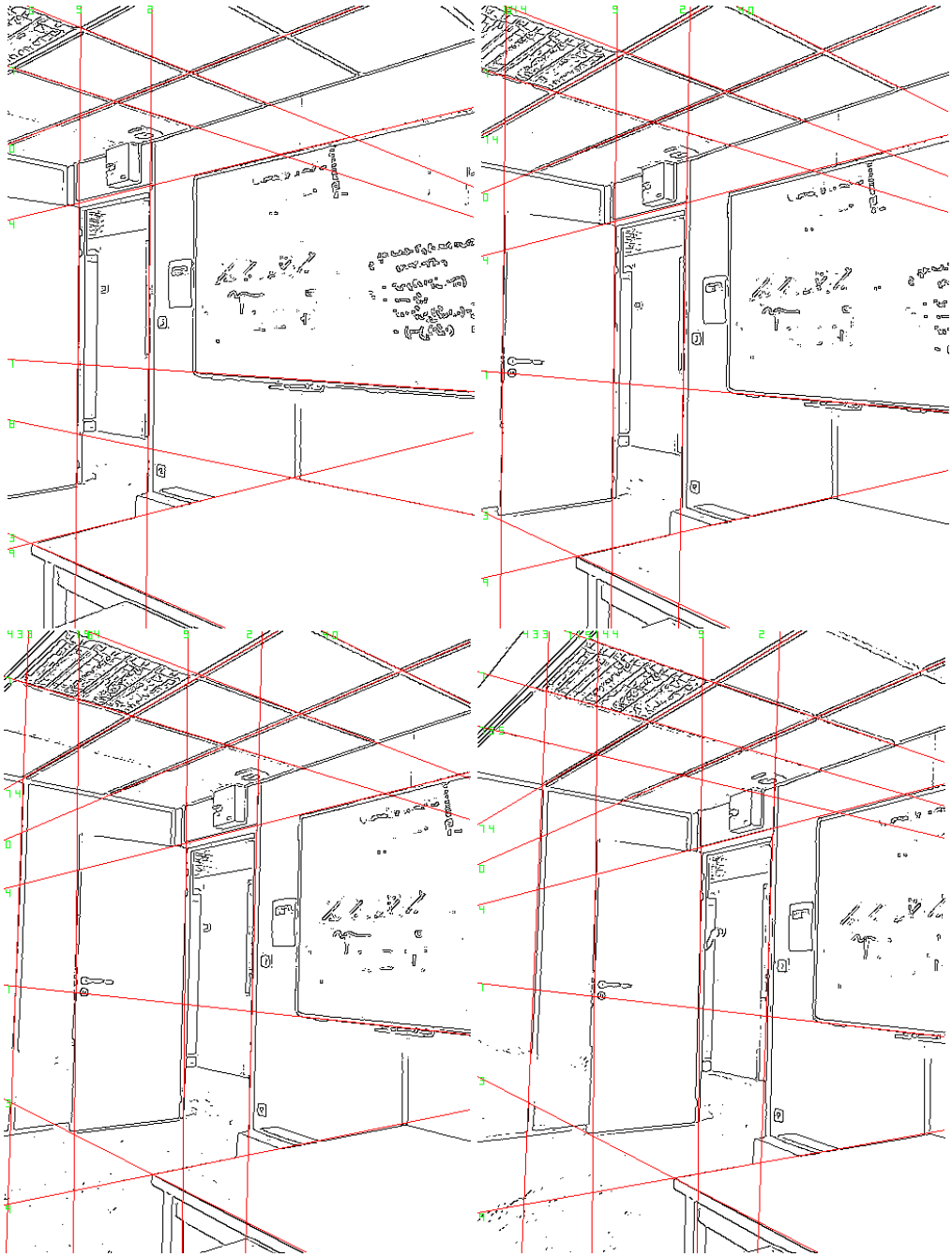


Figure 5: Tracking lines in the radon space and their projections in the corresponding image space. Results are presented in raster-scan format.

3 Correlation between Hough parameter and camera pose

We firstly focused on searching experimentally a *correlation* between the line configuration and camera pose could be expressed. The line configuration refers to the set of all lines, each of them caracterized by the Hough paramters: the distance of the line from the origin, and its normal angle (see section 2.1 and figure 1). A space for such configuration is defined as $\mathscr{H}^N$ in the following section. Similarly, camera pose will evolve in its own space, the space of rigid transformation $SE(3)$. In other words, this research work aims at looking for a correlation between the spaces $\mathscr{H}^N$ and $SE(3)$.

3.1 Notations & Overview

Without loss of generality, we put forward the hypothesis that the scene contains N 3-lines and they are all visible in all images, as well in the learning set as in the testing set. The 3-lines are labeled $1, \ldots, n, \ldots, N$ and we assume that a matching of lines has been performed before. Let us define $\mathcal{H}_n$ the Hough space associated with a 3-line labeled n. $h_n = (\rho_n, \theta_n)$ is an element of $\mathcal{H}_n$ and the feature space is given by:

$$\mathscr{H}^N = (\mathcal{H}_1 \times \ldots \times \mathcal{H}_N)$$

Let us finally define:

$$\begin{array}{rccccc} f: & \mathscr{H}^N & = & (\mathcal{H}_1 \times \ldots \times \mathcal{H}_N) & \longrightarrow & SE(3) \\ & H & = & (h_1, \ldots, h_N) & \longmapsto & M \end{array} \tag{7}$$

where M is the homogeneous matrix associated to the elements of the *Lie* group $SE(3)$.
The aim is to estimate the function $f : \mathscr{H}^N \longrightarrow SE(3)$ which maps an observation $H \in \mathscr{H}^N$ of the N lines in the Hough space to the position $f(H) \in SE(3)$ of the camera. The space $\mathscr{H}^N$ as the space $SE(3)$ are not Hilbert spaces and this is why we used a kernel method. In this document, we will present an experimental approach based on KCCA (Kernel Canonical Correlation Analysis). The idea is to (see sheme of the method in figure 6):

1. map the respective space $\mathscr{H}^N$ and $SE(3)$ into high dimensional Hilbert spaces $\mathscr{W}$ and $\mathscr{Z}$

2. compute couples of canonical vectors $w_i \in \mathscr{W}$ and $z_i \in \mathscr{Z}$ on the which the projection of the *learning point* have maximal correlation.

3. make a prediction from $\mathscr{W}$ and $\mathscr{Z}$ with a *testing point*. The prediction is denoted $\widehat{z} \in \mathscr{Z}$. The reason for predicting in this spaces will become clear later.

4. find a way to retrieve the element in $\widehat{M} \in SE(3)$ corresponding to the prediction $\widehat{z} \in \mathscr{Z}$ (this is the inverse problem)

$$
\begin{array}{rccc}
f: & \mathscr{H}^N & \longrightarrow & SE(3) \\
& H & \longmapsto & M \\
& | & & | \quad \uparrow \\
& \phi_{\mathscr{H}^N} & & \phi_{SE_3} \mathrm{Ip} \\
& \downarrow & & \downarrow \quad | \\
& \mathscr{W} & \longrightarrow & \mathscr{Z} \\
& \phi_{\mathscr{H}^N}(H) & \longmapsto & \phi_{SE_3}(M)
\end{array}
$$

Figure 6: Sheme of the method experimented: Ip denotes *Inverse Problem*

Obviously as a learning method, KCCA is computed from a *learning set* a tested thanks to a *testing set*. The learning set is denoted $\mathcal{L} = \{(H_1, M_1), \ldots, (H_L, M_L)\}$ and the testing set $\mathcal{T} = \{(H_1, M_1), \ldots, (H_T, M_T)\}$.
The remaining of the section is organized as follow: in section 3.2 presents kernel method we used, including the KCCA, regardless to our work space (Hough space and $SE(3)$). Then, in section 3.3 metrics problems related to the work spaces are presented. Section 3.4 give an overview of how going back from the RKHS to the space of camera motion. Finally, in section 3.5 results and conclusion about the method are given.

