Recovering and Characterizing Image Features Using An Efficient Model Based Approach

Thierry Blaszka, Rachid Deriche

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Recovering and Characterizing Image Features Using An Efficient Model Based Approach

Thierry Blaszka, Rachid Deriche

N° 2422

Novembre 1994

PROGRAMME 4

Robotique,

image

et vision

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Recovering and Characterizing Image Features Using An Efficient Model Based Approach

Thierry Blaszka* Rachid Deriche*

Programme 4 — Robotique, image et vision
Projet Robotvis

Rapport de recherche n°2422 — Novembre 1994 — 56 pages

Abstract: Edges, corners and vertices are strong and useful features in computer vision. This paper deals with the development of an efficient model based approach in order to detect and characterize precisely these important features. The key of our approach is first to propose some efficient models associated to each of these features and second to efficiently extract and characterize these features directly from the image. The models associated to each feature include a large number of intrinsic parameters (Grey level intensities, location, orientation of the line segments...) but also an important parameter which is associated to the blurring effect due to the acquisition system. The important problem of the initialization phase in the minimization process is also considered and an original and efficient solution is proposed. In order to test and compare the reliability, the robustness and the efficiency of the different proposed approaches, a large number of experiments involving noisy synthetic data and real images have been carried out. Finally, a comparative study of this approach and the classic corner operators is presented.

Key-words: Low-Level Image Processing, Physics of Image Formation

(Résumé : tsvp)

*{blaszka}{der}@sophia.inria.fr

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Caractérisation de Primitives Image à l'aide d'une Approche par Modèles

Résumé : Les contours, coins et jonctions triples sont des primitives image très utiles en vision par ordinateur. Ce rapport présente le développement d'une méthode à base de modèles pour caractériser ces primitives importantes. L'idée maîtresse de cette approche consiste à définir des modèles correspondant à chacune des primitives et ensuite à les caractériser de manière robuste directement à partir des images. Les modèles définis incluent un grand nombre de paramètres radiométriques et géométriques mais aussi un important paramètre qui est associé à l'effet de lissage introduit par le système d'acquisition. Le problème important de l'initialisation du processus de caractérisation itérative est aussi considéré et une solution originale et efficace est proposée. Afin de tester et de comparer la validité, la robustesse et l'efficacité des différentes approches présentées, un grand nombre d'expérimentations sur des images synthétiques bruitées et des images réelles ont été effectuées. Finalement, une étude comparative de cette approche par rapport à des détecteurs classiques de coins est présentée.

Mots-clé : Vision bas niveau, formation des images

Contents

1 Introduction

Feature extraction is one of the most important areas in computer vision. A great deal of effort has been spent by the computer vision community on this problem and in particular on the problem of edge detection where an extensive literature has been developed from Marr and Hildreth's work [23] to Canny [7] or Deriche's work [9]. Corners and vertices as features are also very important and represent another class of relevant information in computer vision. All these features are used for applications in stereo, automatic visual obstacle avoidance, identification and pose determination of three dimensional objects, displacement vector measuring and an accurate localization of these features is well appreciated and of great interest. Several approaches to the problem of detecting corners and junctions points have been reported in the literature in the last few years.

A first group involves first extracting edges as a chain code, and then searching for points having maxima curvature [1], [11], [25], performing a polygonal approximation on the chains and then search for the line segment intersections [20]. or using an explicit line detector model and exploiting the spatial context to detect junctions [24].

The second group consists of approaches that work directly on grey-level images. These techniques are based either on heuristic techniques [26] or on the measurement of the gradients and of the curvatures of the surface. Among the most popular corner detectors are those proposed by Beaudet [2],Dreschler and Nagel [15], Kitchen and Rosenfeld [21],Zuniga and Haralick [34], Noble [27], Harris and Stephen [19]. Recently on considering geometric properties of isophotes and in particular their invariance under general invertible (non linear) intensity transformation, Ter Haar Romeny [32] proposed to use the gradient of isophote curvature as a good candidate for a T junction detector. Despite the fact that third order derivatives are required in the estimation of this measure, good and promising experimental results are shown. One may note also the interesting approach developed recently by Brunnstrom [6] who considered how junction detection and classification can be performed in an active visual system.

A third group of approaches is emerging, mainly characterized by the use of differential geometry and the development of model based approaches to char-

acterize accurately these features. Hence, using differential geometry, Guiducci [18] characterized corners in an image by three parameters which are the amplitude A of the wedge, its aperture angle θ and a parameter σ that is a measure of the smoothness of the wedge. Analytical expressions for these 3 parameters are then derived to estimate them directly from the grey level image intensity. Deriche and Giraudon [12] considered a corner model and studied analytically its behavior once it has been smoothed using the well-known Gaussian filter. This allowed them to clarify completely the behavior of some well known *cornerness* measure based approaches used to detect these points of interest. In particular, most of the classical approaches presented in the second group have been shown in [12] to detect points that do not correspond to the exact position of the corner. A new scale-space based approach that combine useful properties from the Laplacian of Gaussian and the local maxima of the determinant of the Hessian matrix (Beaudet's measure [2]) has been proposed in order to correct and detect exactly the corner position. An extension of this approach to the problem of trihedral vertex characterization has been developed in [17]. David Beymer [5] analyzed junctions defined as the intersection points of three or more regions in an image. A gradient analysis near junction was performed, by examing a mathematical model for a T junction, providing a framework for a junction detection method that reconstructs junctions from edge maps by growing endpoints. Rohr [30] proposed an interesting approach for the modeling and identification of a certain class of characteristic intensity variations including step edges, corners and more complex junctions types. A general analytical model is proposed and fitted directly to the image intensities to determine the position of grey value structures to sub-pixel accuracy and also additional attributes such as the width of the grey value transitions. The application of this method to real images demonstrates that the identified model functions agree fairly well with the original grey-value structures, but it has been found that this approach is computationally very expensive. Also one can notice that some important questions have not been addressed. No attempt has been made to propose an efficient way to deal with the problem of the automatically choice of the window size for the fit and the problem of the initialization phase in the minimization process has not been addressed in depth.

This paper presents an approach which is a natural extension of the work started in [12], [17] and [30]. We propose to locate and characterize features as edges, corners and vertices directly from the image by searching the parameters of the model that best approximate the observed grey level image intensities. The model associated to each feature include a large number of intrinsic parameters (grey level intensities, position and orientation of the feature...) but also an important parameter which is related to the blurring effect due to the acquisition system. Berzins [4] first employed the idea of a model to analyze the accuracy of Laplacian edge detector. This idea has been proven to be powerful in better understanding of the structure behavior in a scale space approach and has been adopted later by Bergholm [3], who used an explicit model of grey value corner to evaluate the displacement of the corner points depending on the σ parameter of the used Gaussian filter. De Michelli *etal* [8] have used also this idea to present a comparative study between zero-crossing and gradient approaches on corner and trihedral vertex.

In this paper, the focus is on the problem of determining efficiently the model parameters of the considered feature directly from the grey level image intensities. Due to the large amount of time required by the direct approach that assumes the blur of the imaging acquisition system to be describable by a 2D Gaussian filter [30], some alternative and efficient solutions are considered and developed in detail in this paper. The first idea is to replace the Gaussian filter by an exponential filter derived from optimality considerations in edge detection [10] and which can easily be made very close to the Gaussian. Using this filter allows us to derive a close form for the different integrals required in the approximation process and thus enables to avoid the numerical integrations which are very time consuming and make the approach developed by Rohr [30] not practical. The important problem of the initialization phase in the minimization process is also considered in this paper and an original and efficient solution is proposed and analyzed also in detail. Finally, an extensive experimental work on noisy synthetic data and real images has been carried out to evaluate the robustness and the accuracy of the different methods. Some experiments are also presented to demonstrate the advantages of the model based methods in a real application: the 3D-reconstruction from two images.

2 Features Models Using a 2D Gaussian Blurring Filter

2.1 Notations and definitions

First, some functions that will be largely used in the rest of the paper, are introduced here. Let $g(x)$ denote the zero mean Gaussian filter:

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{1}$$

The two-dimensional Gaussian filter G can then be expressed as:

$$G(x, y) = g(x)g(y) \tag{2}$$

Following Berzins [4], the unit length of the considered coordinate system is equal to the scale factor σ of the filter. In order to convert the results into a more general coordinate system (X, Y), the following transformation is used:

$$\begin{cases} x = \frac{X}{\sigma} \\ \\ y = \frac{Y}{\sigma} \end{cases} \tag{3}$$

Let Φ denote the error function given by:

$$\Phi(x) = \int_{t=-\infty}^{x} g(t)dt \tag{4}$$

Let U define the unit step function:

$$U(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The response of the 2D filter F for a 2D input function I can be computed by evaluating the following convolution integral:

$$S(x, y) = \int_{\alpha=-\infty}^{+\infty} \int_{\beta=-\infty}^{+\infty} F(\alpha, \beta) I(x - \alpha, y - \beta) d\beta d\alpha \tag{6}$$

2.2 Edge Model

A convenient representation for a straight line is as follows:

$$-\ x\ \sin(\theta)\ +\ y\ \cos(\theta)\ -\ \rho\ =\ 0 \tag{7}$$

where $[-\sin(\theta), \cos(\theta)]^T$ defines the unit vector perpendicular to the straight line and ρ is the distance of the origin to the closest point on the line. One may note that $\forall(x_0, y_0)$ the sign of this equation determines to which half plane, belongs the point (x_0, y_0).

An ideal edge supported by such line is then modeled as follows:

$$I_{\text{edge}}(x, y) = U(-x\sin(\theta) + y\cos(\theta) - \rho) \tag{8}$$

However, such an ideal model does not correspond generally to edges that are present in images. It is well known that any image acquisition system introduces a blur effect. Taking into account this problem, and considering that the Gaussian function as one of the most advocated model to represent the point spread function caused by defocus [28], leads to a more reasonable model for real edges in images. This model is obtained by convolving the ideal edge model, given by (8) with the 2D Gaussian filter given by (2). Developing and simplifying the convolution operation given by (6) with the 2D Gaussian filter, yields the following result:

$$E_1(x, y) = \Phi(-x\sin(\theta) + y\cos(\theta) - \rho) \tag{9}$$

where $\Phi(.)$ is the error function given by (4).

This expression depends on two parameters θ and ρ. Taking into account the grey level intensities leads to add two other parameters A and B corresponding to each half plane defined by the line supporting the edge. Finally, adding the blur effect described by the parameter σ leads to a total of 5 parameters to express completely an edge model:

$$E_g(x, y, \theta, \rho, \sigma, A, B) = (A - B)\Phi\left(-\frac{x}{\sigma}\sin(\theta) + \frac{y}{\sigma}\cos(\theta) - \frac{\rho}{\sigma}\right) + B \tag{10}$$

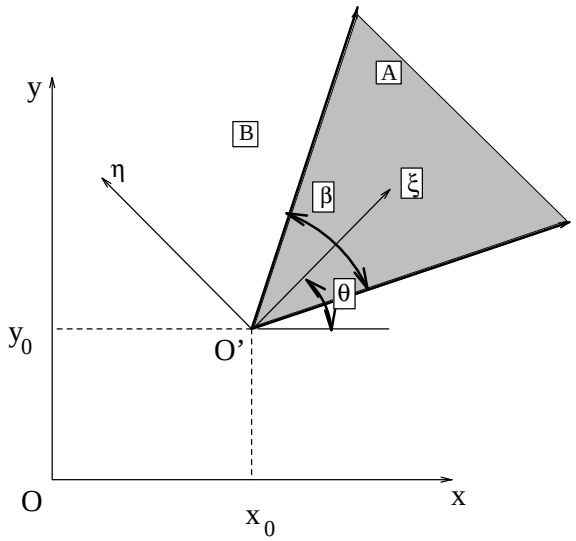

Figure 1: A Corner Model

2.3 Corner Model

An ideal corner with one edge along the ξ-axis and an aperture angle $\frac{\beta}{2}$ (see figure 1) can be modeled by the following 2D step function:

$$I_{\text{corner}}(\xi, \eta) = U(\eta)U(m\xi - \eta) \tag{11}$$

where $m = tan(\frac{\beta}{2})$, and $\beta \in]0, \pi[$.

Convolving this 2D step function with the 2D Gaussian filter given by (2), yields the following filtered image $I_g(\xi, \eta, \beta)$:

$$I_g(\xi, \eta, \beta) = \Phi(\xi)\Phi(\eta) - \int_{\alpha=-\infty}^{\xi} g(\alpha)\Phi(\eta + m(\alpha - \xi))d\alpha \tag{12}$$

This equation describes a Gaussian filtered corner localized in the origin point O' with an aperture angle $\frac{\beta}{2}$ and with one side on the ξ-axis.

A more general model for an ideal corner may be derived first by adding to the model $I_g(\xi, \eta, \beta)$, its reflection with the ξ-axis $I_g(\xi, -\eta, \beta)$:

$$C_1(\xi, \eta, \beta) = I_g(\xi, \eta, \beta) + I_g(\xi, -\eta, \beta) \tag{13}$$

Second, the local coordinate system into which this model is defined is so that its origin is $O' = (x_0, y_0)$ in the global coordinate system (O, x, y), and its ξ-axis form an angle θ with the global x-axis (see figure 1). Let (x, y) denote the global coordinates of a point and (ξ, η) its local coordinates. The transformation of global coordinates onto local ones is given by:

$$\begin{pmatrix} x \\ y \end{pmatrix} \overset{T_{O'O}}{\longrightarrow} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \overset{R_{O',-\theta}}{\longrightarrow} \begin{pmatrix} \xi \\ \eta \end{pmatrix} = \begin{pmatrix} x' \cos(\theta) & + & y' \sin(\theta) \\ -x' \sin(\theta) & + & y' \cos(\theta) \end{pmatrix} \quad (14)$$

Taking into account the change of the coordinate system, the grey level intensities A and B within each region defined by the corner, and adding the blur effect described by the parameter σ leads to a total of 7 parameters to express completely a corner model:

$$C_g(x, y, \vec{p_c}) = (A - B)C_1\left(\frac{\xi(x, y, x_0, y_0, \theta)}{\sigma}, \frac{\eta(x, y, x_0, y_0, \theta)}{\sigma}, \beta\right) + B \quad (15)$$

where $\vec{p_c} = (x_0, y_0, \theta, \beta, \sigma, A, B)$ is the parameter vector characterizing a corner as illustrated in figure 1. The different parameters are:

- x_0, y_0: The position of the corner.
- θ: The orientation of the corner symmetrical axis.
- β: The aperture angle.
- σ: The parameter that characterizes the amount of blur.
- A and B: The grey level intensities inside and outside the corner.

