Ingeniería y Ciencia, ISSN 1794-9165
Volumen 1, número 2, páginas 5-23, septiembre de 2005

Edge and corner identification for tracking the line of sight

Ursula Kretschmer[1], María S. Orozco[2],
Oscar E. Ruiz[3] and Uwe Jasnoch[4]

Recepción: 21 de mayo de 2004 — Aceptación: 30 de julio de 2004
Se aceptan comentarios y/o discusiones al artículo

Resumen

Este artículo presenta un detector de aristas y esquinas, implementado en el dominio del proyecto GEIST (un Sistema de Información Turística Asistido por Computador) para extraer la información de aristas rectas y sus intersecciones (esquinas en la imagen) a partir de imágenes de cámara (del mundo real) contrastadas con imágenes generadas por computador (de la Base de Datos de Monumentos Históricos a partir de posición y orientación de un observador virtual). Las imágenes de la cámara y las generadas por computador son procesadas para reducir detalle, hallar el esqueleto de la imagen y detectar aristas y esquinas. Las esquinas sobrevivientes del proceso de detección y hallazgo del esqueleto de las imágenes son tratados como puntos referentes y alimentados a un algoritmo de puesta en correspondencia, el cual estima los errores de muestreo que usualmente contaminan los datos de GPS y orientación (alimentados al generador de imágenes por computador). De esta manera, un ciclo de control de lazo cerrado se implementa, por medio del cual el sistema converge a la determinación exacta de posición y orientación de un observador atravesando un escenario histórico (en este caso, la ciudad de Heidelberg). Con esta posición y orientación exactas, en el proyecto GEIST otros módulos son capaces de proyectar re-creaciones históricas en el campo de visión del observador, las cuales tienen el escenario exacto (la imagen real vista por el observador). Así, el turista "ve" las escenas desarrollándose en sitios históricos materiales y reales de la ciudad. Para ello, este artículo presenta la modificación y articulación de algoritmos tales como el Canny Edge Detector, "SUSAN Corner detector", filtros 1- y 2-dimensionales, etcétera.

Palabras claves: métodos de sintetización de imágenes, visión por computador,detección de aristas, detección de esquinas.

Abstract

This article presents an edge-corner detector, implemented in the realm of the GEIST

[1] Diplomada ingeniera, ursula.kretschmer@iww.uni-freiburg.de, investigadora, Fraunhofer Institute for Computer Graphics, Abteilung 5 (Geographic Information Systems).
[2] Ingeniera de Sistemas, moroo@uvigo.es, investigadora, Grupo I+D=I Enxeneria e Desenho Universidade de Vigo.
[3] Doctor en ingeniería, oruiz@eafit.edu.co, associate professor, EAFIT University.
[4] Doctor en ingeniería, info@GIStec-online.de, marketing director, GIStec.

project (an Computer Aided Touristic Information System) to extract the information of straight edges and their intersections (image corners) from camera-captured (real world) and computer-generated images (from the database of Historical Monuments, using observer position and orientation data). Camera and computer-generated images are processed for reduction of detail, skeletonization and corner-edge detection. The corners surviving the detection and skeletonization process from both images are treated as landmarks and fed to a matching algorithm, which estimates the sampling errors which usually contaminate GPS and pose tracking data (fed to the computer-image generatator). In this manner, a closed loop control is implemented, by which the system converges to exact determination of position and orientation of an observer traversing a historical scenario (in this case the city of Heidelberg). With this exact position and orientation, in the GEIST project other modules are able to project history tales on the view field of the observer, which have the exact intended scenario (the real image seen by the observer). In this way, the tourist "sees" tales developing in actual, material historical sites of the city. To achieve these goals this article presents the modification and articulation of algorithms such as the Canny Edge Detector, SUSAN Corner Detector, 1-D and 2-D filters, etcetera.

Key words: image synthesis techniques, computer vision, edge detection, feature detection.

1 Introduction

1.1 Goal of this work

The goal of this work is the extraction of edges and cornes from an image. In general, corner identification is successful only if a high level of detail is allowed by filtering stages. However, this level of detail is undesirable because of data size considerations. On the other hand, if the level of detail is reduced, only important landmarks survive (which naturally reduces the data size) but the cornes are lost. In this work, the second strategy was adopted, and complemented with a post-processing which infers the corners from the surviving landmarks (edges).