3.2 Overview of the kernel method

We review with outloss of generality the general idea of kernel method, and more particularly the KPCA (Kernel Principal Component Analysis) and KCCA (Kernel Canonical Correlation Analysis). [8]

3.2.1 Kernel method: outline

Let $\mathscr{X}$ be a space on which we have a distance $d_{\mathscr{X}}$ between its elements $\mathbf{x}$ and denote the mapping $\phi = \phi_{\mathscr{X}} : \mathscr{X} \longrightarrow \mathscr{W}$ where $\mathscr{W}$ is an infinite dimensional Hilbert space in which the scalar product is obviously well defined,

: $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_j(\mathbf{x}), \ldots)$. Note that in practice as well the mapping ϕ as its inverse ϕ^{-1} is unfortunately unknown. Finally, let assume a feature vector $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_L}\} \in \mathscr{X}^L$. In order to use linear algorithm with non linear data, the mapping ϕ is used as follow:

$$g(x) = \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}) + b = \sum_{l=1}^{L} \gamma_l \langle \phi(\mathbf{x}), \phi(\mathbf{x_l}) \rangle + b \qquad \text{(dual representation)} \quad (8)$$

Then, $K(\mathbf{x}, \mathbf{y})$ is a *Kernel function* if $\forall \mathbf{xy} \in \mathscr{X}$

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \quad (9)$$

with the following properties:

Symmetry $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \langle \phi(\mathbf{y}), \phi(\mathbf{x}) \rangle = K(\mathbf{y}, \mathbf{x})$

Cauchy-Schwarz inequality $K(\mathbf{y}, \mathbf{x})^2 = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle^2 \leq ||\phi(x)||^2 ||\phi(y)||^2 = K(\mathbf{x}, \mathbf{x}) K(\mathbf{y}, \mathbf{y})$

Choice of K There exits different choices for K and the most commonly used is the *Gaussian kernel* defined by:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{d_{\mathscr{X}}{}^2}{2\sigma^2}}$$

Kernel method is very convenient thanks to the Mercer's theorem. Indeed, it offers an easy way to map data into a feature space $F \supseteq \phi(\mathscr{X})$ based on a similarity measure between the elements of the $\mathscr{X}$, by computing $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$. We denote the *Graam matrix* the matrix as : $\mathbf{K} = (K(\mathbf{x_{j_1}}, \mathbf{y_{j_2}}))_{j_2, j_2 = 1}^{L}$.
From this point, it become possible to compute classical linear algorithms like PCA or CCA is the space $\mathscr{W}$ in particular thanks to the scalar product well defined in Hilbert spaces.

3.2.2 KPCA - Kernel Principal Component Analysis

We present quickly the KPCA algorithm [32] that compute a PCA in the space $\mathscr{W}$ since it is useful for the KCCA [23] (section 3.2.3).
As in the traditional case, the KPCA aims at computing the eigenvalue of the covariance matrix. $C = \frac{1}{L} \sum_{l=1}^{L} \phi(\mathbf{x_l}) \phi(\mathbf{x_l})^T$ such that: [2]

$$\lambda \mathbf{V} = C \mathbf{V} \quad (10)$$

[2] The PCA is computed on data centered. See appendix in [32] for centering kerneled data

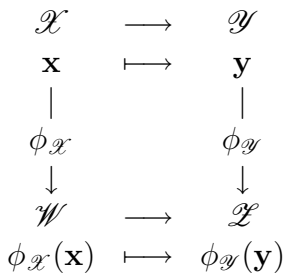


Figure 7: Scheme for the KCCA

where the eigenvector can written $\mathbf{V} = \sum_{j=1}^{L} \alpha_j \phi(\mathbf{x_j})$ ($\alpha = (\alpha_1, \ldots, \alpha_L)$). Then, the column vectors $(\alpha_1, \ldots, \alpha_L)$ are computed by solving the eigen problem:

$$L\lambda\alpha = \mathbf{K}\alpha \tag{11}$$

for non-zeros eigenvalues. The p vector among $(\alpha_1, \ldots, \alpha_L)$ that have non-zeros eigenvalues are normalized such that $\langle \mathbf{V}^k, \mathbf{V}^k \rangle = 1$ where $\mathbf{V}^k$ is the k^{th} column of matrix $\mathbf{V}$: it gives $\lambda_k \langle \alpha_k, \alpha_k \rangle = 1$. Finally the coordinate of any test point $\mathbf{x}$ onto the principal component can be easily computed:

$$\langle \mathbf{V}^k, \phi(\mathbf{x}) \rangle = \sum_{j=1}^{L} \alpha_j^k \langle \phi(\mathbf{x_j}), \phi(\mathbf{x}) \rangle = \sum_{j=1}^{L} \alpha_j^k K(\mathbf{x_j}, \mathbf{x}) \tag{12}$$

3.2.3 KCCA - Kernel Canonical Correlation Analysis

KCCA is quickly outlined in this section [23].
Let define $\mathscr{Y}$ with an associated distance $d_{\mathscr{Y}}$, and $\phi_{\mathscr{Y}} : \mathscr{Y} \longrightarrow \mathscr{Z}$ as we did respectively for $\mathscr{X}$, $d_{\mathscr{X}}$ and $\phi_{\mathscr{X}}$. We can thus construct the scheme presented in figure 7.
Now, let assume that we have L data point $X^{\mathcal{L}} = \{\mathbf{x_1}, \ldots, \mathbf{x_L}\}$ in the space $\mathscr{X}$ and $Y^{\mathcal{L}} = \{\mathbf{y_1}, \ldots, \mathbf{y_L}\}$ in the space $\mathscr{Y}$: they constitutes the *learning points*. We also define the orthogonal projection operator $P_{\mathbf{a}}(.)_A$: $P_{\mathbf{a}}(\mathbf{b})_A$ is the projection of vector $\mathbf{b}$ onto vector $\mathbf{a}$ in the space A. As in the classical case, KCCA tries to find triplets $(w_i, z_i, \lambda_i) \in (\mathscr{W} \times \mathscr{Z} \times [0, 1])$ such that

$$\left(P_{w_i}\left(\phi_{\mathscr{X}}\left(X^{\mathcal{L}}\right)\right)_{\mathscr{W}}\right)_{i \in \mathbb{N}}$$

and

$$\left(P_{z_i}\left(\phi_{\mathscr{Y}}\left(Y^{\mathcal{L}}\right)\right)_{\mathscr{Y}}\right)_{i \in \mathbb{N}}$$

are maximally correlated with respective correlation coefficients $(\lambda_i)_{i\in\mathbb{N}}$. $(w_i)_{i\in\mathbb{N}}$ and $(z_i)_{i\in\mathbb{N}}$ are called the canonical vectors. Obviously, the basis $(w_i)_i$ and $(z_i)_i$ should be orthogonal. In order to achieve such a thing, the coordinates of

$$(\{\phi_{\mathscr{X}}(x_1),\ldots,\phi_{\mathscr{X}}(x_L)\})$$

has to be expressed in the basis of the principal component computed from $\phi_{\mathscr{X}}(X^{\mathcal{L}})$. (Same remark for the space $\mathscr{Y}$). Once the KCCA has been computed, if a new point $\mathbf{x}^t$ from the *test set* appears, it is projected into the basis $(w_i)_i$ of $\mathscr{W}$ and its prediction is given by applying the correlation coefficients λ_i, giving finally the component onto the basis $(z_i)_i$. In a practical point of view, here is how the KCCA is realized:

- The components of $\Phi_{\mathscr{X}} = [\phi_{\mathscr{X}}(\mathbf{x_1}),\ldots,\phi_{\mathscr{X}}(\mathbf{x_L})]$ are computed in the basis of principal components $\mathbf{U}_{\mathscr{X}} = \Phi'_{\mathscr{X}}\mathbf{A}_{\mathscr{X}}$ ($\mathbf{A}_{\mathscr{X}}$ holds the expansion coefficients), giving $\mathbf{C}_{\mathscr{X}} = \Phi_{\mathscr{X}}\mathbf{U}_{\mathscr{X}} = \mathbf{K}_{\mathscr{X}}\mathbf{A}_{\mathscr{X}}$. Same thing for $\mathscr{Y}$
- The canonical vectors $(w_i)_i$ and (z_i) are computed in such way that the canonical variates $\mathbf{a_i} = \mathbf{K}_{\mathscr{X}}\mathbf{A}_{\mathscr{X}}w_i$ and $\mathbf{b_i} = \mathbf{K}_{\mathscr{Y}}\mathbf{A}_{\mathscr{Y}}z_i$ are maximmaly correlated:
$$\max\left(\frac{\langle\mathbf{a_i},\mathbf{b_i}\rangle}{||\mathbf{a_i}||||\mathbf{b_i}||}\right)$$
which lead to the following contrained optimization problem:

$$\begin{array}{ll} argmax & w_i\mathbf{C}'_{\mathscr{X}}\mathbf{C}_{\mathscr{Y}}z_i \\ \text{subject to} & w_i\mathbf{C}'_{\mathscr{X}}\mathbf{C}_{\mathscr{Y}}w_i = z_i\mathbf{C}'_{\mathscr{X}}\mathbf{C}_{\mathscr{Y}}z_i = 1 \end{array} \tag{13}$$

The main difficulties of such a method is the choice of the dissimilarity $d_{\mathscr{X}}$ (respectively $d_{\mathscr{Y}}$) between the elements of $\mathscr{X}$ (respectively $\mathscr{Y}$). This is particularly true in our original problem in which $\mathscr{X} = \mathscr{H}^N$ and $\mathscr{Y} = SE(3)$. This is precisely the matter of the following section to explain our choices of metrics in suhc spaces.