One can note that this model is defined for $\beta \in]0, \pi[$, but for a corner of aperture greater than π, it is clear that it will be represented with the parameters $\theta' = \theta + \pi$, $\beta' = 2\pi - \beta$, $A' = B$ and $B' = A$.

2.4 Vertex Model

Vertices are defined as a superposition of corner models (see figure 2). For example, triple junctions defined by 3 adjacent regions (T,Y or ARROW-corners) can easily be described using a superposition of two corner models. However, some constraints have to be taken into account: the corner positions (x_0, y_0)

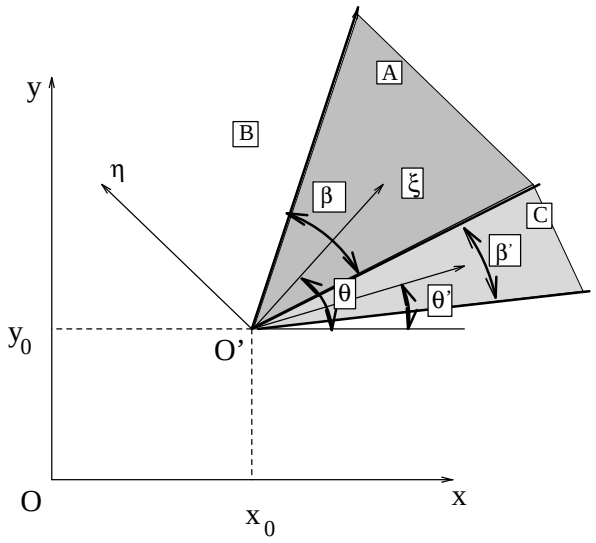

Figure 2: A Vertex Model

and the blur parameter are the same, and both corners must be adjacents. This leads to the following relation between θ and θ':

$$\theta' = \theta - \frac{\beta + \beta'}{2} \tag{16}$$

Consequently, a triple junction, as illustrated in figure 2, will be defined by the the following vector of 9 parameters:

- x_0, y_0: The position of the triple junction.
- θ: The symmetrical axis orientation of the first corner.
- β: The aperture angle of the first corner.
- β': The aperture angle of the second corner.
- σ: The parameter that characterizes the amount of blur.
- A and C: The grey level intensities inside the two corners.
- B: The grey level intensity outside the two corners.

Using the previous corner models, it is very easy to derive the following expression for a triple junction model:

$$J_g(x, y, \vec{a}) = C_g(x, y, \vec{c_f}) + C_g(x, y, \vec{c_s}) + B \tag{17}$$

where $\vec{a} = (x_0, y_0, \theta, \beta, \beta', \sigma, A, B, C)$ denotes the parameter vector characterizing the triple junction, $\vec{c_f} = (x_0, y_0, \theta, \beta, \sigma, A - B, 0)$ the parameter vector of the first corner included in the junction and $\vec{c_s} = (x_0, y_0, \theta', \beta', \sigma, C - B, 0)$ the parameter vector of the second corner of the junction.

3 On Approximating Models to Image Data

Having the models that describe the features of interest, namely edges, corners and triple junctions, the interesting thing is now to apply these models to real data. Given a window centered in an initial estimate of the localization of a feature and given the type of feature lying in this area, an iterative method can be used to characterize this feature by approximating the set of data within this area using this model. But, this approach requires a first vector of parameters to initiate the process.

Even if the method has been proven to be robust to the case where the initialization is far from the solution (see the experimental part), starting with a parameter vector far from the solution leads to a convergence time that may be too long and then inefficient in term of CPU time. To tackle this problem, a fast method has been developed to start from initial conditions that may be far and to produce closer initial conditions by providing a rough solution to the minimization problem. This method is denoted the *variance descent approach.*

The next subsection describes the iterative algorithm used to fit the model to the real data and the second subsection presents the *variance descent approach* used to supply the initial set of parameters to the iterative algorithm.

3.1 The approximation procedure

The problem is thus to try to find the best parameters that characterize the model and make it to fit as well as possible the real data within the working

area. The adequation of the fitting of the selected feature to the data is defined by the energy term:

$$F(X) = \sum_{i=1}^{m} (f_i(X))^2 \tag{18}$$

where $X = (X_1, X_2, \ldots, X_n)^T$ is the parameter vector of the feature and its n components to be estimated correspond to the 5, 7 or 9 unknown parameters that characterize the edge, corner or vertex models respectively. The number m denotes the number of points in the working area. The functions $f_i(X)$ are often referred to as residuals and are defined by:

$$f_i(X) = I(x_i, y_i) - M(x_i, y_i, X) \tag{19}$$

where, $I(x_i, y_i)$ denotes the grey-level intensity of the pixel (x_i, y_i) in the image and M is the selected feature model. The goal of the fitting is to minimize the energy term $F(X)$ defined in 18 and because of its formulation, a nonlinear least-square minimization algorithm is required.

3.1.1 Nonlinear least-square minimization

The energy term (18) can be approximated up to second order, near its minimum, by:

$$F(X) \approx \gamma + \nabla F(X).X + \frac{1}{2} X^T \ \boldsymbol{H} \ X \tag{20}$$

where $\boldsymbol{H}$ denotes the Hessian matrix. From the current set of parameters X, a first approach to minimize the energy consists in defining a new set of parameters X_{next} by:

$$X_{\text{next}} = X + \boldsymbol{H}^{-1}.[-\nabla F(X)] \tag{21}$$

which is denoted as the *Inverse Hessian method*. But if the approximation given by (20) is poor, a best estimate of the next parameters is given by:

$$X_{\text{next}} = X - \text{constant} \times \nabla F(X) \tag{22}$$

which is the classic *steepest gradient descent*.

3.1.2 Evaluation of Gradient and Hessian

The gradient of the energy (18) is defined by:

$$\frac{\partial F}{\partial X_k} = -2\sum_{i=1}^{m} f_i(X).\frac{\partial f_i(X)}{\partial X_k} \qquad k = 1 \dots n \tag{23}$$

and the corresponding Hessian matrix is defined by:

$$\frac{\partial^2 F}{\partial X_k \partial X_l} = 2\sum_{i=1}^{m}\left[\frac{\partial f_i(X)}{\partial X_k}\frac{\partial f_i(X)}{\partial X_l} - f_i(X).\frac{\partial^2 f_i(X)}{\partial X_k \partial X_l}\right] \qquad k = 1 \dots n, l = 1 \dots n \tag{24}$$

To simplify, it's conventional to define:

$$\beta_k \equiv -\frac{1}{2}\frac{\partial F}{\partial X_k} \qquad \alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2 F}{\partial X_k \partial X_l} \tag{25}$$

With these notations, and as $[\alpha] = \frac{1}{2}\boldsymbol{H}$, the expression of the *Inverse Hessian method* given in (21) becomes:

$$\sum_{l=1}^{n} \alpha_{kl}\delta X_l = \beta_k \tag{26}$$

where $\delta X = X_{\text{next}} - X$. Following the same scheme the *steepest gradient descent* becomes:

$$\delta X_l = \text{constant} \times \beta_l \tag{27}$$

One can note that the components of the Hessian matrix depends on the first partial derivatives and on the second partial derivatives; the usual methods ignore the second derivatives and define:

$$\alpha_{kl} = \sum_{i=1}^{m} \frac{\partial f_i(X)}{\partial X_k}\frac{\partial f_i(X)}{\partial X_l} \tag{28}$$

In the following, any reference to α_{kl}, will correspond to this last definition (for more details see [29]).

3.1.3 The Levenberg-Marquardt algorithm

As the condition at the F minimum ($\beta_k = 0$ for all k), is independent of how $[\alpha]$ is defined; the main idea of this algorithm is to varying smoothly between the extremes of the *Inverse Hessian method* and the *steepest gradient descent* by modifying the definition of $[\alpha]$. For a specific choice of the 'constant' in the *steepest gradient descent*, the equation (22) becomes:

$$\delta X_l = \frac{1}{\lambda \alpha_{ll}} \beta_l \tag{29}$$

The Levenberg-Marquardt insight is that equations (29) and (26) can be combined if a new matrix α' is defined as:

$$\begin{array}{lcll} \alpha'_{jj} & \equiv & \alpha_{jj}(1+\lambda) & \\ \alpha'_{jk} & \equiv & \alpha_{jk} & (j \neq k) \end{array} \tag{30}$$

Then the two methods given by equations (29) and (26) are mixed by the use of the formulation:

$$\sum_{i=1}^{n} \alpha'_{kl} \delta X_l = \beta_k \tag{31}$$

When λ is very large, the matrix α' is forced to be diagonally dominant, so equation (31) goes over to be identical to (29). On the other hand, as λ approaches zero, equation goes over to (26). Given an initial estimate of the parameter vector X and a precision (tol) required on the energy, the Levenberg-Marquardt algorithm is defined by:

1. Pick a modest value for λ, say $\lambda = 0.001$.
2. Solve the linear equations (31) for δX
3. if $|F(X) - F(X + \delta X)| <$ tol then end
4. if $F(X + \delta X) \geq F(X)$, increase λ by a factor of 10 (or any other substantial factor) and go to step 2.
5. if $F(X + \delta X) < F(X)$, decrease λ by a factor of 10 (or any other substantial factor), update the solution $X \leftarrow X + \delta X$ and go to step 2.

This Levenberg-Marquardt method works very well in practice and has become the standard of nonlinear least-squares routines (for more details see [29]).

3.2 A variance descent approach to initialize the approximation

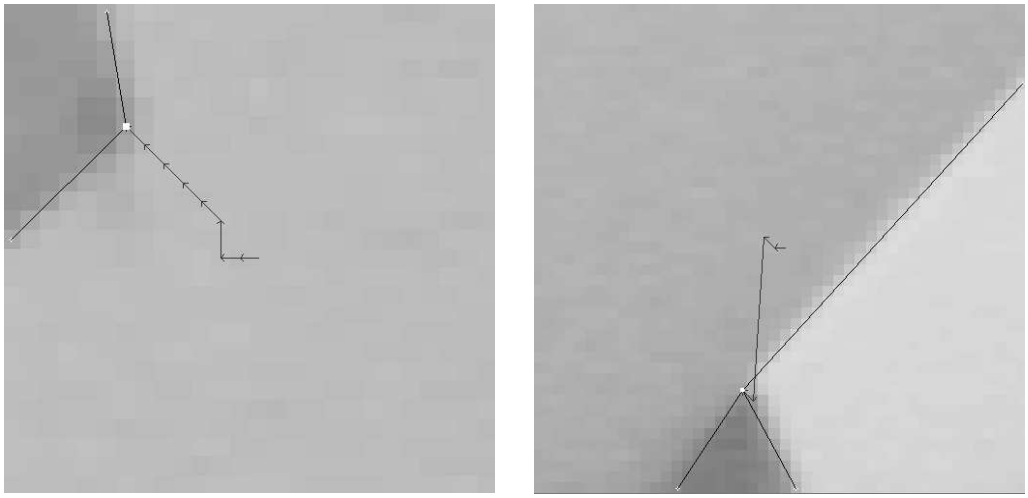

Figure 3: Convergence of the variance descent approach for an angle (left) and for a triple junction (right).

To detect and characterize image features such as edges, corners or vertices, the solution proposed in the previous subsection was to approximate the image data using the analytical models. Because of the non-linearity of the criterion to be minimized, an iterative solution has to be adopted which requires an initial estimate of the different parameters to be refine. This subsection proposes a method to get such initial conditions while reducing the number of search iterations. Using the measure proposed by Beaudet, i.e the determinant of the Hessian matrix [2], (or any corner measure) yields an initial estimation of the corner position. To refine this corner position and get an estimate for the other parameters that will be used by the approximation procedure, the following approach is used:

A roughly estimated position of a corner is given interactively by the user or by Beaudet's measure. A window of size $(2*L+1)*(2*M+1)$ pixels is centered on this corner position. First, the $4*(M+L)$ pixels located in the frontier of this window area are considered as a 1D signal and edge points are extracted on this

signal using the 1D detector proposed in [10]. Next, two points are selected in the case of an edge or a corner and three in the case of a vertex. Let's consider the corner case to describe this approach. The two regions corresponding to the inside and outside of the corner are considered. These regions are formed by the two lines joining the center of the window to the two detected edge points in the frontier of the window. Then, for each region an energy term assuming planar intensity regions is calculated. This energy term is related to the variance in grey level intensities within the considered areas:

$$\epsilon^2 = \sum_{(i,j) \in Region} (I(i,j) - \bar{I})^2 \tag{32}$$

where $\bar{I}$ denotes the average of the intensity level within the considered region. Moreover, one can note that ϵ^2 is equal to zero for planar intensity region. The energy term corresponding to a given point is then defined as the sum of the variances in both regions it defines. In order to refine the corner position, the following approach defined as the *variance descent approach* is used.