1.2 Context

The *GEIST* project[1] [1] is an educational game that aims to transmit historical facts both to youngsters and adults by means of Augmented Reality (AR). The aim of the game is by means of AR the player is immersed in ancient times: the times of the thirty years war in Heidelberg.

AR requires devices such as a mobile computer, sensors (Differential Global Positioning System (DGPS) and orientation tracker), a see-through display and a video camera as shown in figure (1). These devices are used as a system for displaying three dimensional ($3D$) images in the player's line of sight. Such images are superimposed on real world objects seen by the observer through the display. For a correct superposition of virtual and real images a video-based approach is pursued.

[1]The term *geist* is the German expression for the English word *ghost*

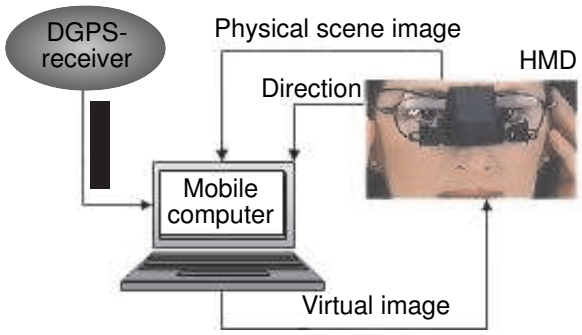


Figure 1: Tracking system for Augmented Reality

The player takes a path at will. In the game there are predefined places where he must stop. There, he sees reconstructions of the city as it was in ancient times on the see-through display. Besides, he interacts with a ghost. By asking questions the ghost leads the game, and the next step of the game depends on the player's answers. This means that the sequence of places to be visited is different for all players.

In order to superimpose virtual reconstructions on the real world objects it is necessary to track the line of sight of the observer. Tracking for the *GEIST* project takes initial estimates of the user's pose from the DGPS and the orientation tracker. Due to the required accuracy in the current application, these estimates must be corrected. A video-based approach is proposed for the correction of the initial estimates captured by the sensors.

The video-based approach is a system composed of several elements as shown in figure (2). The input data are:

1. Video frames coming from the camera figure (2-block a).
2. Pose estimates (position $\widehat{p}^T = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}$ and direction $\widehat{v}^T = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$) coming from orientation tracker and DGPS figure (2-block b).

The output data are: 3D images projected on the see-through display figure (2-block d). The process starts by obtaining from the sensors estimates of the player's pose. These estimates are used in querying the 3D model database figure (2-block c). The 3D model view is a reference image where the parameters position ($\widehat{p}$) and direction ($\widehat{v}$) known. This view is processed to extract edges and corners which are fed into a Matching Algorithm[2] [2] figure (2-block c) along with the edges and corners extracted from the video-frame of the camera. In case of a match, correction of direction ($\widehat{v}$) and position ($\widehat{p}$) is processed in order to obtain corrected direction ($\widehat{v_c}$) and position ($\widehat{p_c}$) of the user's line of sight. Finally, with this corrected estimates, a 3D reconstruction image is rendered on the HMD.