3.3 Metric related problems

From now on, we consider that $\mathscr{X} = \mathscr{H}^N$ and $\mathscr{Y} = SE(3)$ and the dissimilarity measure $d_1 = d_{\mathscr{H}^{\mathscr{N}}}$ and $d_2 = d_{SE_3}$ are to be developped.

3.3.1 Dissimilarity measure in $\mathscr{H}^N$: choice of d_1

Let denote $\rho_n = \rho(l_n)$ and $\theta_n = \theta(l_n)$ the Hough parameters of line l_n. First of all we need to define a "distance" $d_l(.,.)$ between two lines in an image. This is

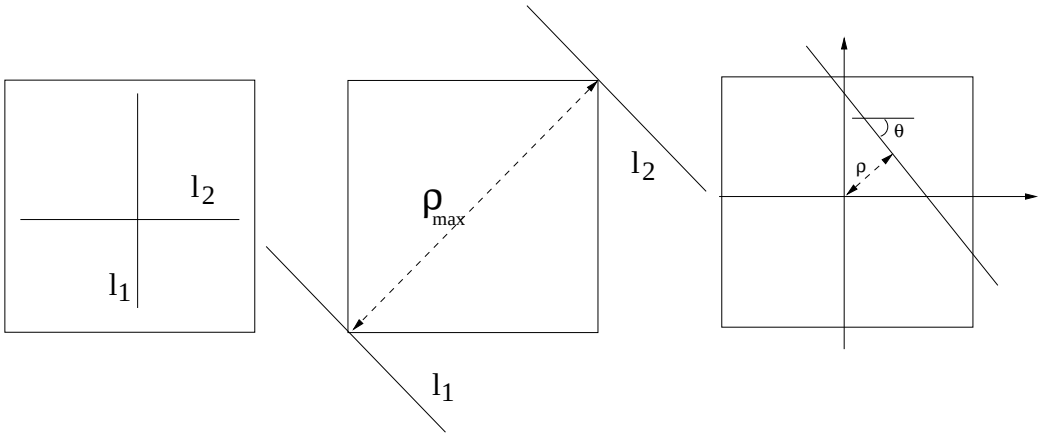


Figure 8: Two independent cases in which $d_l(l_1, l_2)$ should be maximal and have the same value, on the right : Hough parametrization chosen. $\theta \in [-\frac{\pi}{2}\frac{\pi}{2}]$ and $\rho \in \left[-\frac{\rho_{max}}{2}, \frac{\rho_{max}}{2}\right]$

not not an obvious problem since in the case of Hough parameters we have to deal with two kind of measurements: angular and non angular. Furthermore, $d_l\left(.,.\right)$ is necessarily related to the framework of the image. Indeed, $d_l\left(.,.\right)$ should take into consideration that the dissimilarity between two lines l_1 and l_2 is maximal and has the same value if we consider independently angular and non-angular part of the corresponding Hough parameters [3](see figure 8):

- $\theta_1 \perp \theta_2, \quad \forall \rho_1, \rho_2$
- $\rho_1 = -\frac{\rho_{max}}{2}$, $\rho_2 = \frac{\rho_{max}}{2}$ and $\rho_1 = \rho_2 = \frac{\pi}{4}$

where ρ_{max} is the length of the diagonal. Last but not the least, $d_l\left(.,\right)$ should be homogeneous. We can straightforwardly assert (see figure 8 for the Hough parametrization chosen):

$$d_l\left(l_1, l_2\right) = \tan\left(\frac{d_\theta\left(\theta_1, \theta_2\right)}{2}\right) + \tan\left(\frac{\pi}{4}\frac{|\rho_1| + |\rho_2|}{\rho_{max}}\right) \tag{14}$$

where $d_\theta\left(\theta_1, \theta_2\right) = \min\left(|\theta_2 - \theta_1|, \pi - |\theta_2 - \theta_1|\right)$. We can now define the dissimilarity measure $d_1\left(H_u, H_v\right), \quad H_u, H_v \in \mathcal{L}$. Let l_n^u be the n^{th} line of the u^{th} image (u^{th} element of the learning set $\mathcal{L}$). As previously mentionned, $\left(l_n^u\right)_{u\in[1,L]}$ is

[3]We consider that we work in normalized coordinates, with same ratio along x-axis and y-axis of the image

the projection of the same n^{th} 3-lines in all images from the learning set. Finally, we have:

$$d_1\left(H_u, H_v\right) = \sum_{n=1}^{N} d_l\left(l_n^u, l_n^v\right) \tag{15}$$

Note that we can imagine small variations of the measure proposed in the last lines.

3.3.2 Dissimilarity measure in $SE(3)$: choice of d_2

In this section we based our dissimilarity measure on Riemannian metrics on $SO(3)$ and $SE(3)$ [10, 5]. First of all we give "relative" definitions of spaces $GL_+(3)$, $GA_+(3)$, $SO(3)$ and $SE(3)$. $GL_+(3)$ denote the set of all positive-definite 3×3 real matrices. Thus we have:

$$SO(3) = \left\{R | R \in GL_+(3), R^T R = I\right\} \tag{16}$$

$$GA_+(3) = \left\{B | B = \begin{bmatrix} M & d \\ 0 & 1 \end{bmatrix}, M \in GL_+(3); d \in \mathbb{R}^3\right\} \tag{17}$$

$$SE(3) = \left\{A | A = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, R \in SO(3); d \in \mathbb{R}\right\} \tag{18}$$

Without loss of generality, we present [left-invariant] metrics in this spaces. Again, a lot of details can be found in [5, 10]. Let $G_1, G_2 \in GL_+(3)$. Thus, we have:

$$\begin{aligned} ||G_1 - G_2||^2_{GL_+} &= \langle G_1 - G_2, G_1 - G_2\rangle_{GL_+} \\ \text{where } \langle X, Y\rangle_{GL_+} &= Tr\left[X^T Y W\right] \end{aligned} \tag{19}$$

and W is a symmetric positive-definite 3×3 matrix. Any element $G \in GL_+$ can be projected onto $SO(3)$[5] by computing the SVD decomposition of GW: $GW = U\Sigma V^T$. Then, the projection $R \in SO(3)$ of G is given by $R = UV^T$.
In the same way, let $A_1, A_2 \in GA_+(3)$ and we have:

$$\left\{\begin{array}{rcl} ||A_1 - A_2||^2_{GA_+} &=& \langle A_1 - A_2, A_1 - A_2\rangle_{GA_+} \\ \text{where } \langle X, Y\rangle_{GA_+} &=& Tr\left[X^T Y \widetilde{W}\right] \\ \text{and } \widetilde{W} &=& \begin{bmatrix} W & a \\ a^T & \omega \end{bmatrix} \end{array}\right. \tag{20}$$

We can compute the projection $M \in SE(3)$ of any element $A \in GA_+(3)$. If

$$A = \begin{bmatrix} B_1 & B_2 \\ 0 & 1 \end{bmatrix}, \quad B_1 \in GL_+(3); \quad B_2 \in \mathbb{R}^3 \tag{21}$$

$$B_1 W = U\Sigma V^T \text{ (singular value decomposition)} \tag{22}$$

then:

$$M = \begin{bmatrix} UV^T & B_2 \\ 0 & 1 \end{bmatrix} \tag{23}$$

In our experiments, we chose $W = I$, and if we set $a = \mathbf{0_3}$, equation 20 can be expressed as:

$$||A_2 - A_1||_{GA_+} = ||R_2 - R_1||^2_{GL_+} + \omega||t_2 - t_1||^2 \qquad \text{where } A_k = \begin{bmatrix} R_k & t_k \\ \mathbf{0}_3 & 1 \end{bmatrix}$$

and ω is become a parameter that balance the relative proportion of the rotational and translational part of the motion. In figure 9, we show a way that use the previous result in order to approximate a dissimilarity measure in $SE(3)$. Let M_1 and M_2 be two points in $SE(3)$. They are also in $GA_+(3)$. The geodesic in $GA_+(3)$ [in the sense of $||.||_{GA_+}$] between the two points is easily obtained. Then if we sample this geodesic and project each sample point onto $SE(3)$, we can get a good approximation even using $||.||_{GA_+}$. Let S be the number of sample points, $(t_s)_{s\in\{1,\dots,S\}}$ be the discrete variable such that $t_s = \frac{s}{S}$, and $P_{SE}(.)$ be the projection of a $GA_+(3)$ matrix element onto $SE(3)$. If we write