First, the energy term corresponding to the current position of the point and the energy terms corresponding to its 8 neighbors are calculated. The refinement of the current corner position is done by selecting the one with the minimum energy term as the next position of the corner. The refinement is repeated until the energy term stops to decrease. Moreover, in the next iterations, the energy term is evaluated just for those points where it has not been already calculated in the previous iteration i.e. just for 3 points if the displacement direction is horizontal or vertical and 5 points if the displacement direction is diagonal. When this algorithm has converged, the energy term would have reached the global minima, because the initial corner position was assumed to be not so far from the solution. This position, the orientation of the lines joining this point to the edges detected in the frontiers and the average intensity on each region, constitute the initial conditions of the model based approach. An histogram of the grey level intensities within the window is also calculated. This information, combined with the number of edges extracted along the frontiers, are used to better discriminate between the corner and vertex cases.

This approach has been widely tested for different models and size windows and an interactive version has been developed with the possibility for the user

to choose the window and its size interactively. The center point of the window is then considered as the first point to analyze and the process iterates. This approach, which works also for vertices, has been proven to be be very fast and enables to get efficiently close initial conditions so that the model based approach will not fail in local minima. The two examples shown in figure 3 illustrate the different iterations for the corner and vertex models. In each figure, the trajectory with the arrows describes the convergence path from the center of the window to the final solution. Then, this final solution could be refined to a sub-pixel accuracy using the model based approach as described in this paper.

4 Features Models Using a 2D Exponential Blurring Filter

Assuming the blur of the imaging acquisition system to be describable by a 2D Gaussian filter, as it has been done in the previous section, leads to a large amount of CPU time that renders the approach inefficient. This is mainly due to the large number of numerical integrations performed during the minimization process. Therefore, the process has been found inefficient (see the experimental part and in particular the parameter related to the CPU time) and a strong need to make it more efficient in term of CPU time, clearly appeared. Two solutions have been considered, analyzed and implemented to tackle efficiently this problem: The first solution described within this section proposes to replace the Gaussian filter by an exponential filter that allows us to deal with a close form solution in the expression of the different models. The second solution described in the next section proposes more simplified models. These ideas have been proven to make the approximation process very fast and rendered this model based approach much more attractive.

4.1 Notations and definitions

Let $h(x)$ denotes the smoothing operator derived using some optimality considerations on edge detection [10]:

$$h(x) = \frac{1}{4}\ (|x| + 1)\ e^{-|x|} \tag{33}$$

Like in the Gaussian case, its 2D separable version is defined as:

$$H(x, y) = h(x)h(y) \tag{34}$$

In this case, the unit length of the coordinate system is equal to the inverse of the scale factor α of the filter. In order to convert the results into a more general coordinate system (X, Y), the following transformation is used:

$$\begin{cases} x = \alpha X \\ y = \alpha Y \end{cases} \tag{35}$$

It is worthwhile to note that a simple way to make this operator close to the Gaussian is to select the parameters α and σ in such a way that both smoothing filters have the same total energy. This yields the following relation between α and σ [10]:

$$\alpha\sigma = \frac{5}{2\sqrt{\pi}} \tag{36}$$

Let $sign(x)$ be the function defined as follows:

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{37}$$

Let $\Phi_e(x)$ be the error function associated to the exponential filter, and defined as follows:

$$\begin{aligned} \Phi_e(x) &= \int_{t=-\infty}^{x} \frac{1}{4}(|t| + 1)e^{-|t|}\ dt \\ &= \frac{1}{2} + \text{sign(x)}\left(\frac{1}{2} - \frac{1}{4}\ e^{|x|}(2 + |x|)\right) \end{aligned} \tag{38}$$

4.2 Features Models

Replacing the Gaussian filter in (9) by the exponential filter given in (33) leads to the following close form solution for the edge model:

$$E_2(x,y) = \Phi_e(-x\sin(\theta) + y\cos(\theta) - \rho) \tag{39}$$

This expression depends on two parameters θ and ρ. Taking into account the blur effect, introduced by the parameter α, and also the grey level intensities A and B, corresponding to each half plane defined by the line supporting the edge, leads to the following five parameters edge model:

$$E_e(x,y,\theta,\rho,\alpha,A,B) = (A-B)\Phi_e(-\alpha x\sin(\theta) + \alpha y\cos(\theta) - \alpha\rho) + B \tag{40}$$

This expression is much more easier and less time consuming to evaluate than the previous one derived using the Gaussian filter and given by (10). No numerical integration is required and this renders this approach much more computationally attractive.

This idea of replacing the Gaussian filter by the exponential filter can also be applied to the corner and triple junction models. So, using the 2D exponential filter given in (34) and the expression of the ideal corner given by (11) in the convolution operation given by (6) yields to a close form solution, derived using *Maple*. Then the expression of the complete corner of an aperture $\beta \neq \frac{\pi}{2}$ is:

$$C_2(x,y,\beta) = \frac{1}{2} + \left(m^2-1\right)^{-3} Z_1 + \text{sign}(xm - |y|)\left(\frac{1}{2} + \left(m^2-1\right)^{-3} Z_2\right) \tag{41}$$

where:

$$\begin{aligned}
Z_1 &= \frac{1}{4}U(x)\left(2 - 6\,m^2 + (1-m^2)(xm + |y|)\right)\, e^{-(xm+|y|)} - \frac{m}{8} Z_3\, e^{-(|x|+|y|)} \\
Z_2 &= \frac{1}{4}U(x)\left(2 - 6\,m^2 + (1-m^2)||y| - xm|\right)\, e^{-|xm-|y||} \\
&\quad + \frac{m^3}{4}\left(2\,m\,(3-m^2) + (1-m^2)||y| - xm|\right)\, e^{-\frac{|xm-|y||}{m}} \\
Z_3 &= |xy|(m^2-1)^2 + 3 + 3m^4 - 14\,m^2 + (m^2-1)(|x|\,(3\,m^2-1) + |y|\,(m^2-3)) \\
m &= \tan\left(\frac{\beta}{2}\right) \quad \text{with} \quad \beta \in]0, \frac{\pi}{2}[\cup]\frac{\pi}{2}, \pi[
\end{aligned}$$

And for a right corner (i.e. $\beta = \frac{\pi}{2}$), the expression becomes:

$$\begin{aligned} C_2(x,y) &= U(x-|y|) \\ &+ \frac{1}{96}\Big((|y|-x)^3 - \mathrm{sign}(x-|y|)(9(x-|y|)^2+48) + 33(|y|-x)\Big)\, e^{-||y|-x|} \\ &- \frac{1}{96}\Big(3x^2|y| + |x|^3 + 9(x^2+|xy|) + 24|x| + 9|y| + 24\Big)\, e^{-(|x|+|y|)} \end{aligned} \tag{42}$$

The use of the transformation given by (14), yields a more general model described in a global coordinate system; and taking into account the blur effect described by the parameter α (using the transformation given by (35)), and the grey level intensities A and B within each region defined by the corner leads to a total of 7 parameters, like in the Gaussian case, to express completely the corner model:

$$C_h(x,y,x_0,y_0,\theta,\beta,\alpha,A,B) = (A-B)C_2(x',y',\beta) + B \tag{43}$$

with:

$$\begin{aligned} x' &= \alpha(x-x_0)\cos(\theta) + \alpha(y-y_0)\sin(\theta) \\ y' &= -\alpha(x-x_0)\sin(\theta) + \alpha(y-y_0)\cos(\theta) \end{aligned} \tag{44}$$

The vertex model, like in the Gaussian case, is defined as the superposition of two corner models. However, some constraints have to be taken into account: the corner positions (x_0, y_0) and the blur parameter α are the same, and both corners must be adjacents (constraint given by equation 16). Then it is very easy to derive the following close form expression for the triple junction model:

$$J_h(x,y,\vec{a}) = C_h(x,y,\vec{c_f}) + C_h(x,y,\vec{c_s}) + B \tag{45}$$

with:

$$\begin{aligned} \vec{a} &= (x_0, y_0, \theta, \beta, \beta', \alpha, A, B, C) \\ \vec{c_f} &= (x_0, y_0, \theta, \beta, \alpha, A-B, 0) \\ \vec{c_s} &= (x_0, y_0, \theta', \beta', \alpha, C-B, 0) \end{aligned}$$

where $\vec{a}$ denotes the vector parameter characterizing the triple junction, and $\vec{c_f}$ and $\vec{c_s}$ denote the parameter vector of the first and second corner, respectively, included in the junction.

These expressions look complicated but they are much more easier and faster to evaluate than the expressions involving the 2D Gaussian. The main reason is that no numerical integration is required.

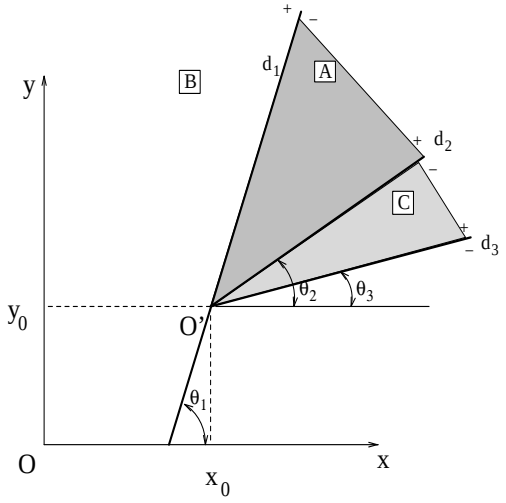

Figure 4: Simplified models

5 Simplified Models

In this section, another simplification is considered in order to make the approximation process much more computationally efficient. A simplified version for corner and triple junction models is designed by assuming that these primitives are just formed by the intersections of a given number of straight lines.

5.1 Separable Models

The ideal half-corner expressed in equation (11) is just the product of two term involving the U function. The first term corresponds to the application of the U function to the equation of the ξ-axis, and the second one is the application of the U function to the equation of the line with an orientation of $\frac{\beta}{2}$ with respect to the ξ-axis. If the two lines of a corner are considered in a global coordinate system (O, x, y) and not in the local one (O', ξ, η) as in equation (11), the expression of an ideal corner model becomes:

$$I(x,y) = U(-f_{d_1}(x, y, x_0, y_0, \theta_1))\ U(f_{d_2}(x, y, x_0, y_0, \theta_2)) \tag{46}$$

where f_{d_1} and f_{d_2} denote the two following straight lines intersecting in (x_0, y_0) (see figure 4):

$$f_{d_i}(x, y, x_0, y_0, \theta_i) = -(x - x_0)\ \sin(\theta_i)\ +\ (y - y_0)\ \cos(\theta_i) = 0 \quad i = 1, 2 \quad (47)$$

Considering the bluring effect as normal to each lines yields:

$$\tilde{C}_s(x, y, x_0, y_0, \theta_1, \theta_2, \delta) = \Phi_s(-\frac{1}{\delta} f_{d_1}(x, y, x_0, y_0, \theta_1))\Phi_s(\frac{1}{\delta} f_{d_2}(x, y, x_0, y_0, \theta_2)) \quad (48)$$

where the subscript s refers to the Gaussian or Exponential smoothing operator and the blurring effect δ corresponds to the parameter σ or $\frac{1}{\alpha}$, depending on the smoothing operator used. The complete expression that takes into account the grey level intensities and the amount of blur δ is then given by:

$$C'_s(x, y, x_0, y_0, \theta_1, \theta_2, A, B, \delta) = (A - B)\tilde{C}_s(x, y, x_0, y_0, \theta_1, \theta_2, \delta) + B \quad (49)$$

One can note that the parameters are a little different than in the section (2.3), but they could be easily related to by:

$$\begin{array}{rcl} \theta_1 & = & \theta + \frac{\beta}{2} \\ \theta_2 & = & \theta - \frac{\beta}{2} \end{array} \quad (50)$$

Having developed the corner model, it is very easy to deduce the vertex model using the same approach as described in the previous section. This yields the following expression:

$$J'_s(x, y, \vec{a}) = C'_s(x, y, \vec{c_f}) + C'_s(x, y, \vec{c_s}) + B \quad (51)$$

with

$$\begin{array}{rcl} \vec{a} & = & (x_0, y_0, \theta_1, \theta_2, \theta_3, \delta, A, B, C) \\ \vec{c_f} & = & (x_0, y_0, \theta_1, \theta_2, A - B, 0, \delta) \\ \vec{c_s} & = & (x_0, y_0, \theta_2, \theta_3, C - B, 0, \delta) \end{array}$$

where $\vec{a}$ denotes the vector parameter characterizing the triple junction and $\vec{c_f}$ and $\vec{c_s}$ denote the parameter vectors of the first and second corner of the junction (see figure 4). The relations between the parameters of the section (2.4) and the previous ones are the same as those of equation (50) for θ_1 and θ_2, and the relation for θ_3 is given by:

$$\theta_3 = \theta_2 - \beta' \quad (52)$$

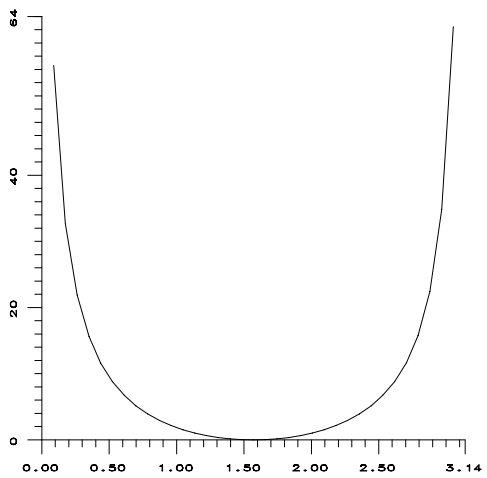

Figure 5: Curve representing the error function given by (56) for the Gaussian filter.