[2]The Matching Algorithm is not part of this work

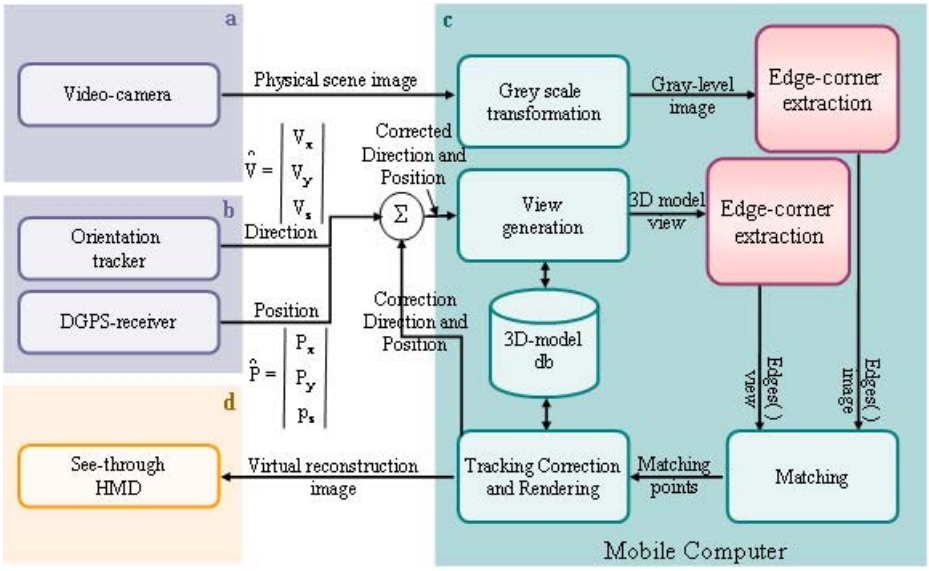


Figure 2: Block diagram of the tracking workflow of the *GEIST* project

2 State of the art

Image processing is required to transform data into analytical representation. Research in the field of human vision has shown that edges are the features that leads to object recognition by the brain: "Edges characterize object boundaries and are therefore useful for segmentation, registration and identification of objects in scenes"[3]. Edge identification can be done by different methods: the earliest methods used small convolution masks to approximate the first derivative [4, 5]. The use of zero crossings of the Laplacian of Gaussian (LoG) was proposed by Marr and Hildreth [6]. The first derivative of a Gaussian was proposed by Canny [7]. Directional second-derivative zero crossings was proposed by Haralick [8]. Morphology was proposed by Noble [9]. Venkatesch [10] uses "local energy" in the frequency domain to find edges.

Corners as well as edges are required for a full image description. In real time applications, such as the current one, corners are basic in object identification from frame to frame. Methods using binary edge maps to find corners have been suggested, as in [11, 12]. The edges are detected and then edge curvature is calculated in order to find corner locations. Beaudet [13] enhanced high curvature edges by calculating image Gaussian curvature. "Points of interest" were developed by Moravec [14]. These are defined as occurring when there are large intensity variations in every direction. Kitchen and Rosenfeld [15] used a local quadratic fit to find corners. Their work included methods based on change of direction along the edge, angle between most similar neighbors, turning of the fitted surface and gradient magnitude of gradient direction. Harris and

Stephens [16] described what has become know as the Plessey feature point detector. This is built on similar ideas to the Moravec interest operator, but the measurement of local autocorrelation is estimated from first order image derivatives. Rangarajan et al. [17] proposed a detector which tries to find an analytical expression for an optimal function whose convolution with the windows of an image has significant values at corner points. Arrebola et al. introduced different corner detectors based on local [18] and circular [19] histogram contour chain code. Quddus and Fahmy [20] presented a wavelet-based scheme for detection of corners on 2D planar curves. Zheng et al. [21] proposed gradient-direction corner detector that was developed from the popular Plessey corner detector. Smith and Brady [22] proposed the Smallest Univalue Segment Assimilating Nucleus (SUSAN) corner detector: given a circular mask delimiting a circular region of the image, the Univalue Segment Assimilating Nucleus (USAN) area is the area of the circular mask made up of pixels similar in intensity to the intensity of the central pixel (nucleus) of the mask. In a digital image, the USAN area reaches a minimum when the nucleus lies at a corner point. SUSAN is not sensitive to noise and is very fast for it only uses very simple operations. The SUSAN corner detector was chosen in this work because it detects corners at more than 2 adjacent regions, say junctions. Besides, it has a characteristic that makes it very attractive: no image derivatives are used, hence no noise reduction is needed.

3 Methodology

The Canny edge detector was already developed for the *GEIST* project. It had 2 deficiencies: (i) resultant edges were fragmented in little pieces, and (ii) corners were not detected. In the beginning a corner detector was developed to overcome the second. Though corner data improved, the large amount of fragmented edges remained as a problem. Therefore, the matching algorithm that processed the edges and corners was too slow or did not give any results at all. It was decided to build the Canny edge detector at new. In this section it is explained the methodology followed in the development of the edge-corner detectors.

3.1 Identification of edges in pixel domain

It is the process of locating pixels corresponding to edges in a given image. This process uses the Canny edge detector [7].