$$\left(M_i = \begin{bmatrix} R_i & T_i \\ 0 & 1 \end{bmatrix}\right)_{i\in\{1,2\}}$$

and

$$P_{SE}(s) = P_{SE}\left(\begin{bmatrix} R_1 + (R_2 - R_1)\, t_s & T_1 + (T_2 - T_1)\, t_s \\ \mathbf{0_3} & 1 \end{bmatrix}\right)$$

then we have the following dissimilarity measure:

$$d_2(M_1, M_2) = \sum_{s=1}^{S-1} ||P_{SE}(s+1) - P_{SE}(s)||_{GA_+} \tag{24}$$

3.4 The inverse problem : from $\mathscr{Z}$ to $SE(3)$

As previously explained, the prediction is achieve between in the space $\mathscr{Z}$ while we need a prediction of the position of the camera in $SE(3)$. Therefore, only spaces of the right part of scheme 6 is considered in this section: $SE(3)$ and $\mathscr{Z}$ Once we have the prediction in the basis of canonical vector (z_i), we can reproject the coordinates of the projection onto the principal component. Then, we have straightforwardly:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in SE(3)}{argmin} \left|\left|\mathbf{A}^T_{SE_3}\mathbf{K}(\mathbf{y}) - p\right|\right|^2 \tag{25}$$

where:

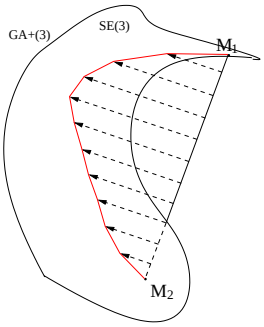


Figure 9: approximating the dissimilarity measure in $SE(3)$

- $\mathbf{K}(\mathbf{y}) = [K(\mathbf{y}, \mathbf{M_1}), \ldots, K(\mathbf{y}, \mathbf{M_L})]^T$
- p is the prediction expressed in the principal component basis of $\mathscr{Z}$

The initial estimate $\mathbf{y_{initial}}$ for this optimization procedure is chosen among the learning set $\{M_1, \ldots, M_L\}$ such that :

$$\mathbf{y_{initial}} = \underset{\mathbf{y} \in \{M_1, \ldots, M_L\}}{argmin} \left|\left|\mathbf{A}_{\mathscr{Y}}^T \mathbf{K}(\mathbf{y}) - p\right|\right|^2 \tag{26}$$

Obviously the KPCA has been performed without including the test samples. Figure 10 shows some steps of the optimization.

3.5 Results and conclusion

The method previously presented has been tested by simulation. A set of camera observing a line-based object has been generated. This is shown in figure 11 in which $m = 124$ (size of the learning set). We achieved tests with different values of m. Since the resolution of the inverse problem gives reasonable results, we tried to work closer on the prediction part. Figure 12 shows some results obtained by predicting in the $(z_i)_i$-basis. In case of $m = 124$, the prediction is quite correct for the 15 fisrt component which could be sufficient to recover the position of the camera by solving the inverse problem. In order to increase the number of components predicted, the size of the learning set has to be hugely raised: to have 25 correctly predicted component, the value of m is changed to 1000. Apart from the fact $m = 1000$ is unthinkable in a pratical case, it is highly computational which is a very negative point. Furthermore, even though predictions on at least 6 components in $(z_i)_i$ are a priori sufficient to recover the 6 intrinsec parameters [by solving the inverse problem], they have to be very precise.
Nervertheless, the choice of the dissimilarity measure in $SE(3)$ seems to be suitable since we obtained successful results while solving the inverse problem.

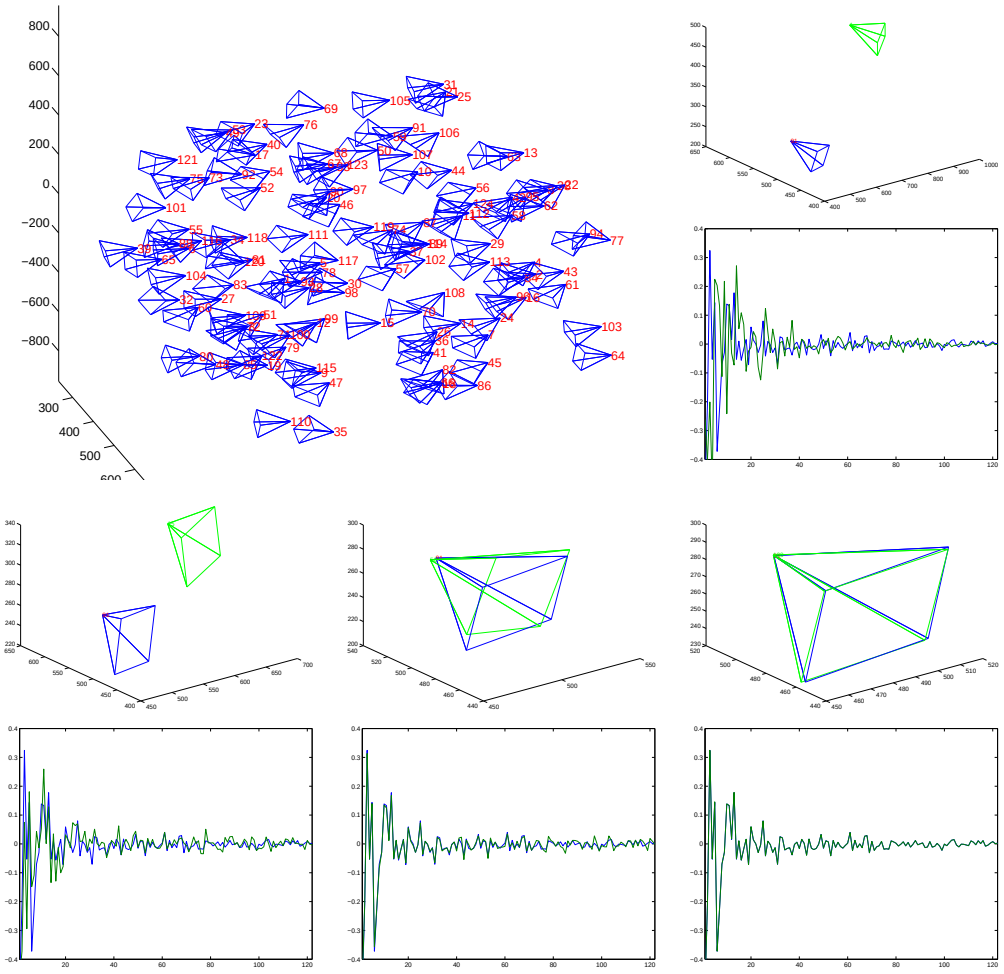


Figure 10: Results of the inverse problem - on the top left: the learning set used to compute de KPCA. Other: Some steps of the optimization algorithm used: Camera position and expansion on the principal component in $\mathscr{Y}$ - The first on the top left correspond to the initial estimate In blue: Real. In Green: Prediction