5.2 Accuracy of the approximation

An interesting question related to the approximation process proposed in the previous subsection is to ask about the errors introduced using such simplification. Using a 2D smoothing Gaussian filter leads to the following expression for the exact 2D model:

$$\begin{array}{l} C_g(x,y,x_0,y_0,\theta,\beta,A,B,\sigma) = \\ \int_{\gamma=-\infty}^{\infty}\int_{\beta=-\infty}^{\infty} U(-f_{d_1}(x,y)+\gamma')\ U(f_{d_2}(x,y)-\beta')\ g(\gamma)\ g(\beta)d\beta\ d\gamma \end{array} \tag{53}$$

where:

$$\begin{array}{l} \left\{ \begin{array}{rcl} \gamma' & = & -\gamma\ \sin(\theta_1)+\beta\ \cos(\theta_1) \\ \beta' & = & -\gamma\ \sin(\theta_2)+\beta\ \cos(\theta_2) \end{array} \right. \\ f_{d_i}(x,y) = f_{d_i}(x,y,x_0,y_0,\theta_i) \end{array} \tag{54}$$

Noting that,

$$U(x) = 1 - U(-x)$$

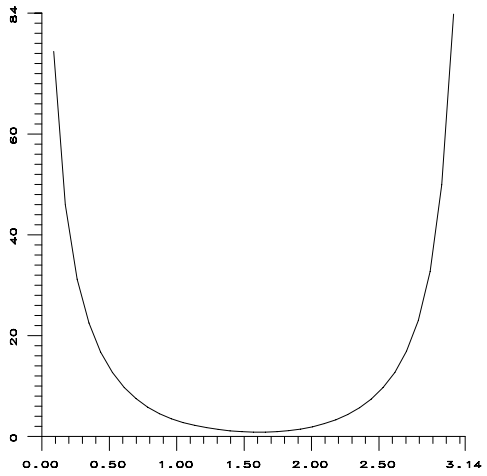

Figure 6: Curve representing the error function given by (56) for the exponential filter.

Operating some change of variables, developing, and simplifying yields the following result:

$$C_g(x, y, x_0, y_0, \theta, \beta, A, B, \sigma) = I_1 - I_2 \tag{55}$$

where:

$$I_1 = \frac{1}{2\pi \sin(\theta_2 - \theta_1)} \int_{\gamma'=-\infty}^{\infty} \int_{\beta'=-\infty}^{f_{d_2}(x,y)} e^{-\frac{\gamma'^2 + \beta'^2 - 2\gamma'\beta' \cos(\theta_2 - \theta_1)}{2 \sin^2(\theta_2 - \theta_1)}} d\beta' \; d\gamma'$$

$$I_2 = \frac{1}{2\pi \sin(\theta_2 - \theta_1)} \int_{\gamma'=-\infty}^{f_{d_1}(x,y)} \int_{\beta'=-\infty}^{f_{d_2}(x,y)} e^{-\frac{\gamma'^2 + \beta'^2 - 2\gamma'\beta' \cos(\theta_2 - \theta_1)}{2 \sin^2(\theta_2 - \theta_1)}} d\beta' \; d\gamma'$$

One may note that for $\theta_2 - \theta_1 = \frac{\pi}{2}$, we have:

$$\begin{aligned} I_1 &= \Phi_g(f_{d_2}(x, y)) \\ I_2 &= \Phi_g(f_{d_1}(x, y))\Phi_g(f_{d_2}(x, y)) \end{aligned}$$

which implies that:

$$\begin{aligned} C_g(x,y,x_0,y_0,\theta,\beta,A,B,\sigma) &= \Phi_g(f_{d_2}(x,y)) - \Phi_g(f_{d_1}(x,y))\Phi_g(f_{d_2}(x,y)) \\ &= \Phi_g(-f_{d_1}(x,y))\Phi_g(f_{d_2}(x,y)) \end{aligned}$$

This shows that for the case of a right angle, the general expression reduces to the simplification proposed in the previous subsection, which presents the interest to be much more easier and faster to evaluate. In order to numerically analyze the behavior of the errors between the expression (55) and the proposed simplification, we consider angles presenting an horizontal symmetry axis (i.e. $\theta = 0$). The summit is located at the origin (i.e. $x_0 = 0, y_0 = 0$), the parameter σ is set to unity and the parameters A and B are set to 1 and 0 respectively. Let:

$$\theta_1 = -\theta_2 = \frac{\beta}{2}$$

and the following error criterion:

$$\Sigma^2 = \sum_{i=0}^{I}\sum_{j=0}^{J}(C_g(x_i,y_j,x_0,y_0,\theta,\beta,A,B,\sigma) - C_s(x_i,y_j,x_0,y_0,\theta_1,\theta_2,A,B,\delta))^2 \quad (56)$$

where:

$$I = N(M_2 - M_1),\ J = N(N_2 - N_1),\ x_i = M_1 + \frac{i}{N},\ y_i = N_1 + \frac{j}{N} \quad (57)$$

The curve illustrating the amount of the error function of the angle in the case of Gaussian smoothing is shown in figure 5.

Considering the exponential filter (with the parameter $\alpha = 1.5$) and working in an analogous way, leads to exactly the same conclusions. For a right angle, the 2D case reduces to the simplified proposition and the errors introduced by the 1D simplification increase if the angle of aperture for the corner moves away from the right corner. The curve illustrating the amount of the error function of the angle in the case of exponential smoothing is shown in figure 6. This result will be confirmed in the experimental part where two different corners will be considered.

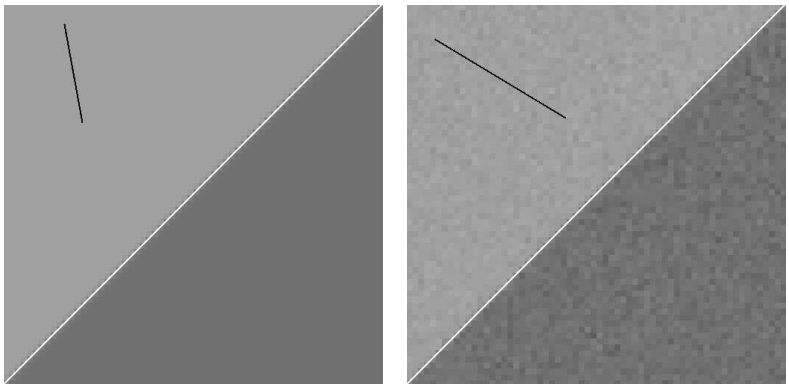

Figure 7: Convergence on edges (synthetic images). Left: without noise; Right: the same edge but with a Gaussian white noise with standard deviation $\sigma = 5$ added.

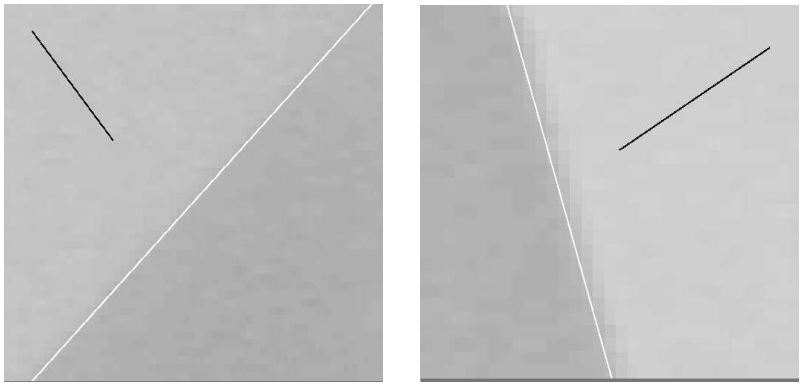

Figure 8: Convergence on edges (real images). For the left image see table 2 and for the right image see table 3.

6 Experimental results

6.1 Models Comparison

For the purposes of experimentation, different types of features were considered. Noisy synthetic images of grey-value edges, corners and triple junctions have been created, a large number of real images were selected and an exten-

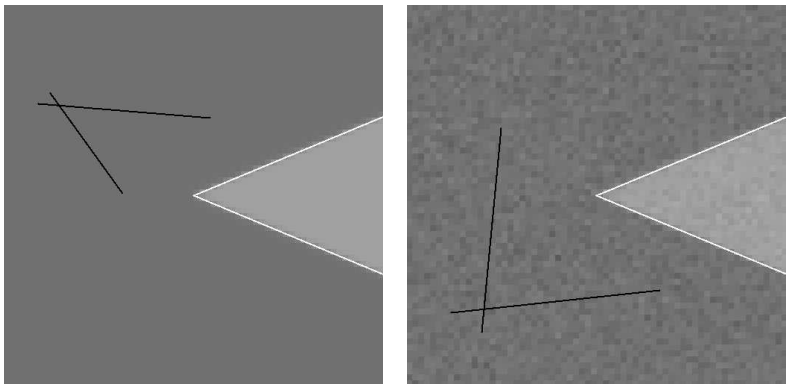

Figure 9: Convergence on corners (synthetic images). Left: a synthetic corner without noise. Right: The same corner with a Gaussian white noise of standard deviation $\sigma = 5$ added.

sive experimental work has been carried out in order to test the robustness, the accuracy and evaluate the CPU computational time of the differents approaches presented here. In this section, just a summary of these experiments is given, and only the most important results are presented. The results on noisy synthetic images are presented into the figures 7, 9, and 11 where the initialization (in black) was given manually far from the solution (in white) to test the robustness on the convergence step. The characterized features appear very close to the reality even in the noisy images.

In the case of real images, the results are presented into two different forms: a set of images of edges, corners, and triple junctions with lines superimposed and a set of tables where all the parameters of the different models are grouped to see the accuracy of the different models. In these table are used some notations presented into table 1 and in addition Σ^2 represents the error value per pixel evaluated from the equation (18) as:

$$\Sigma^2 = \frac{1}{M} F(\vec{x}) \tag{58}$$

where M is the number of considered pixels and $\vec{x}$ the vector of the result parameters.

As expected, the computational time has been found closely related to the window size and it is almost directly proportional to the number of points

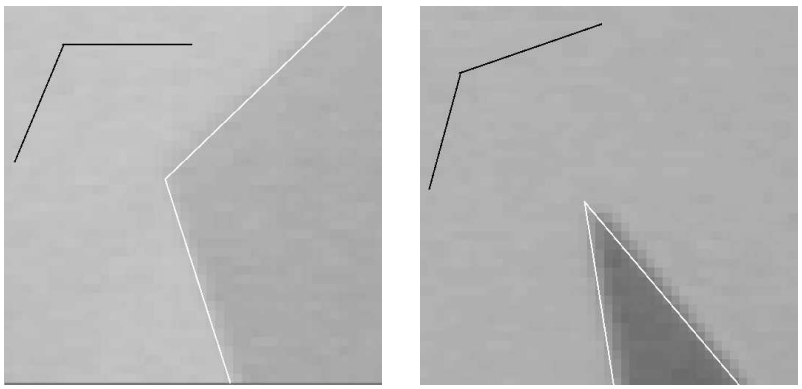

Figure 10: Convergence on corners (real images). For the left corner see table 4 and table 5 for the right corner.

within the window. This can be observed in tables 2 and 3 where different window sizes have been used for the case of an edge model. The ratio of the number of points between the two windows is 2.73 ($\frac{55*46}{33*28}$) and it is close to the ratio in CPU time 2.98 ($\frac{18.9}{6.33}$). One immediately can see the benefit of using the exponential and simplified models by looking at the CPU time required by each method in tables 2, 3, 4 and 5. In particular in tables 4 and 5, one can see that using a 2D Gaussian model is really very time consuming and there is a considerable time saving and many orders of magnitude improvements (between 50 and 60) in efficiency for the exponential and simplified models while the fit is very accurate for all the methods.

The different methods have been found to converge well even if the initialization is rather far for the solution. This is well illustrated in figures 8, 10, and 12 left, where the initial positions have been given by the user interactively far away from the solution. The initial solutions are superimposed in each image in black together with the final solution which is plotted in white (In all these figures, the solutions correspond to the application of the 2D exponential filter).

The parameters that correspond to the initial and final positions of the models are given in tables 2 and 3 for the edges shown in the figure 8 and in table 4 for the left corner of figure 10. In these tables, we give also the parameters found using the different methods and the associated CPU time and

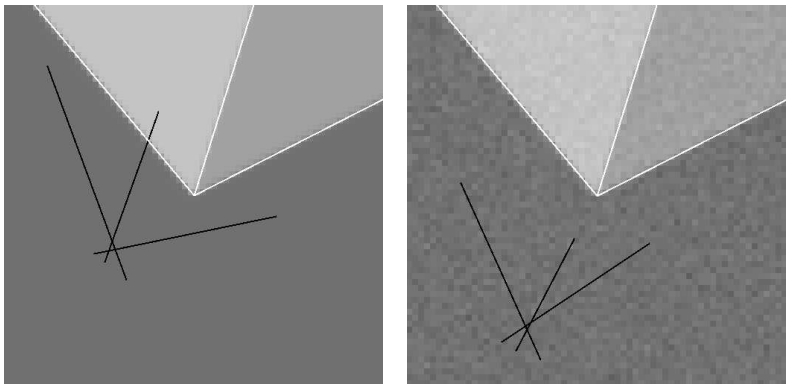

Figure 11: Convergence on vertices (synthetic images). Left: A synthetic triple junction. Right: The same junction with a Gaussian white noise of standard deviation $\sigma = 5$ added.

errors values. In the table 5 is grouped the result parameters obtained after the convergence of all the different models initialized by the *variance descent approach*; the considered corner is illustrated in right of figure 10 where the convergence of the exponential model manually initialized is shown. As it can be seen, the fit is found usually slightly better when using the exponential filter rather than the Gaussian filter. However, the difference in the amount of error using these two filters, cannot be considered as significant. This leads to say that there seems to be, at least, no *practical* justifications for the use of 2D Gaussian filter to modelize the blurring effect. On real images, the approximation errors corresponding to the different methods are very close to each other. One can see also that the reliability of the results given by the *variance descent approach* which works at a pixel precision but gives rather good results. Starting from the position given by the *variance descent approach* makes the model based approach to converge much faster and renders possible to start automatically the process. The right image of the figure 12 illustrates the difference between the results obtained using the *variance descent approach* (black lines) and the results obtained using the 2D exponential model (white lines) with initial conditions corresponding to the results given by the *variance descent approach*.