The identification of edges in pixel domain process is composed of four steps as shown in figure (3). The first step, the Gaussian Convolution (see definitions 3 and 9) aims at noise reduction and fine detail elimination. The First Derivative Convolution (definitions 7 and 8) step finds the big changes in the image intensity function. The Non-maxima Suppression step reduces edges to a single pixel width (or "path", see definition 4). The Hysteresis Thresholding step eliminates weak edges. As a result, edges in raster representation are obtained.

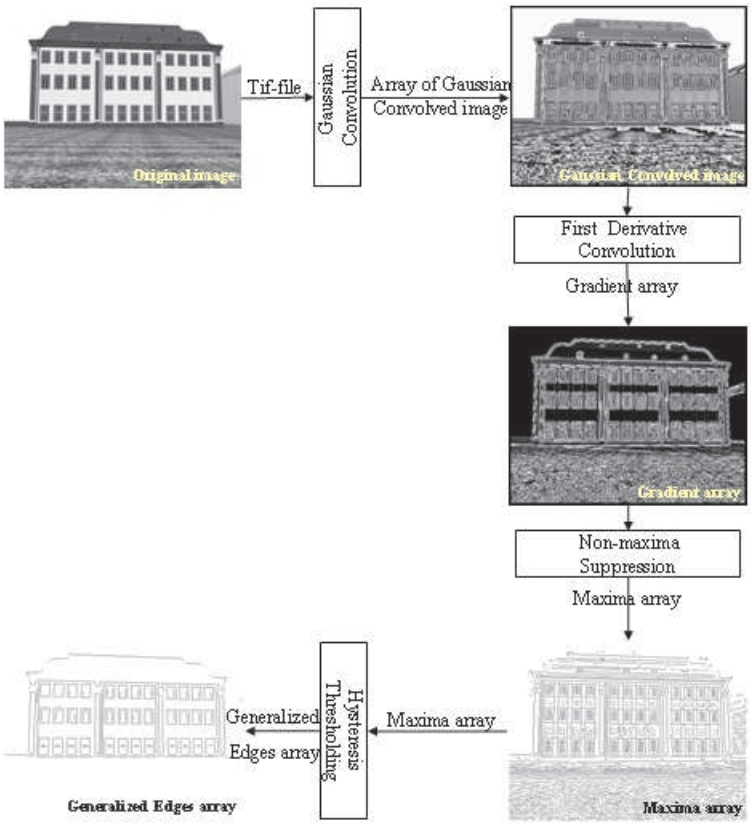


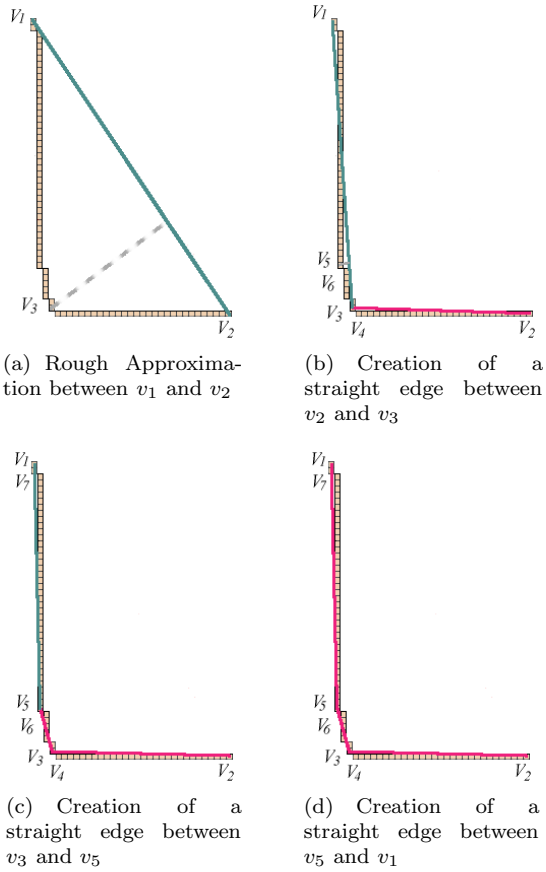
Figure 3: Block diagram of Identification of Edges in Pixel Domain

3.2 Vectorized edge synthesis

Consists on the transformation from a data structure, in the present case "raster" into vector representation. This process uses the Douglas-Peucker algorithm [23]. The amount of data that describes an edge based image is reduced by this process. Edges are transformed into straight segments as shown in figure (4). Two steps are followed in this process as explained below.