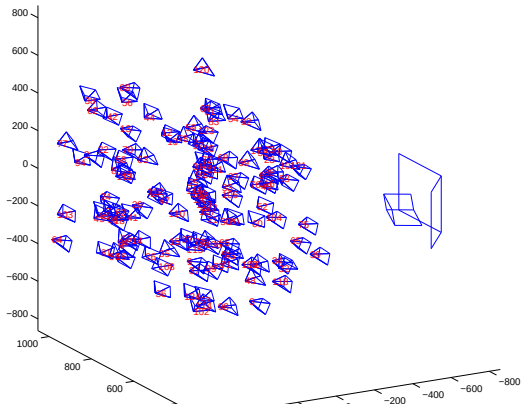


Figure 11: Learning set used for the simulation

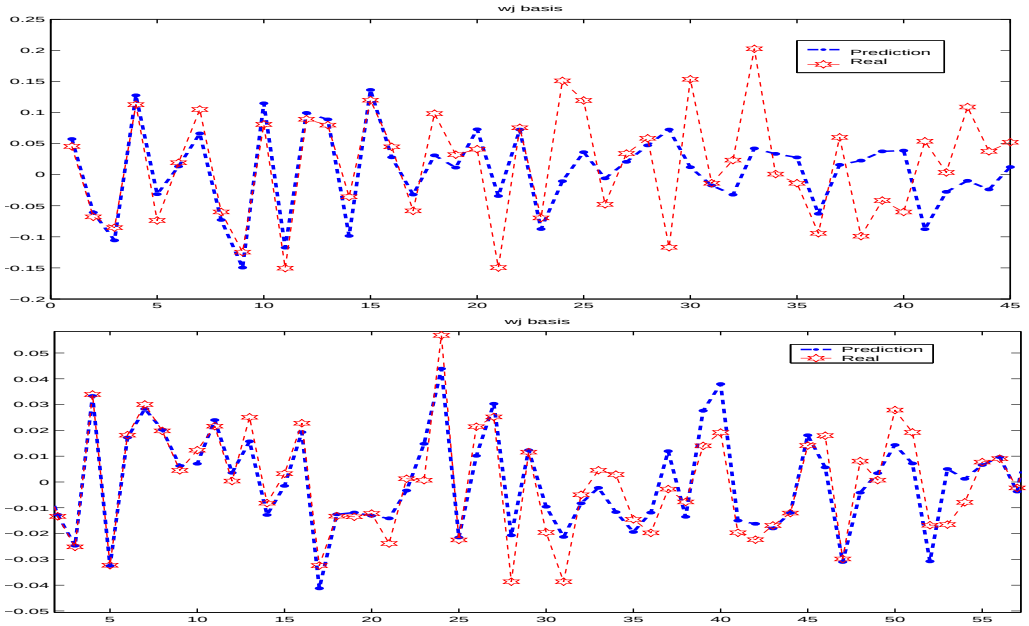


Figure 12: Two examples of prediction for 2 values of m (size of learning set)- Above: 124. Below: 1000

4 Inference from complete Hough/Radon space

4.1 Objectives & Problem formulation

Contrary to the previous works, lines are not only characterized by the Hough parameter but also by features extracted in the Radon space. Our method consists of a learning and an inference step. During the learning stage, the scene is learnt from an image sequence and its corresponding 3D reconstruction. A geometry-based learning is achieved by recovering geometric relations between lines and consequently between their projections. In parallel to the feature-based learning, 3d lines are associated through AdaBoost learners with their 2D projection in the Radon space (local maxima). Such an information space is used within a matching process to recover camera's pose from a new image. Matching between plausible line candidates in a new image dictate multiple correspondences between the 2D new image lines and the 3D reconstructed lines. The most probable configuration in terms of appearance while satisfying geometric consistency constraints provides the camera position. The overview of such an approach is shown in [Fig. (13)].

Let us consider a viewer centered coordinate system that is defined with the camera lens center or the observer located at the origin and such that the view axis is collinear to the z axis. We further assume that the image plane is perpendicular to the view axis. Using the perspective model, the image of any point in space is equal to the intersection of the image plane and the line joining the point to the center of the camera lens.

The main stream of research in 3D reconstruction and pose estimation has

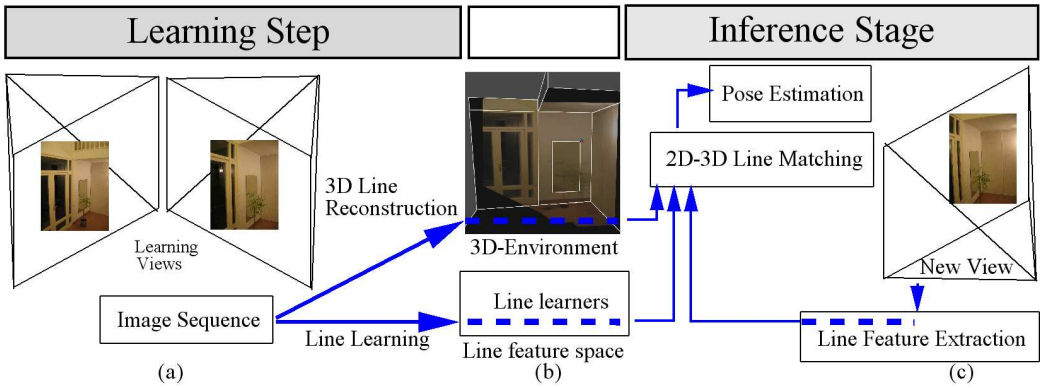


Figure 13: Overview of the proposed pose estimation approach where both learning and estimation steps are delineated.

been devoted to point correspondence [15, 28, 30]. Line correspondences could be an efficient alternative to such an approach [25, 24]. Such a feature space inherits the advantage of being more robust than point correspondences as well as more global. On the other hand, line tracking algorithms are computationally intensive and low sampling frequency and long time delays can therefore be expected. Such a limitation can be addressed using image transformations like the Hough [11] and the Radon space [34] which project images into convenient lines spaces.

The remainder of the current section is organized in the following fashion: Section 2.3 described our approach toward line tracking. Since lines tracked through a video sequence, their 3d information is recovered and the 3d-2d relation can be indirectly learned by a boosting algorithm as presented in section 4.2. Section 4.3 is devoted to inference and pose estimation and finally a discussion for this approach is given in the conclusion of this document in section 5.

4.2 3D-2D Line Relation through Boosting

As previously mentionned, the first step is compute a three dimensional model of the scene and more particularly the lines. 3d reconstruction from image sequences of video sequence has been widely studied in the past years. Since it is not in the scope of this document, this part is not developped. Figure 14 a 3d reconstruction of an indoor scene based on lines.

Once the scene and 3D lines have been reconstructed [Fig. 14], one would like to establish a connection between such 3D lines and their corresponding projections. Since our approach is both features and geometric based, we aim at learning both kind of constraints.

First, geometrical constraints can be straight and naturally deduced from the 3D reconstructed scene implying 2d constraints on the projected lines. Since extraction of the relative geometry is not critical - once 3D reconstruction has been completed -, more attention is to be paid on feature extraction, learning and modeling.

Let us consider that our feature learning stage consists of $\mathcal{L} = \{l_1, l_2, ..., l_n\}$ 3D lines, and our training consists of c images. Without loss of generality we assume that such geometric elements were successfully detected within this c images. Let $\mathcal{P}_k = \{p_k^1, p_k^2, ..., p_k^c\}$ being the projections in the radon space of line l_k at these c images. Such projections correspond to the 2D local radon patches represented as d-dimensional vectors.

Traditional statistical inference techniques can be used to recover a distribution of such d-dimensional vectors. To this end, one can consider simple Gaussian assumptions and classical dimensionality reduction techniques like principal component analysis. Such a selection could fail to account for the highly no-linear structure of the Radon space and so of the corresponding features. Furthermore,