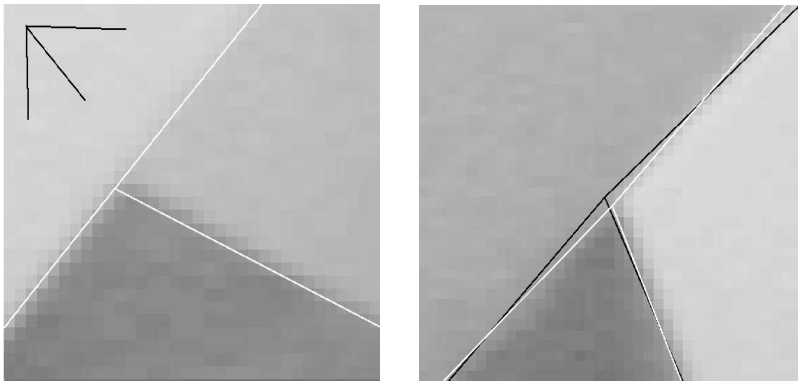

Figure 12: Convergence on triple junctions (real images).

As analyzed in the previous section, table 4 which refer to the corner in figure 10 clearly show that the fit using a 2D filter or the simplified models lead both to almost the same amount in the approximation error when dealing with a corner having an angle close to $\frac{\pi}{2}$. While table 5 shows that using a 2D filter leads to better results than using the simplified models for corners with small angle of aperture. Another interesting point to note in tables 4 and 5 is the difference in the recorded CPU time while the size windows is almost the same in both cases. This is mainly due to the initialization phase. In table 5, the approximation process started from the values given by the *variance descent approach* and which are very close to the solution, while in table 4, the initialization was given interactively and far from the solution by the user, in order to test the convergence of the approach. This illustrate the importance of the *variance descent approach* which initialize the process close to the solution, and reduces the CPU time by a ratio of roughly 2.

I	Interactive Initialization
V	Variance Descent Approach
G	Gaussian model of a step edge
E	Exponential model of a step edge
G2D	2D Gaussian model of a corner or a vertex
G1D	Simplified Gaussian model of a corner or a vertex
E2D	2D exponential model of a corner or a vertex
E1D	Simplified exponential model of a corner or a vertex

Table 1: Notations used in the other tables.

	θ	ρ	δ	A	B	time	Σ^2
I	-0.85024	31.13472	1.00000	150.00000	50.00000	...	...
V	-2.38755	2.73576	1.00000	147.75457	121.23070	.010	...
G	-2.38817	2.87059	3.30150	149.49144	119.23376	18.92	10.0075
E	-2.38829	2.82573	0.53104	149.70041	119.06906	18.31	9.47668

Table 2: Numerical results of the left step edge of figure 8 (window size: 55∗46 pixels)

	θ	ρ	δ	A	B	time	Σ^2
I	-2.61073	-5.984411	1.0000	150.0	50.0	...	...
V	-1.20599	15.941729	1.00000	166.91797	124.90028	.007	...
G	-1.255961	15.763497	0.87939	167.96623	123.70519	6.33	5.0145
E	-1.25599	15.766064	2.10067	167.99633	123.66725	5.35	5.1785

Table 3: Numerical results of the right step edge of figure 8 (window size: 33*28)

	I	V	G2D	E2D	G1D	E1D
x_0	6.71875	18.00000	18.059011	18.183433	18.107212	18.09888
y_0	37.62500	23.00000	22.767333	22.70569	22.733340	22.72887
θ, θ_1	-0.98667	0.758378	-0.250640	-0.249962	0.748979	0.748377
β, θ_2	1.97335	-1.23605	1.997360	2.007286	-1.250315	-1.249627
α, σ	1.0000	...	1.75608	1.391807	1.311966	1.295308
A	150.0000	117.80747	116.33554	116.20605	116.29908	116.14578
B	50.00000	146.66081	147.18489	147.26239	147.21306	147.28608
t(sec)	...	1.77	2639.94	55.31	43.61	33.92
Σ^2	...	...	15.2214	15.0845	15.2238	15.0328

Table 4: Numerical results on the real corner in the left of figure 10. (window size: 43*42 pixels)

	V	G2D	E2D	G1D	E1D
x_0	21.0	19.60480	19.594038	19.77442	19.786284
y_0	16.0	20.259840	20.279905	19.985608	19.966574
θ, θ_1	-0.755104	-1.114310	-1.113836	-0.824302	-0.824421
β, θ_2	-1.385448	0.57275	0.570550	-1.409809	-1.409163
α, σ	1.000000	1.049259	1.71040	1.007956	1.741566
A	52.386792	49.000757	48.326767	49.298809	48.554290
B	118.954064	120.692366	120.748031	120.693216	120.766856
t(sec)	1.68	1336.39	25.36	21.08	15.64
Σ^2	...	13.980211	13.764550	14.392698	14.3925925

Table 5: Numerical results on the real corner in the right of figure 10. The window size is 45*42 pixels.

6.2 Localization Accuracy of the Exponential models

In the previous section, the exponential model of features appeared as the most useful model based approach. Then, it will be used in this section for the purposes of a comparative study between the model based approach and the "classic" methods used in computer vision. These tests are done on a SUN SS10 workstation while those of the previous section were done on a SUN SS2 workstation.

6.2.1 Edge

The principle of the tests is as follows: from a set of edge parameters a series of edge images is derived, using formula (40), by moving the edge along its normal (i.e. modifying its distance to the origin: parameter ρ), or by changing its orientation (parameter θ) or by changing both parameters. The moving steps are from 0 to .99 pixel (16 values are used) for the translation and from 0 to .2 radian for the rotation (8 values are used). To these images a Gaussian white noise is added with 6 differents values of standard deviation σ from $\sigma = 0$ (no noise) to $\sigma = 5$. This yields a set of approximatively 200 test images similar to those of figure 7.

The test results are presented in the figure 13 where the mean error corresponds to the mean distance, given in pixels, of edge points from the real position. The first five curves correspond to the application of the model based approach where the window size vary from 8*8 to 64*64 pixels. The last two curves correspond, respectively, to the application of the nms classical approach usually used to extract edges (denoted as Nms in the figure)(see [9]), and the refinement of this method proposed by Devernay in [14].

The curves of the model based approach error are similar whatever the size of the window: The error is under five hundredths of a pixel. The nms approach is the worst presented in this test with an error of more than 3 tenths of a pixel, but it is very stable. The refinement of the nms proposed by Devernay allows to reduce the previous error to approximately one tenth of a pixel.

The bottom curves of the image 13 represent the CPU time needed by the different methods. This graphic shows that the model based approach applied with large window sizes (64*64, 32*32) needs a lot of CPU time, when for

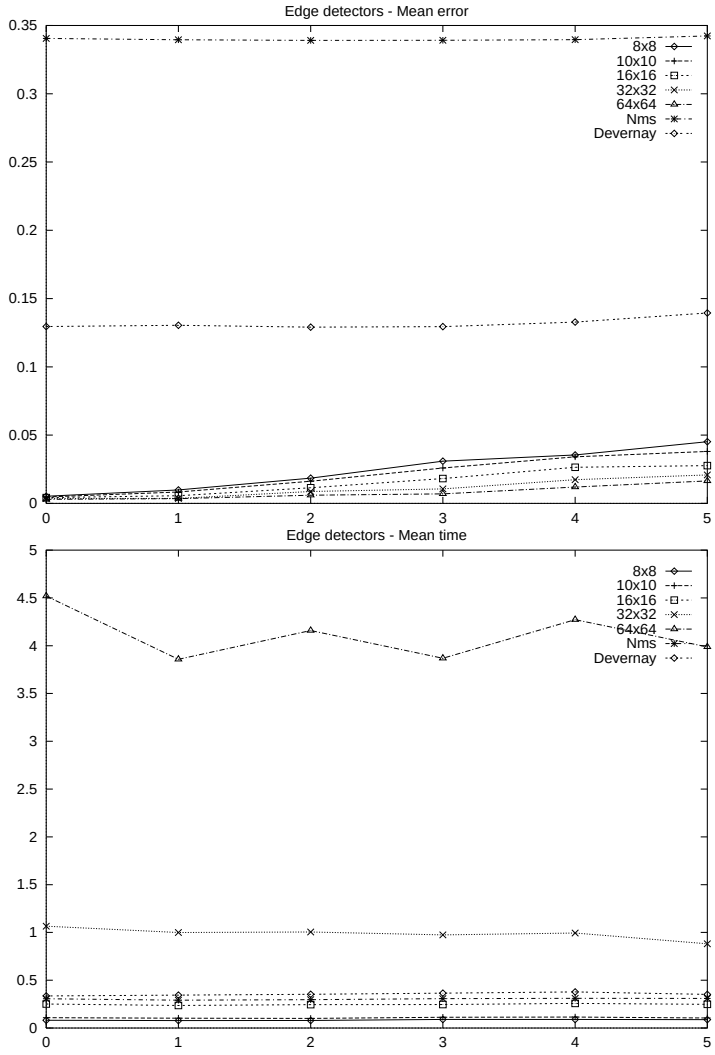

Figure 13: Edge detectors. Top: mean error in pixels; Bottom: mean time in seconds.

smaller window as for the two nms approaches the CPU time needed is under one half second.

These results show that the refinements up to sub-pixel accuracy improve the precision by a non-negligible factor for a small extra amount of CPU time.

Another conclusion after these tests is that, in most application, it is sufficient to use a window size around 16*16 pixels for the model based approach; In this case the mean error is under three hundredths of a pixel and the CPU time is just around a quarter of second.

6.2.2 Corner

In the corner case, the comparison tests are done on three sets of images. Each set corresponds to one fixed corner aperture, the different used apertures are $\frac{\pi}{4}$, $\frac{\pi}{2}$ and $\frac{3\pi}{4}$. Each set contains approximatively 300 synthetic noisy images of one corner: from a parameter set of a corner, a set of images is derived, using equation 43, by moving this corner by a variable step in the x direction, y direction and in the direction of the vector $\vec{n} = (1,1)^T$; 16 steps have been used from 0 pixel to 1 pixel. The added noise is a Gaussian white noise of variable standard deviation σ; 6 values of noise have been used from $\sigma = 0$ (no noise) to $\sigma = 5$. Two test images are presented in the figure 9.

The first goal of these experiences is to test the importance of the window size. This is done by running the approach on all the images with different window sizes (square windows with side of 64, 32, 16, 10 or 8 pixels). The mean distance of the obtained corners from the real position in function of the noise standard deviation is presented in the top part of figures 14, 15 and 16. As expected, if the noise level increases, the error increases significantly, but even for small windows the error does not exceed three tenths of a pixel. For window sizes of 32*32 pixels or 64*64 pixels the approach is very robust to the noise, as the error is under one tenth of a pixel. For smaller windows the results are less accurate, but for a noise level close to the amount of noise in real images the precision is roughly one tenth of a pixel.

In a previous section, the selection of the exponential model was done with respect to the precision criterion and also to improve the CPU time needed by the approach using the Gaussian model. The CPU time was an important parameter and it is interesting to test in which cases does it grows. The bottom part of figures 14, 15 and 16 shows a set of curves representing the mean time of the model based approach in function of the noise standard deviation for all the window sizes used previously. As expected the CPU time increases with the window size, but the noise level does not interact with the CPU time which

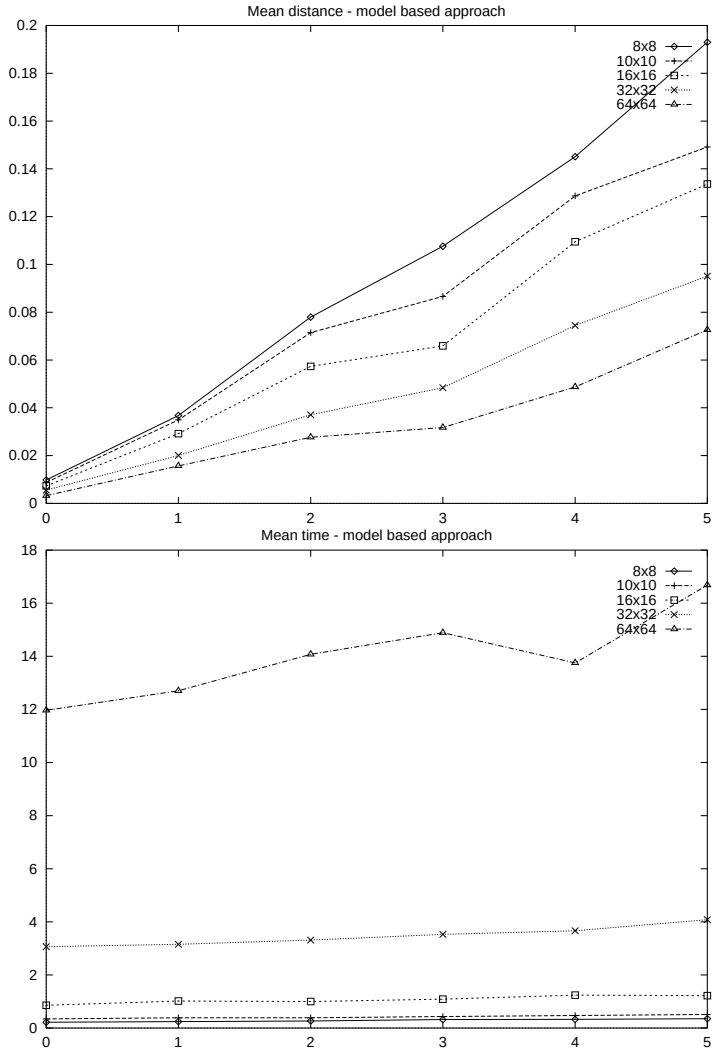

Figure 14: Corner characterization using the model based approach on corners of $\frac{\pi}{4}$ aperture. Top: average error in pixels; Bottom: mean CPU time in seconds.