1. Generalized edges construction. The arrangement of lists of pixels related by 8-neighborhood (see definition 2) among them. Besides, pixels in such lists must comply with the Generalized Edge definition (see definition 5).

2. Vectorized edge synthesis. The straight edges are constructed in this step, based on the edge pixels of an E() list constructed in the previous step. The process is



(a) Rough Approximation between v_1 and v_2

(b) Creation of a straight edge between v_2 and v_3

(c) Creation of a straight edge between v_3 and v_5

(d) Creation of a straight edge between v_5 and v_1

Figure 4: Approximation by straight lines

shown in figure (4). It can be seen how a raster represented edge is transformed to vectorial representation. Initial data is a matrix of $n \times n$ pixels, this data is reduced to three vectors represented by three pairs of points, i.e. (v_1, v_5), (v_3, v_5), (v_3, v_2). It can be seen how the amount of data is siginificantly reduced.

3.3 Direct corner extraction from images

It is the identification of corners directly from an image. The corner detector implemented is the SUSAN algorithm proposed by Smith and Brady [22].

A circular area is delimited around a central pixel -called the nucleus (v_0). The procedure consists on deciding whether or not the nucleus is a corner. Four steps compose this algorithm as shown in figure (5).

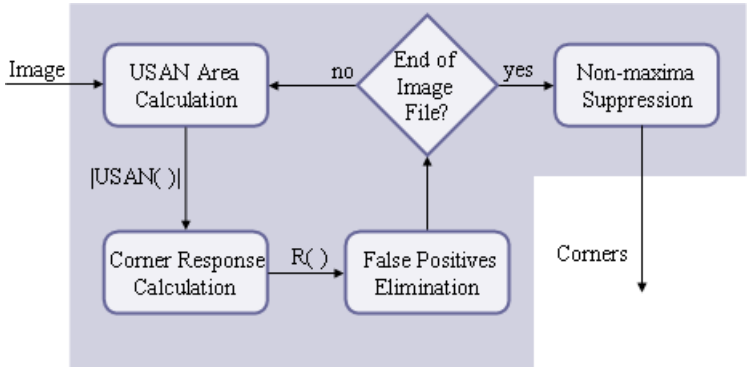


Figure 5: Block diagram of Direct Corner Extraction from a given Image

1. Univalue Segment Assimilating Nucleus (USAN) area calculation. The USAN area is the area of the circular mask made up of pixels similar in intensity to the intensity of the nucleus (v_0) of the mask.

   Pixels whose intensity value is similar to the nucleus of the circular mask are found and counted in this step. The delta intensity function (see definition 12) applies the intensity difference threshold ΔI_t (see definition 11) to the pixels of the mask in order to find the USAN area.

2. Corner response calculation. The corner response $R(v_0)$ (see definition 14) is the capacity of the nucleus to be considered a corner. It is calculated based on the geometrical threshold g_t (see definition 13) and the size of the USAN area. Larger values mean grater possibility for the nucleus to be considered a corner.

3. False positives elimination. False positives are points wrongly reported as corners. Cases like noise, lines across the circular mask may be reported as corners. Hence the aim of the false positives elimination is to detect and eliminate such points.

4. Non-maxima suppression. This step finds a maximum value of corner response in a subwindow of 5 x 5 pixels. Once the previous three steps are finished for all pixels of the image, the non-maxima suppression is applied to the set of pixels represented by their corner response.

   The subwindow is placed on each pixel and is moved from left to right and from top to bottom of the corner response array. A maximum value is found for the subwindow. In case that more than one maxima value were found in the subwindow, the current pixel (the subwindow central pixel) is suppressed i.e. its corner response is turned to zero.

3.4 Edge-corner alignment

This process aims at putting together the detected corners and edges. The guiding principle is the relationship of collinearity between an edge and its neighboring corners. The parametric equation of the line [24, page 118] was used in this step.

In order to find a collinear corner, the distance between the corner and each of the vertices of the current edge must be calculated. This step helps to ensure that neighboring corners are chosen for the edge-corner alignment. In case a corner is along the same straight line of an edge, then the edge is extended to meet the corner.