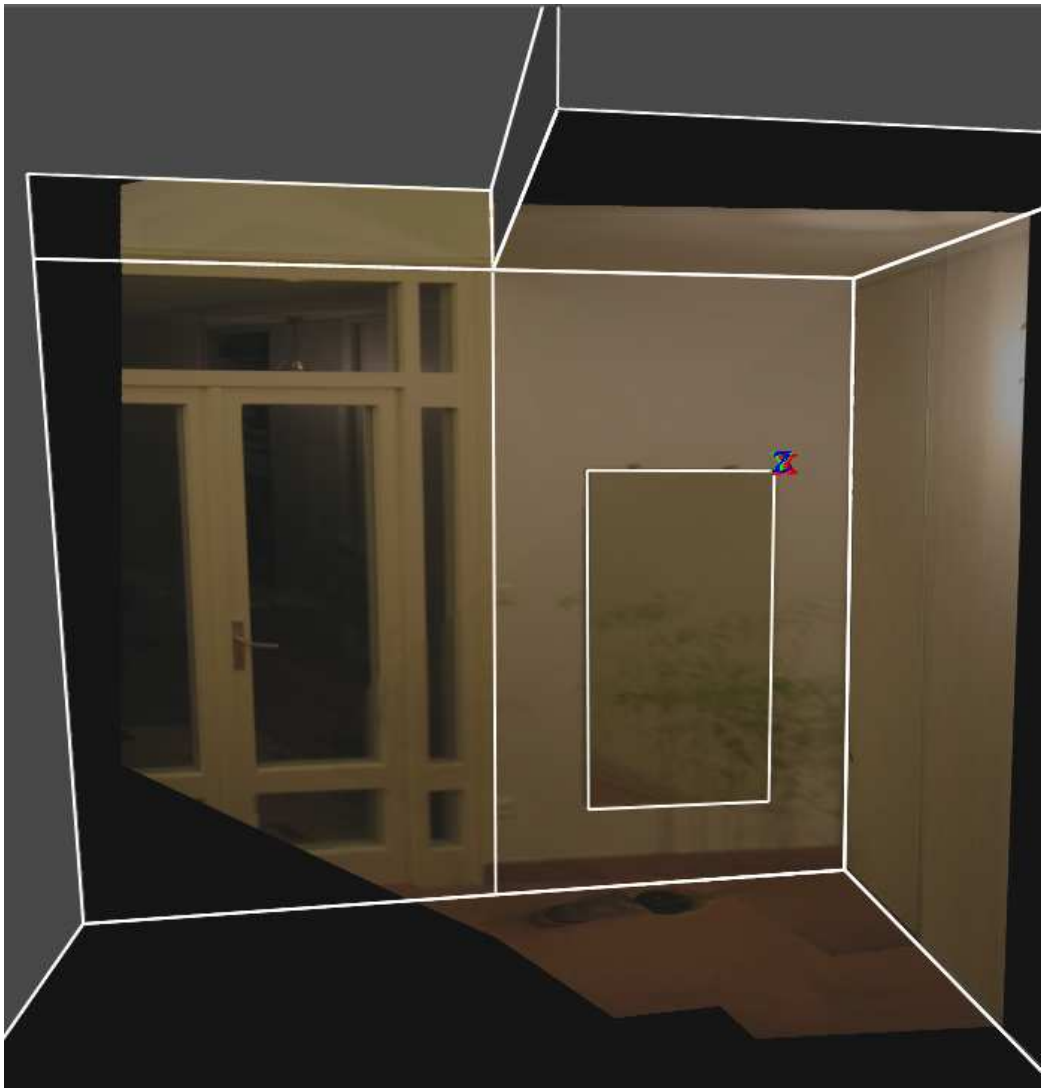


Figure 14: 3d reconstruction of an indoor scene

since recovering a training shots from all possible virtual positions of the observer it is almost impossible, one should also account for sparse observations and learning from small training sets. Therefore, more advanced classification techniques are to be considered that are able to cope with some of the above limitations.

Our basic classifier consists of given two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ find an appropriate transformation/function F that can measure the distance between a sample p and these classes $F(C_k, p)$. To this end, within the context of our application one can consider n bin classification problems F_k,

$$F_k(p) = \begin{cases} 1, & p \in C_k \\ 0, & p \in C_j,\ j \neq k \end{cases}$$

In other words, we are looking for a way to compute the boundary of a binary partition between the features corresponding to line l_k versus the others. Stump classification can deal with this problem: it tests binary partitions along all the d dimensions and all possible thresholds. The model is given by:

$$\begin{aligned} R =& \{\alpha_0 \mathbb{1}_{x_j<\tau} + \alpha_1 \mathbb{1}_{x_j \geq \tau} : j \in 1, \ldots, d, \tau \in \mathbb{R}, \\ & \alpha_0 \in [0;1], \alpha_1 \in [0;1]\} \end{aligned} \tag{27}$$

The threshold τ^* and the dimension j^* that minimizes the desired criteria $\mathcal{W}(j, \tau)$ are kept to form the partition parameters. The reader can refer to [4] to get further details about stumps and more particularly about the criteria $\mathcal{W}$ we used.

Consequently, stump classification returns a function f_m that defines a partition of the space according to an hyperplane which is orthogonal to the canonical basis of $\mathcal{X}$:

$$f_m = f_{m,<} \mathbb{1}_{x \in \mathcal{X}^{<}_{j,\tau}} + f_{m,\geq} \mathbb{1}_{x \in \mathcal{X}^{\geq}_{j,\tau}}$$

Stumps were implemented and tested with a synthetic data set formed with a video sequence of a basic 3D structure. Towards producing a realistic test case, a set of perturbations (random lines) are introduces in the 3D scene [Fig. (17)]. In order to account for possible sensor noise, the corresponding video images are convolved with a Gaussian operator (white noise), and additional lines are also added in the observation set. Radon transformations of such images are used to recover local patches that guide the learning step while a test set is also created. The classification error for all experiments which were conducted was close to 0.5. Therefore, one can conclude that bin classification on such a space induces a high risk in pose estimation.

One can overcome such a limitation through the transformation of the stump classifier into a "weak" learner. In addition, the learning algorithm should determine the origin of the sample as accurate as possible by the use of a multitude of "weak" learners. AdaBoost [17, 18] is one of the most prominent techniques

to address such a task among others such as neural networks [11] and support vector machines [35]. Boosting improves significantly the accuracy of any given learning algorithm, even in the case of a "weak" learner. Such techniques have a number of interesting empirical properties. It has been shown [18] that boosting does not perfom an overfit to the training data.

The general idea of boosting is to **1-** repeatedly use a "weak" learner [stumps returning a regression function f_m in our case] with some weights w_i^m on the training data - m *beeing the iteration index* - **2-** focus on misclassified data from one iteration to the next through the update of w_i^m:

$$w_i^m = \frac{w_i^{m-1} e^{-Y_i f_m(X_i)}}{\mathrm{K}} \quad \begin{array}{l} \forall i \in \{1,\ldots,N\} \\ \text{K: normalizing constant} \end{array} \tag{28}$$

where Y_i is the classication corresponding to the feature X_i, (X_i, Y_i) beeing an element of the learning and N its size.

Then, at each step a weight c_m associated with the current learner is determined according to the corresponding classification performance. The final classification is given by the thresholded regression function $\mathbb{1}_{G_M(x)>T}$, $G_M(x)$ beeing the weighted combination of the "weak" learners:

$$G_M(x) = \sum_{m=1}^{M} c_m f_m \tag{29}$$

This is a slightly modified version of the "real AdaBoost algorithm" [31, 4] presented in figure 15. Indeed, at the end of the "real" AdaBoost algorithm, the decision is based on $\mathbb{1}_{G_M(x)\geq\frac{1}{2}}$ implying implicitly a fixed threshold on the regression function $G_M(x) = \sum_{i=1}^{M} c_i f_i$. Instead of doing this and since $G_M(x)$ is by definition piece-wise constant, we preferred to choose dynamically the threshold T among the finite set of possible values so that the error can be decreased.

Finally, the feature learning stage outputs n classifiers

$$\mathcal{S}^n = \left\{ \mathbb{1}_{G_M^1(x)>T_1}, \ldots, \mathbb{1}_{G_M^k(x)>T_k}, \ldots \mathbb{1}_{G_M^n(x)>T_n} \right\}$$

-one for each line- that are going to be used for line inference and pose estimation.

4.3 Line Inference & Pose estimation

Line inference consists of recovering the most prabable 2D patches-to-3D lines configuration using the set of classifiers

$$\mathcal{S}^n = \left\{ \mathbb{1}_{G_M^1(x)>T_1}, \ldots, \mathbb{1}_{G_M^k(x)>T_k}, \ldots \mathbb{1}_{G_M^n(x)>T_n} \right\}$$

(X, Y) forms the learning set, $X \in \mathcal{X} = R^d$ is a feature and Y the corresponding true classification decision
Start with weights $w_i^0 = \frac{1}{N}$ for any $i \in \{1, \ldots, N\}$.
For $m = 1$ **to** M **do**

- Determine $j \in \{1, \ldots, N\}$ and $\tau \in \mathbb{R}$ minimizing $\mathcal{W}_{w^{m-1}}(j, \tau)$.
- Choose $f_m = f_{m,<} \mathbb{1}_{x \in \mathcal{X}_{j,\tau}^{<}} + f_{m,\geq} \mathbb{1}_{x \in \mathcal{X}_{j,\tau}^{\geq}}$ where

$$\begin{cases} f_{m,<} & \triangleq & \frac{1}{2} \log \left(\frac{\mathbb{P}_{w^{m-1}}(Y=1; X \in \mathcal{X}_{j,\tau}^{<}) + \beta}{\mathbb{P}_{w^{m-1}}(Y=0; X \in \mathcal{X}_{j,\tau}^{<}) + \beta} \right) \\ f_{m,\geq} & \triangleq & \frac{1}{2} \log \left(\frac{\mathbb{P}_{w^{m-1}}(Y=1; X \in \mathcal{X}_{j,\tau}^{\geq}) + \beta}{\mathbb{P}_{w^{m-1}}(Y=0; X \in \mathcal{X}_{j,\tau}^{\geq}) + \beta} \right) \end{cases}$$

  and $\beta = \frac{1}{4N}$
- Set $w_i^m = \frac{w_i^{m-1} e^{-Y_i f_m(X_i)}}{\text{Cst}}$ for any $i \in \{1, \ldots, N\}$, where Cst is the normalizing constant.

EndFor

- Output the classifier $\mathbb{1}_{G_M(x) \geq \frac{1}{2}} = \frac{1+\text{sign}[G_M(x)]}{2}$ where $G_M(x) = \sum_{i=1}^{M} c_i f_i$

Figure 15: "Real" AdaBoost [31] using stumps as defined in [4]

. In this section, we first explore the straighforward solution and then we propose an objective function that couples the outcome of the weak learners with geometric constraints inherited from the learning stage. Such an objective function also solves pose estimation since the optimal camera parameters refer to its lowest potential.