is approximatively constant for a given window size. This leads to choose the best compromise between the time and the accuracy required. These remarks about the accuracy and the CPU time are independents of the corner aperture

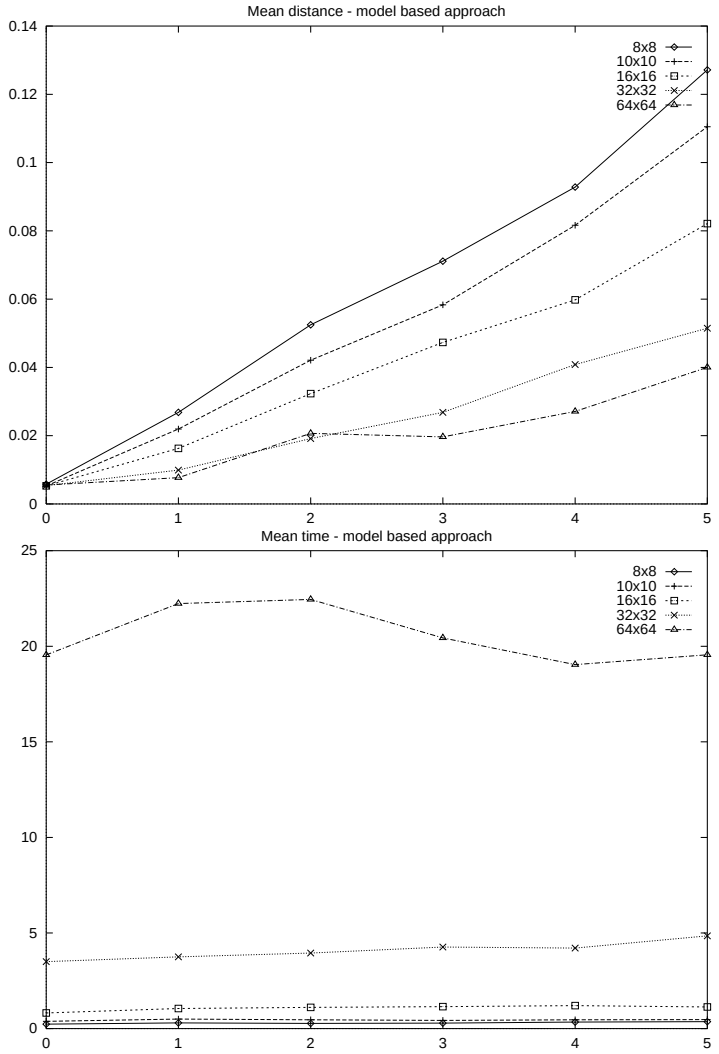

Figure 15: Corner characterization using the model based approach on right angles. Top: average error in pixels; Bottom: mean CPU time in seconds.

(see figures 14, 15 and 16), even if in the case of a right angle the accuracy is a bit better, there is no significant difference. Therefore, a window size around 16∗16 pixels appears to be the best window size to have a good compromise

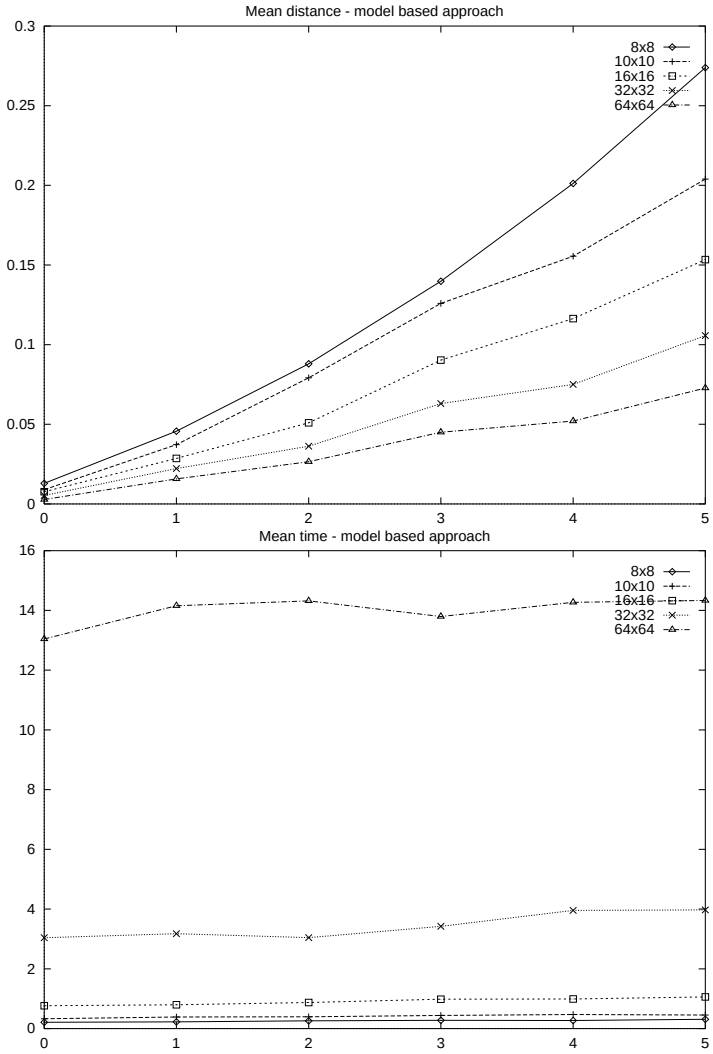

Figure 16: Corner characterization using the model based approach on corners of $\frac{3\pi}{4}$ aperture. Top: average error in pixels; Bottom: mean CPU time in seconds.

between a good accuracy and a fast CPU time, with respect to the considered noise levels.

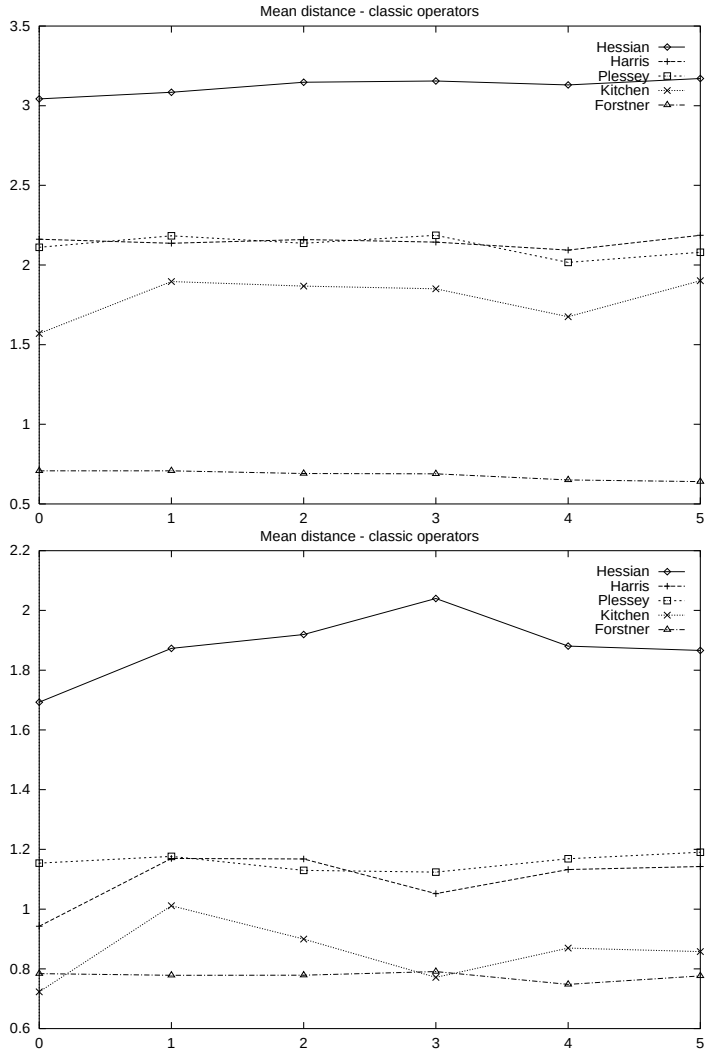

Figure 17: Average localization error in pixels of the classic operators. Top: $\frac{\pi}{4}$ corner. Bottom: $\frac{\pi}{2}$ corner.

The model based approach for the corner localization could be initialized manually or by using any classical corner operator. The used operators are those of Plessey, Harris, Kitchen, and of the Hessian determinant. Another

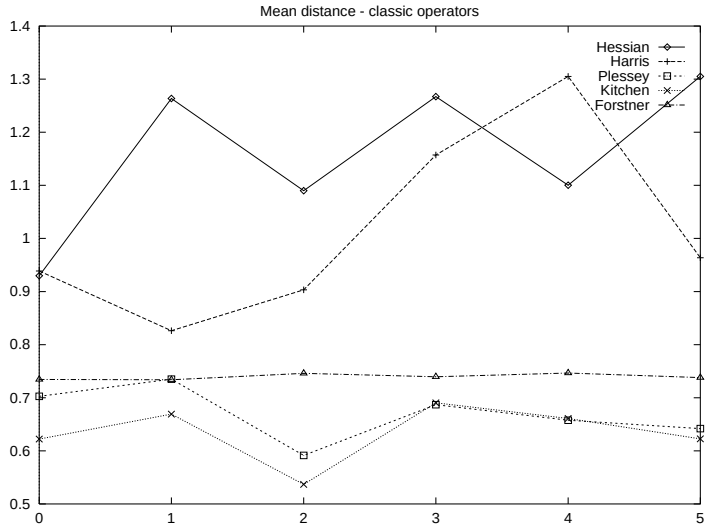

Figure 18: Average localization error in pixels of the classic operators for a $\frac{3\pi}{4}$ corner.

corner operator has been added to this comparison study, this is the Förstner operator (described in [16]); It works on a window centered on an estimate position of a corner and refines it, up to sub-pixel accuracy. The figures 17 and 18 show the curves of the mean distance, in pixels, of the obtained points from the real position for each detector; the different curves correspond to the different corner apertures considered in this test. These operators are very fast (CPU time<0.1 sec.) on these small images (64*64) and very stable with respect to the noise. In this case, the mean error depends on the corner aperture: for a small aperture the corner position is translated of two or three pixels away from the real position, when for a large aperture the error reduces to roughly one pixel. The more sensitive operator is the Hessian one and the best one is the Förstner one with only seven tenths of a pixel of error whatever the corner aperture and the considered noise levels.

Even if the accuracy of these operators seems quite good it is not sufficient for a lot of applications. Moreover an important information is given by these curves: the window size to use in the model based approach after these detec-

tors. The maximum mean error of these operators is approximatively 2.5 pixels then in the worst case the real corner is in a circle of a radius of 2.5 pixels, and in the best case the radius of this circle is 1 pixel. As the working window must include the corner and its neighborhood to insure the convergence of the model based approach, the radius of the circles must be enlarged of 2 or 3 pixels. So, the window size must be at least 7*7 pixels in the best case and 11*11 in the worst case.

After these tests, the best window size with respect to the accuracy, the CPU time and the corner detector uncertainty appears to be at least 11*11 pixels and better choice for the robustness needed in some applications (see section 6.3) is a window size of 16*16 pixels.

6.2.3 Triple Junctions

In this case, the comparison tests have been done on a series of approximatively 300 noisy synthetic images generated, using the formula 45, as for one set of images in the corner case. The figure 11 shows two test images. Here, the comparison is done between the model based approach and the Förstner's one. The other detectors used for corner localization have not been used, because on triple junction images, they could have multiple responses and the problem should be the selection of only one.

The figure 19 presents the mean distance, in pixels, of the extracted junction from the real position. The left curves of this figure show the mean errors of the model based approach in function of window size and of the noise standard deviation added to the image. Even if the error increases with the noise level, the worst error is less than one tenth of a pixel for a noise standard deviation of $\sigma = 5$. For real images, where the noise level is expected to be lower, the expected accuracy of the model based approach will be less than five hundredths of a pixel.

The bottom curve of the figure 19 presents the mean error of the Förstner approach applied on the same images. In this case (approximatively 1.2 pixels of error), this method is worse than in the corner case (near 0.7 pixel of error). but this is due to the inadequation of the method to non-symmetric features (see [16]). In fact the weights used in this method, for finding the closest point from the lines of the image, correspond to the gradient norm and one line of

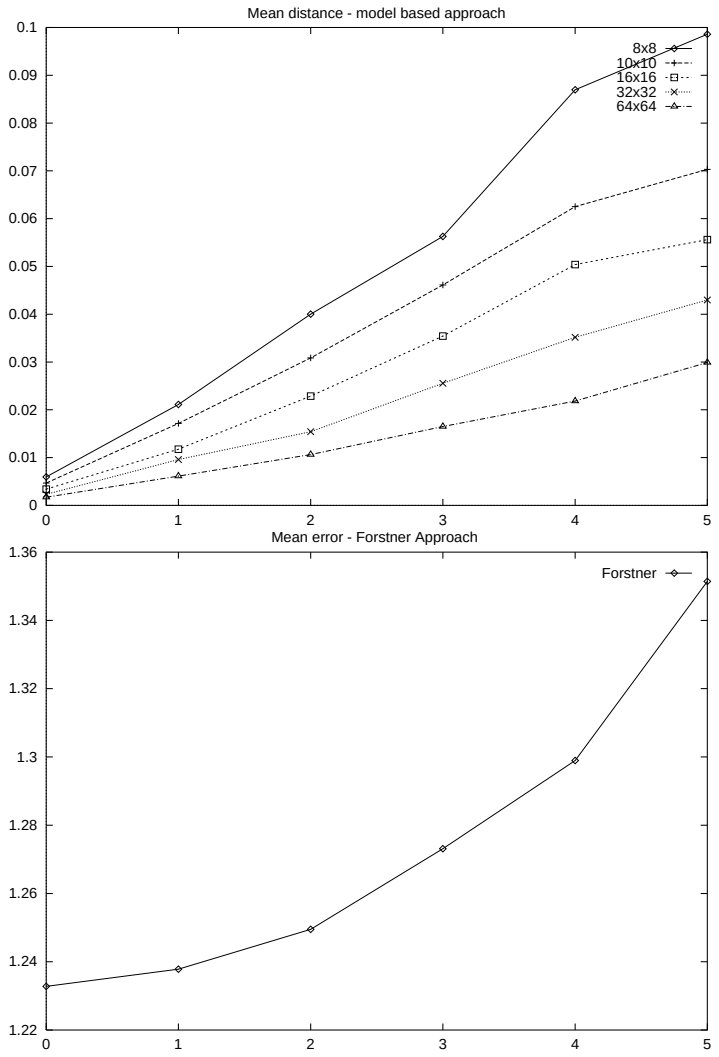

Figure 19: Mean error in pixels in the triple junction case. Top: Model-based approach. Bottom: Förstner method.

the triple junction has a gradient norm greater than the two others, then the first line "attracts" the junction position.