In order to validate the performance of the AdaBoost classifier, we have created a realistic synthetic environment where inference results can be compared with the true configuration. The feature vector for one-preselected line has been learnt, and the corresponding classifier was tested with new images. Results for the 30 first iterations of the real AdaBoost are presented in [Fig. (16)]. We can clearly make several observations. First, learning error converges to zero while the error of the classification in the test remains stable. Such a remark is consistent with the expected behavior of the classifier; boosting does not overfit as

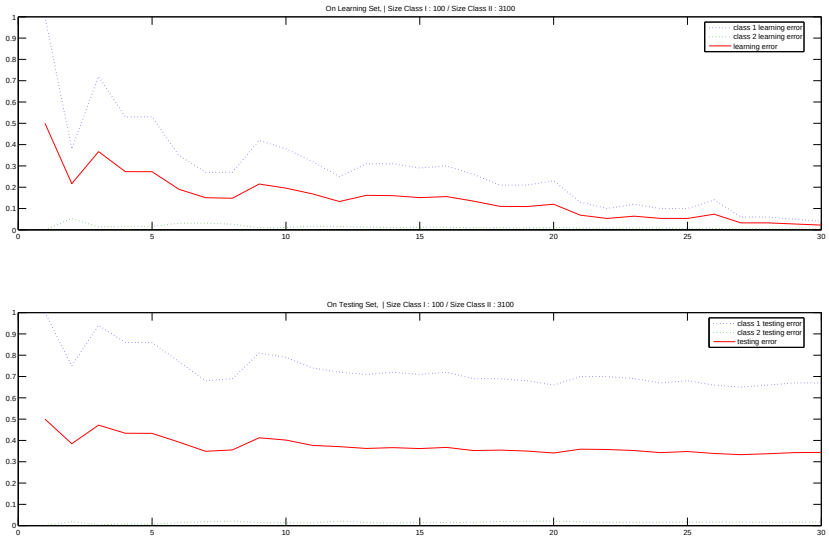


Figure 16: Error rates during the 30 iterations of real AdaBoost: (top row) learning set, (bottom row) test set. Red: mean error - Blue: error rate of Class I, class of the line learnt - Green: error rate of Class II, class of the other lines

previously mentioned. Then, samples from Class II are almost never misclassified while classification error of Class I is very important and therefore direct pose estimation is almost impossible. On top of that, one can claim that the lines that are visible change from one image to the next; therefore pose is ill-posed. Such a limitation can be dealt with the use of geometrical constraints encoded in the learning state during the 3D reconstruction step. Such an assumption could allow us to relax the AdaBoost, since classification errors become less significant once geometry is introduced.

A modified classification model is now constructed based on the previous oberservations. Let j be a new image outside the vido sequence. Any sample p such that $\left(G_M^k(p) > T_k\right)$ (Class I) is a potential match. Moreover, classification confidence depends on the distance of the data to be classified from the the boundary and so on the value of $\text{sd}^k(x) = G_M^k(x) - T_k$: the greater is $|\text{sd}^k(x)|$ the more confident is the classification. Thus, the easiest classification choice is:

$$\arg \max_{\substack{i \in \{1,\ldots,n\} \\ \text{st:} G_M^k(p_i^j) > T_k}} G_M^k(p_i^j) - T_k \tag{30}$$

The correspondance expressed in eqn. (30) is not sufficient since the most important value does not necessarily correspond to the real match. Let us assume for a line k, we are interested in the B best potential matches $\{p_{n_1}[k], \ldots, p_{n_B}[k]\}$. Such candidates are determine through the eqn. (30). If less than B lines verify the constraint $G_M^k(p_i[k])) > T_k \, \forall i$, then it is "relaxed" as earlier explained. In others words, lines misclassified are authorized to be taken into consideration by removing the constraint in eqn. (30). A weighting function $h(.)$ is also used to influence the importance of a potential match based on the quantity $\text{sd}^k(.)$.

Actually we want to express a geometrical constraint GC between the projections of C lines $\{l_{s_1}, \ldots, l_{s_c}, \ldots, l_{s_C}\}$ ($C < B$). For each lines s_c we keep the B best potential matches $\{p_{n_1}[s_c], \ldots, p_{n_b}[s_c], \ldots, p_{n_B}[s_c]\}$. Finally, the energy to be minimized is given by:

$$\min_{\substack{(i_1 \ldots \imath_C) \in \\ (\mathcal{A}_1, \ldots, \mathcal{A}_C)}} \sum_{c=1}^{C} h(\text{sd}^{i_c}(p_{i_c}[s_c])) \text{ subject to } \text{GC}(p_{i_1}[s_1], \ldots, p_{i_C}[s_C]) \quad (31)$$

where:

- $\mathcal{A}_c$ is the indice set of potential matches with line l_{s_c}
- $h(x)$ is as in our implementation inversely proportional to x. More complex model can however be imagined.

with GC being the geometric constraint. One can recover the lowest potential of such a cost function using classical optimization methods but at the sight of the small number of lines detected, we consider an exaustive search approach. Numerous formulations can be considered for the GC term. Corners are prominent characteristics of 3D scenes. Therefore, 3D lines going through the same point (that can also define an orthogonal basis) is a straighforward geometry-driven constraint. One can use such an assumption to define constraints in their projection space; that is:

$$\text{GC}(l_1, l_2, l_3) = |(l_1 \times l_2)^T \, l_3| \quad (32)$$

where $\times$ is the cross product of 2 projective points/lines and ^{T} is the transpose sign.

Such a term takes into account the scene context. Offices, buildings, etc. are scenes where the use of such a constraint is mostly justified (corners, vanishing points etc ...). For example in figure 18, the learning step of lines 1,2 and 3 gives a set $\left\{\mathbb{1}_{G_M^1(x)>T_1}, \mathbb{1}_{G_M^2(x)>T_2}, \mathbb{1}_{G_M^3(x)>T_3}\right\}$. If only feature constraint is used through eqn.

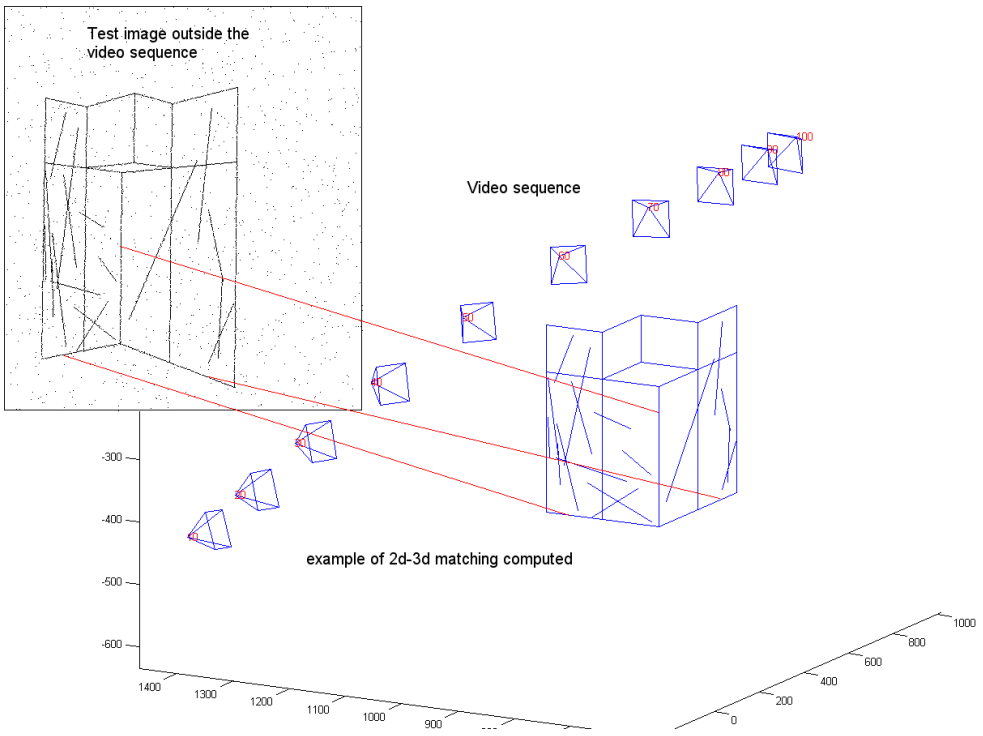


Figure 17: Example of learning results on synthetic data. Red lines show a matching of 3 lines using geometrical constraint

30, only line 2 is well matched. However, by using relaxation and the geometrical constraint associated to these lines, the algorithm retrieves the good matching. In more complex scenes, more advanced terms can be considered to improve the robustness of the method. Once the line correspondence problem has been solved, the pose parameters of the camera can be determine using a number of methods [13, 27, 7], but we choose to implement a fast efficient linear method presented in [3].

5 Conclusion & Discussion

In this document, we explored two approaches to pose estimation based on line configuration in images. The former tried to find a correlation between the space of line configuration and the space of rigid transformation. Interesting results could be shown be it was not powerful enough to plan to compute a pose estimation in a practical case. The latter gave more promising results and several experiments were conducted to determine the performance of the method. To this

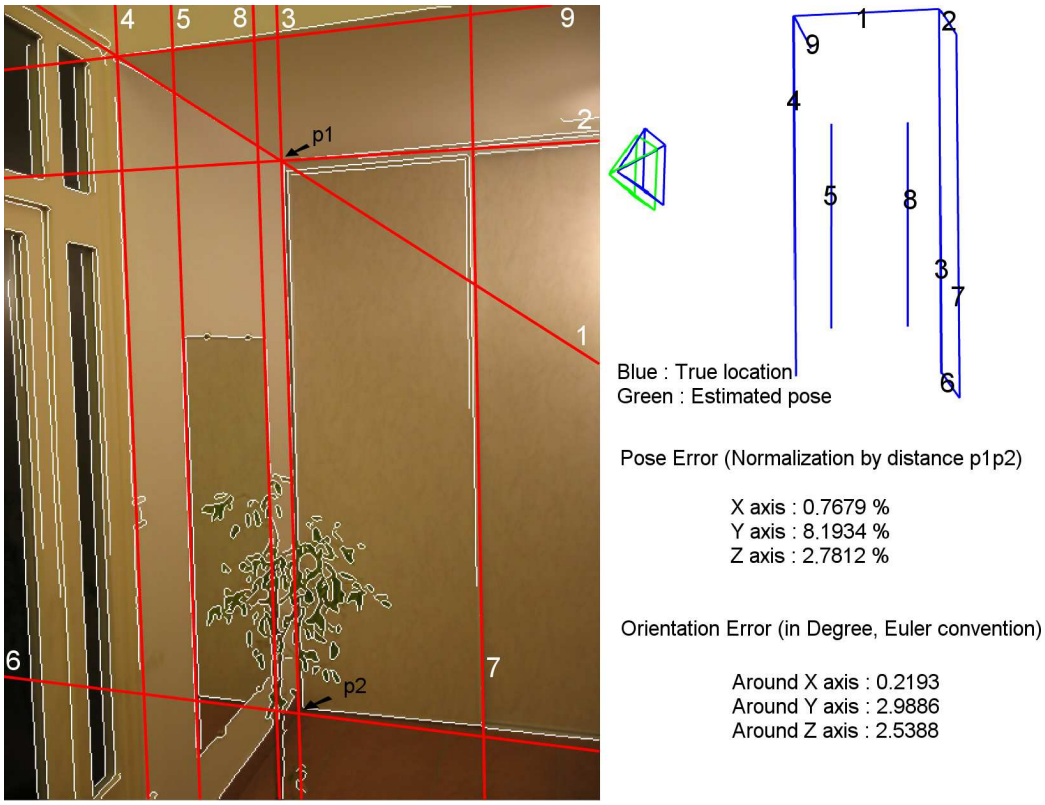


Figure 18: Final calibration: the image to be calibrated is overlayed by the edge map (in white) and the 3D line reprojection (in red)

end, first a video stream along with the corresponding 3D geometry [that can be recovered standard reconstruction techniques] of the scene were used to learn the model. Such a model refers to n classifiers with their features space being patches of the radon transformation of the original image. Then, new images of the same scene was considered and self-localization of the observer based on 2d-3d line matching [Fig. (17) & (18)] was performed.

In this paper, we have proposed a new technique to pose estimation from still images in known environments. Our method comprises a learning step where a direct association between 3D lines and radon patches is obtained. Boosting is used to model that statistical characteristics of these patches and weak classifiers are used to determine the most optimal match for a given observation. Such a classification process provides multiple possible matches for a given line and therefore a fast prunning technique that encodes geometric consistency in the process is proposed. Such additional constraints overcome the limitation of classification errors and increase the performance of the method.

Better classification and more appropriate statistical models of lines in radon space is the most promissing direction. The use of radon patches encode to some extend clutter and therefore separating lines from irelevant information could improve the performance of the method. Better tracking of lines through linear prediction techniques like kalman filter could improve the learning stage and make the method more appropriate for real-time autonomous systems. Last, but not least representing the camera's pose parameters using non-parametric kernel-based statistical models seems to be more suitable term to further develop the inference process.

References

[1] A. S. Aguado, E. Montiel, and M. S. Nixon. On the intimate relationship between the principle of duality and the hough transform. *Proc. Royal Soc. London*, A-456:503–526, 2000.

[2] O. Ait-Aider, P. Hoppenot, and E. Colle. Adaptation of lowe's camera pose recovery algorithm to mobile robot self-localisation. *Robotica*, 20(4):385–393, 2002.

[3] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 282–296, London, UK, 2002. Springer-Verlag.

[4] J.-Y. Audibert. *PAC-Bayesian Statistical Learning Theory*. PhD thesis, Université Paris 6, Francec, Jul 2004.

[5] C. Belta and V. Kumar. An svd-based projection method for interpolation on se(3). *IEEE Transactions on Robotics and Automation*, 18(3):334–345, Jun 2002.

[6] P. Bhattacharya, A. Rosenfeld, and I. Weiss. Point-to-line mappings as hough transforms. *Pattern Recogn. Lett.*, 23(14):1705–1710, 2002.

[7] S. Christy and R. Horaud. Iterative pose computation from line correspondences. *Computer Vision and Image Understanding*, 73(1):137–144, January 1999.

[8] N. Cristianini and J. Shawe-Taylor. *Introduction to Support Vector Machine and other kernel-based learning methods*. Cambridge University Press, 2000.

[9] R. Deriche and O. Faugeras. Tracking line segments. *Image Vision Comput.*, 8(4):261–270, 1990.

[10] M. P. do Carmo. *Riemaniannian Geometry*. Birkauser, 1992.

[11] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[12] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[13] D. F. and G. C. "pose estimation using point and line correspondences". *Journal of Real-Time Imaging, Academic Press*, 5(3):217–232, June 1999.

[14] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, 1993.

[15] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.

[16] O. Faugeras. *Three-dimensional computer vision. A geometric view point*. MIT Press, 2003. ISBN : 0-262-06158-9.

[17] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, pages 23–37, 1995.

[18] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.

[19] P. Hough. Method and means for recognizing complex patterns, Dec 1962.

[20] Q. Ji and Y.Xie. Randomized hough transofrm with error propagation for line and circle detection. *Pattern Anal. Appl.*, 6(1):55–64, 2003.

[21] H. Kalviainen, N. Kiryati, and S. Alaoutinen. Randomized or probabilistic hough transform: unified performance evaluation, 1999.

[22] N. Kiryati, Y. Eldar, and A. M. Bruckstein. A probabilistic hough transform. *Pattern Recogn.*, 24(4):303–316, 1991.

[23] M. Kuss and T. Graepel. The geometry of kernel canonical correlation analysis. (108), May 2003.

[24] S. C. Lee, S. K. Jung, and R. Nevatia. Automatic pose estimation of complex 3d building models. In *WACV*, pages 148–152, 2002.

[25] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):28–37, 1990.

[26] J. Matas, C. Galambos, and J. Kittler. Progressive probabilistic hough transform, 1998.

[27] T. Phong, R. Horaud, A. Yassine, and P. Tao. Object pose from 2d to 3d point and line correspondences. *International Journal of Computer Vision*, 15(3):225–243, July 1995.

[28] M. Pollefeys. Obtaining 3d models with a hand-held camera/3d modeling from images.

[29] A. Rosenfeld. *Picture Processing by Computer*. Academic Press, 1969.

[30] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau. Localization in urban environments: Monocular vision compared to a differential gps sensor. In *CVPR (2)*, pages 114–121, 2005.

[31] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[32] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998.

[33] D. Shaked, O. Yaron, and N. Kiryati. Deriving stopping rules for the probabilistic hough transform by sequential analysis. *Comput. Vis. Image Underst.*, 63(3):512–526, 1996.

[34] M. van Ginkel, C. L. Hendriks, and L. van Vliet. A short introduction to the radon and hough transforms and how they relate to each other. Technical Report QI-2004-01, Quantitative Imaging Group, Delft University of Technology, 2004.

[35] V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[36] L. Xu and E. Oja. Randomized hough transform (rht): basic mechanisms, algorithms and computational complexities. *CVGIP: Image Underst.*, 57(2):131–154, 1993.