The figure 20 presents the CPU time needed by the model based approach. The Bottom curve shows the time needed by a window size equal to 64∗64

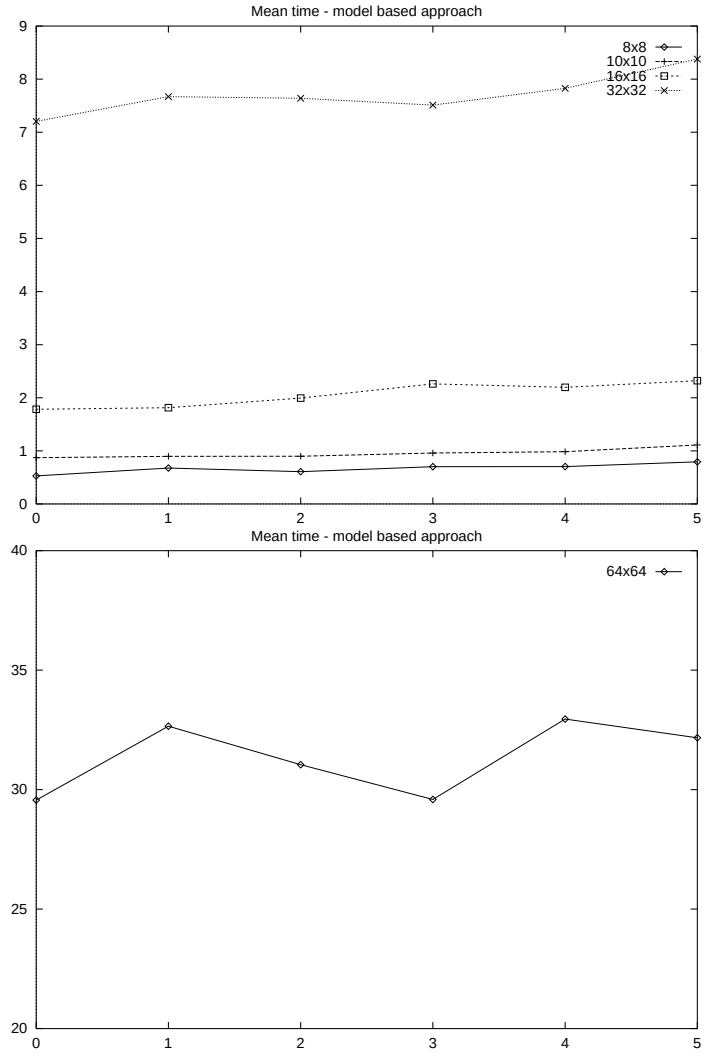

Figure 20: Mean CPU time needed by the vertex model based approach. Top: window sizes less than 64∗64 pixels. Bottom: window size of 64∗64 pixels.

pixels which is inefficient in practice. The curve, corresponding to a window size of 32∗32 pixels, denote also a slow speed of convergence. The CPU time

Method	Nb Pts	$\bar{d}$ in mm	σ_d
Harris	16	3.688760	2.406324
	32	4.990577	10.653361
	64	4.436885	8.325071
	96	4.555742	7.690778
	128	4.452017	7.120481
Model	16	0.819462	0.106355
Based	32	0.744987	0.090319
Approach	64	0.841448	0.243161
	96	0.821165	0.236551
	128	0.893669	0.252267

Table 6: Mean distances between the reconstructed 3D-points and the real points and the associated standard deviation, in the case of the calibration grid reconstruction.

needed by the Förstner method is approximatively constant and less than one tenth of a second, as in the corner case.

After observing these figures, the choice of window size for the model based approach is limited to windows smaller than 16*16 pixels because of the CPU time needed in the other cases. The error curves for windows smaller than 16*16 pixels show that the accuracy is less than one tenth of a pixel, so the precision required by the application will select the best size. And, as in the corner case, the choice of the window size depends also on the uncertainty on the first position estimation of the junction, because if the window is too small and badly placed around the junction, the method could fail, when for a larger window it would converge. In the following applications, a window size around 16*16 pixels appeared to be a good compromise between accuracy, CPU time and robustness with respect to the noise level present in the considered real images.

6.3 One Application: 3D-Reconstruction

This part presents some results about the use of the corner detectors. The goal of this experiment is to reconstruct projectively a 3D-objet from two im-

ages. First, two sets of matching points are extracted from the two images, one set in each image. These points are sufficient to determine the Fundamental matrix (see [22], [13]), if their number is greater than eight. From the Fundamental matrix and the choice of a plane in the projective space, the projective projection matrix are defined (see [33],[31]). And finally, from these projection matrix the extracted points can be reconstruct in the projective space. In order to evaluate the accuracy of this reconstruction, the experiments presented here are done on known objects. The reconstructed points are part of these three dimensional objects and the positions of some of these points are known in a Euclidean frame. Then, the transformation matrix between the Euclidean frame and the projective one into which the points have been reconstructed is computed, and the frame change is done. The error of the reconstruction is evaluated as the mean distance between the reconstructed points in the Euclidean frame and their theoritical positions and the associated standard deviation.

In the first example the object is a calibration grid (see figure 21). The extracted points correspond to the 128 corners of the 32 squares of the grid. In the Harris case, after extracting the corners with a low threshold, only the interesting points have been manually selected. For the model based approach, these points have been refined with the exponential corner model. This experiment is done for different numbers of points selected within the 128 points extracted by the Harris filter or those refined by the model based approach.

The figure 22 shows the reconstructed grids corresponding to the model based extracted points (left) and to the Harris points (right). In this case the 128 points were used but clearly, the right grid is not satisfactory because some squares became quadrilaterals, while the left one is very close to the expected result. The table 6 show the numerical results obtained for the various number of considered points. In the case of the Harris points, the error is roughly 5 mm and the associated standard deviation around 7. This experiment shows that the use of a lot of points does not reduce the error which is not very satisfactory. In the case of the model based points, the error is always under the millimeter, even in the case where only 16 points have been used. This information implies that for an application which require accuracy the use of 128 Harris points is worse than the use of only 16 points refined by the model based approach. Moreover, in some scenes the extraction of 128 pairs

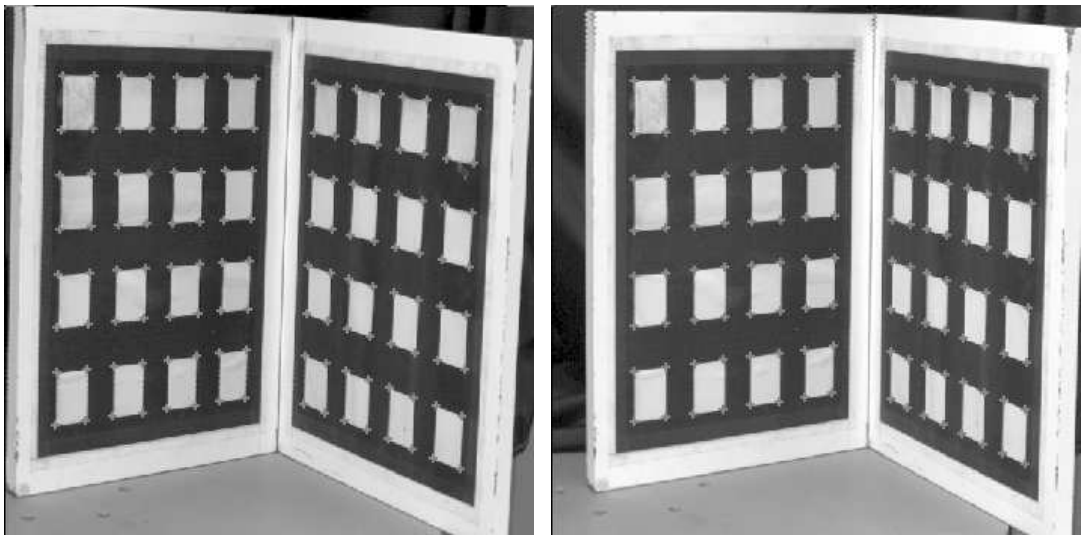

Figure 21: The images used for the reconstruction.

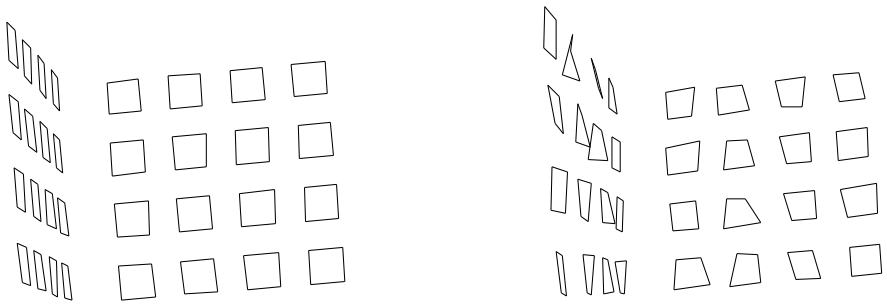

Figure 22: The reconstructed calibration grids. From the same point of view: left the result of the reconstruction with the model based approach and right the reconstruction using the Harris points.

Method	Nb Pts	$\bar{d}$ in cm	σ_d
Plessey	18	1.577284	0.748000
Model	18	0.554668	0.119528

Table 7: Mean distances between the reconstructed 3D-points and the real points and the associated standard deviation, in the case of the table reconstruction.

of points with a good accuracy is impossible, as in the next example, so the use of the model based approach is a good way to solve this problem. Another important point is to notice that the error is approximatively constant in these experiments for a given type of points; for example, the error is roughly 0.8 mm in the case of the model based points. This implies that in the succession of steps in this experiment, one step should not be as accurate as the others and then even with perfect points the reconstruction will not be perfect.

The second reconstructed object is a table with a box on its top (see figure 23). Only 18 points have been used, they have been extracted by the Plessey method and refined by the exponential corner model. The reconstruction scheme is the same as for the calibration grid, but here the objects have been measured manually. The results are presented in the figure 24, where the left picture shows the results of the reconstruction using the model based points, when the right one shows the reconstruction using the Plessey points. In the Plessey reconstruction, the box and the top of the table are deformed, when on the left the reconstruction seems perfect. Moreover, one can note the different appearance of the two reconstructed tables seen from the same viewpoint. The table 7 presents the mean distance of the reconstructed points from the real ones in cm. The errors are greater than in the previous reconstruction, but the calibration grid was closer to the camera than the table was. This explains the different accuracy between the grid reconstruction and the table one; but the use of the model based approach improves the results roughly of a factor 3 ($\frac{1.57}{0.55}$) with respect to the use of the Plessey detector.

The last example on 3D-reconstruction is about a desk reconstruction. The two used images are presented in the figure 25. In this experiment, the comparison is done between a reconstruction using 2-D points extracted by the

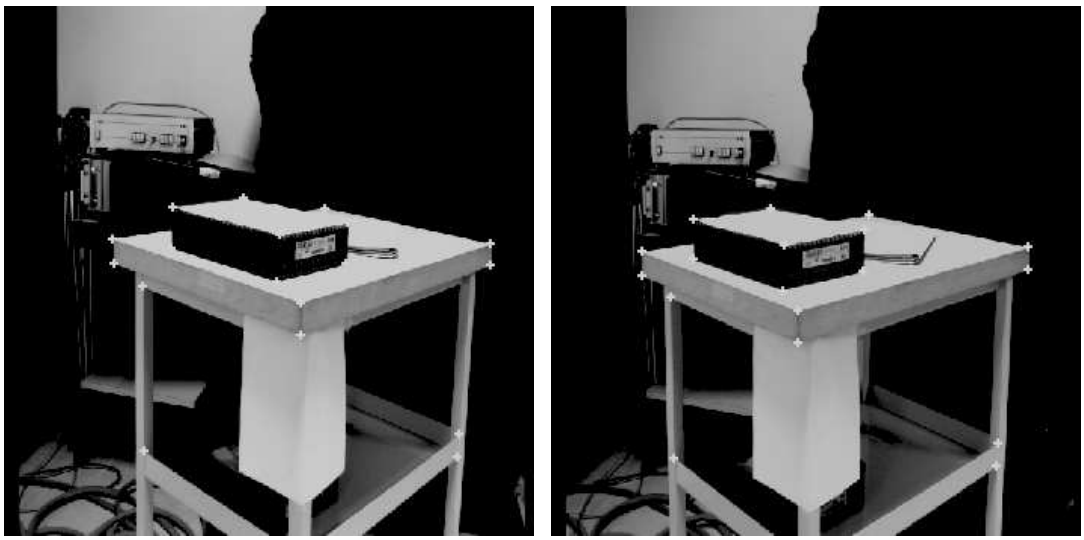

Figure 23: Images used for the reconstruction of a table

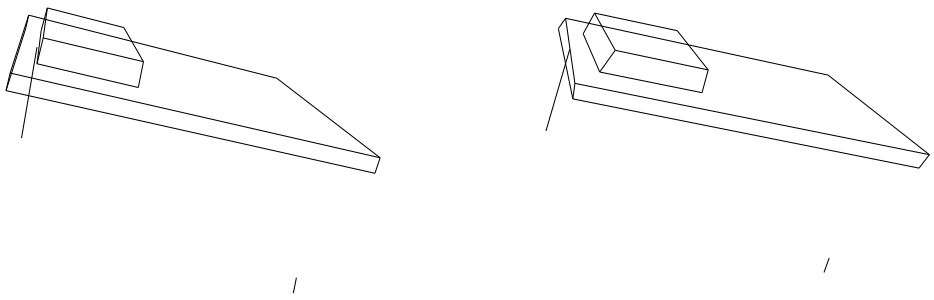

Figure 24: The reconstructed table. Left: with the model based approach; Right: with the Plessey points.

Figure 25: Another example of 3D-reconstruction: a room scene.

Hessian operator, the Plessey operator and 2-D points refined by the model based approach. The error values given in the table 8 show the improvement due to the use of the model based approach. In these images the objects are further away from the cameras than the table or the calibration grid were in the previous scenes; this explains that the error is greater than in the previous tests because the same uncertainty on the 2-D points implies a greater uncertainty on further away points. The figure 26 shows three different views of the reconstructed scene obtained from the points refined by the model based approach. Finally, the graphical and numerical results show that the refinement

Method	Nb Pts	$\bar{d}$ in cm	σ_d
Plessey	47	19.516485	365.948111
Hessian	56	18.210554	291.383838
Model	70	2.756333	5.213007

Table 8: Mean distance between the reconstructed 3D-points and their real positions and the associated standard deviation, in the case of the desk reconstruction.

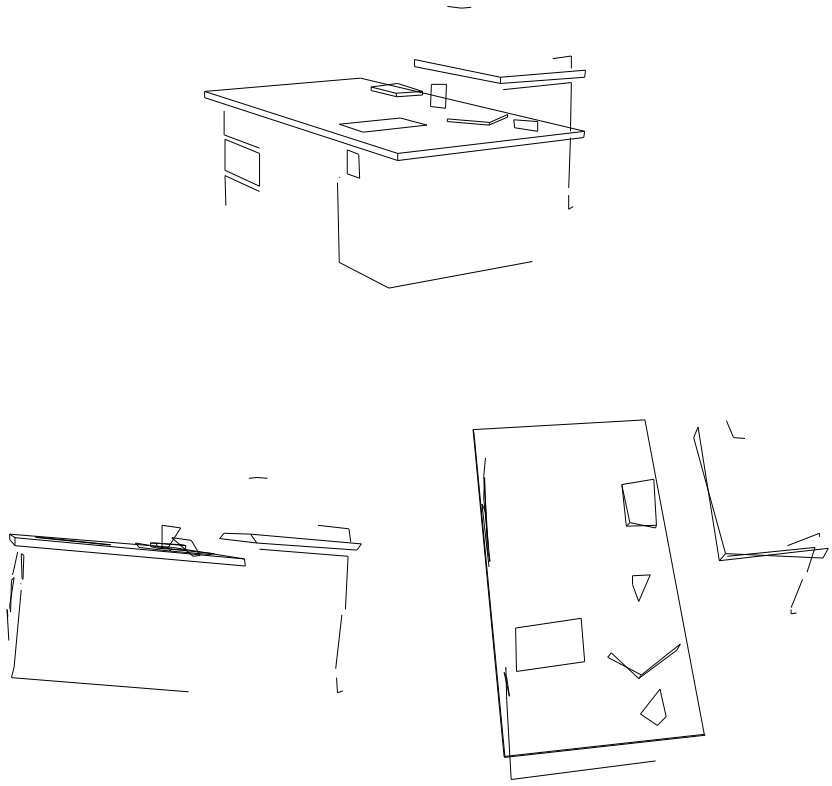

Figure 26: Three different views of the reconstructed room scene

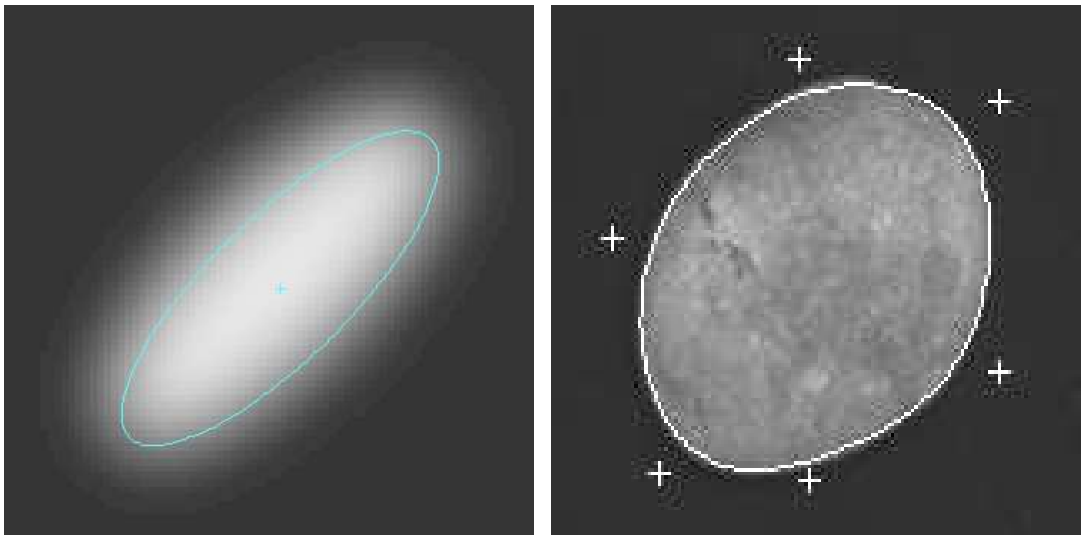

Figure 27: Extension to curve features.

of the corner positions by the model based approach improves greatly the final result.

7 Conclusion

An efficient model based approach has been developed in order to locate and characterize precisely important features such as edges, corners and vertices. Different models have been developed to describe efficiently these features and a minimization process has been proposed to find the parameters that best approximate locally the observed grey level image intensities.

Finally and as expected, an important point that can make the technique diverge, has been found to be the fulfillment within the working area of the underlying assumptions related to the different models. Different models than those considered within the working window leads to unsatisfactory results. Moreover, if another feature intersect the working window, the process could diverge because of the variance descent approach initialization. But, if the model based refinement could be initiated manually, it would converge. However, the use of the histogram information and in particular the number of

detected peaks in the histogram combined with the number of significant edge points detected along the frontier of the window, has proven to be a powerful idea to initialize correctly the model and also to give more information for the automatic choose of an appropriate window size. The value of the error in the approximation process has also been found to be a good indicator for the fulfillment of the underlying assumptions on the models. As seen in the previous sections, the window size did not appeared as an important parameter for an accurate detection of the parameters. Obviously, increasing the window size will reduce the noise effect, but generally size of order $15 * 15$, as seen in the section about reconstruction, appeared to be sufficient to tackle efficiently the amount of noise present in real images. In fact the good window size depends on the considered application and the type of images involved.

There are three mains directions in which the approach presented in this paper can be extended: The first direction is to generalize the models in order to take into account non planar intensity regions. The second direction is to deal with general curves rather than edges or polyhedral objects frontiers. The third direction is related to the application of the estimation of the blurring parameter to the problem of recovering depth from focus.

The figure 27 illustrates the extension of this approach to curve features: Left, an ellipse is obtained from a synthetic image created by bluring a synthetic ellipse with a 2D Gaussian filter; Right, a B-Spline closed curve is extracted from a real image of a stone. The ideas presented in this paper represent an important step in the application of a powerful formulation of the low level vision problems. While, a first step in this formulation has shown that some computational time requirements may decrease the interest of such model based approach, the simplifications and their application presented in this paper have shown that this formulation can be explored in depth. Although much work remains to be done for non-planar regions and curves, the results obtained in the application of this work to 3D-reconstruction indicate that this approach to locate and characterize image features accurately is a great improvement of the existing algorithms.

References

[1] H. Asada and M. Brady. The Curvature Primal Sketch. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, 8(1):2–14, January 1986.

[2] P.R. Beaudet. Rotational Invariant Image Opereators. In *Proc. International Conference on Pattern Recognition*, pages 579–583, 1978.

[3] F. Bergholm. Edge Focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, November 1987.

[4] V. Berzins. Accuracy of Laplacian Edge Detectors. *Computer Vision, Graphics, and Image Processing*, 127:195–210, 1984.

[5] D.J. Beymer. Finding Junctions Using the Image Gradient. In *Computer Vision and Pattern Recognition*, pages 720–721, Lahaina, Maui, Hawai, June 3-6 1991.

[6] K. Brunnström, T. Lindeberg, and J.O Eklundh. Active Detection and Classification of Junctions by Foveation with a Head-Eye System Guided by The Scale-Space Primal Sketch. Technical Report ISRN KTH/NA/P-91/31-SE, CVAP, Royal Institute of Technology, 1992.

[7] J.F. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1987.

[8] E. De Micheli, B. Caprile, O. Ottonello, and V. Torre. Localisation and Noise in Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1106–1117, October 1989.

[9] R. Deriche. Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector. *The International Journal of Computer Vision*, 1(2):167–187, May 1987.

[10] R. Deriche. Fast Algorithms For Low-Level Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–88, January 1990.

[11] R. Deriche and O.D. Faugeras. 2-D Curve Matching Using High Curvature Points: Application to Stereo Vision . In *Proc. International Conference on Pattern Recognition*, pages 240–242, 1990.

[12] R. Deriche and G. Giraudon. Accurate Corner Detection : An Analytical Study. In *Third International Conference On Computer Vision*, pages 66–70, Osaka, Japan, December 4-7 1990.

[13] R. Deriche, Z. Zhang, Q. T. Luong, and Olivier D. Faugeras. Robust Recovery of the Epipolar Geometry for an Uncalibrated Stereo Rig. In *Third European Conference on Computer Vision*, May 2-6 1994.

[14] Frédéric Devernay. A fast and efficient subpixelic edge detector. In *Quatrièmes Journées Orasis*, pages 146–151, Mulhouse, France, Oct 1993.

[15] L. Dreschler and H.H Nagel. On the Selection of Critical Points and local Curvature Extrema of Region Boundaries for Interframe Matching. In *Proc. International Conference on Pattern Recognition*, 1982.

[16] M. A. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centers of circular features. In *Proceedings of the Intercommission Workshop of the International Society for Photogrammetry and Remote Sensing*, Interlaken, Switzerland, 1987.

[17] G. Giraudon and R. Deriche. On Corner and Vertex Detection. In *Computer Vision and Pattern Recognition*, pages 650–655, Lahaina, Maui, Hawai, June 3-6 1991.

[18] A. Guidicci. Corner Characterization by Differential Geometry Techniques . *Pattern Recognition Letters*, 8(5):311–318, December 1988.

[19] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings 4th Alvey Conference*, pages 147–151, Manchester, August 1988.

[20] R. Horaud, F. Veillon, and T. Skordas. Finding Geometric and Relational Structures in an Image . In *Proceedings First European Conference on Computer Vision*, pages 374–384, 1990.

[21] L. Kitchen and A. Rosenfeld. Gray-Level Corner Detection. *Pattern Recognition Letters*, pages 95–102, December 1982.

[22] Q. T. Luong. *Matrice Fondamentale et Calibration Visuelle sur l'Environnement vers une plus grande autonomie des Systèmes Robotiques.* Thèse de doctorat, Université de Paris-Sud centre d'Orsay, 1992.

[23] D. Marr and E. Hildreth. Theory of Edge Detection. *Proc. Royal Soc. Lond.*, B 207:187–217, 1980.

[24] J. Matas and J. Kittler. Contextual Junction Finder. In *Proceedings of the British Machine Vision Conference*, pages 119–128, Leeds, September, 22-24 1992.

[25] G. Medioni and Y. Yasumuto. Corner Detection and Curve Representation using cubic B-spline. In *IEEE Int. Conf, Robotics and Automation*, pages 764–769, 1986.

[26] H.P Moravec. Towards Automatic Visual Obstacle Avoidance. In *Proc Int. Joint Conf Artificial Intelligence*, pages 584–587, August 1977.

[27] J.A. Noble. Finding Corners. *Image and Vision Computing*, 6:121–128, May 1988.

[28] A.P. Pentland, T. Darrell, M. Turk, and W. Huang. A Simple Real-Time Range Camera. In *Proc. International Conference on Computer Vision and Pattern Recognition*, pages 256–261, 1989.

[29] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C, The Art of Scientific Computing*, chapter 15.5, pages 681–688. Cambridge University Press, second edition, 1994.

[30] K. Rohr. Modelling and Identification of Characteristic Intensity Variations. *Image and Vision Computing*, 10(2):66–76, March 1992.

[31] Charlie Rothwell, Gabriella Csurka, and Olivier D. Faugeras. A comparison of projective reconstruction methods for pairs of views. Technical Report to appear, INRIA, 1994.

[32] B.M Ter Haar Romeny, L.M.J. Florack, J.J Koenderink, and M.A. Viergever. Invariant Third-Order Properties of Isophotes: T-Junction Detection. In *Proceedings Scandinavian Conference on Image Analysis*, August, 13-16 1991.

[33] Cyril Zeller and Olivier D. Faugeras. Applications of non-metric vision to some visual guided tasks. Research Report 2308, INRIA, Jul 1994.

[34] O.A. Zuniga and R.M. Haralick. Corner Detection Using the Facet Model. In *Proc. International Conference on Computer Vision and Pattern Recognition*, pages 30–37, 1983.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399