The same procedure must be repeated for all edges obtained in the vectorized edge synthesis.

3.5 Indirect corner extraction

The results obtained in the edge-corner alignment process were insufficient, because many pairs of edges remained without a common corner after a lengthy edge-corner alignment process. Therefore, an alternative algorithm was developed, using indirect corner calculation based on the existent edges. Its description follows:

Indirect corner calculation. Corners are calculated by finding the intersection point ($p_i()$) between two non-parallel edges. For a given pair of edges to be related through a common corner they must be closer than a user defined threshold. The threshold is an integer number in the range of [0 - 10]. Two edges being far away from each other are not sought to be related by a corner. The calculation of the intersection point (pixel) follows:

Intersection point $p_i\left(e_1, e_2\right)$. The intersection point is the pixel where two edges ($e()$) meet. Let two edges be: $e_1\left(\left(x_1, y_1\right), \left(x_2, y_2\right)\right)$ and $e_2\left(\left(x_3, y_3\right), \left(x_4, y_4\right)\right)$. The parameter t of the intersection point is [24, page 113]:

$$t = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}\,.$$

And so, the intersection point is defined as :

$$\begin{aligned} x &= x_1 + t(x_2 - x_1)\,, \\ y &= y_1 + t(y_2 - y_1)\,. \end{aligned}$$

In addition to the indirect corner calculation, 1-D filters were used to improve the time response in the process of identification of edges in pixel domain (see definition 15).

4 Results

The results of the implemented algorithms is discussed in this section. They are based on the photo of one of the buildings of the University of Heidelberg as shown in figure (6).

Figure 6: Original image

This section is composed of 3 cases of study:

1. Edge Vectorization (EV).
2. Edge Vectorization and Direct Corner Extraction (DCE).
3. Edge Vectorization and Indirect Corner Extraction (ICE).

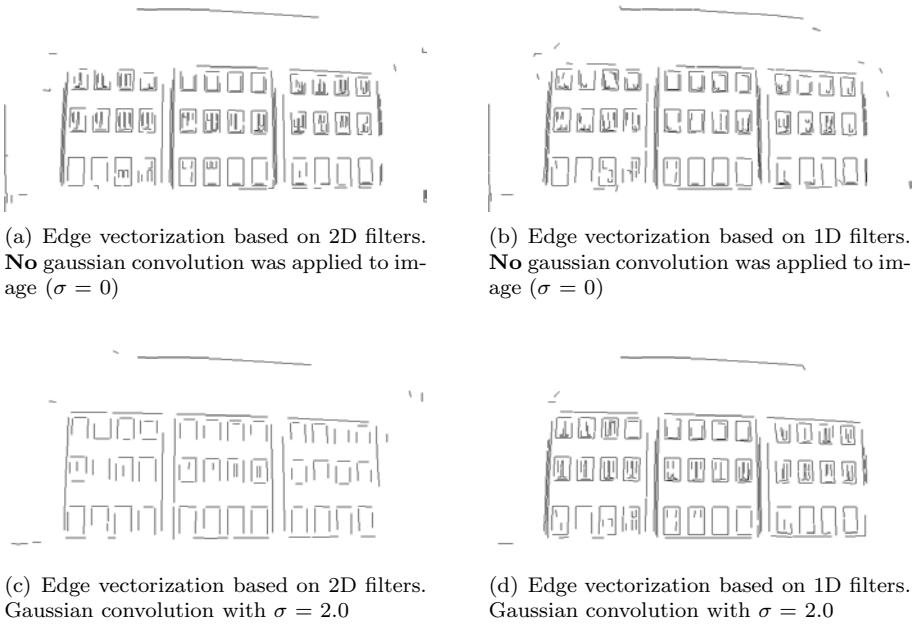
Parameter values for these cases of study are described in table (1).

Table 1: Parametrization values for the experiments

	σ	$T1$	$T2$	s_t (pix.)	filter	ΔI_t	g_t (pix.)	$dist$ (pix.)	Process
Fig 6a	0,0	250	200	2	2D	-	-	-	Edge Vect.
Fig 6b	0,0	80	60	2	1D	-	-	-	Edge Vect.
Fig 6c	2,0	400	300	2	2D	-	-	-	Edge Vect.
Fig 6d	2,0	100	80	2	1D	-	-	-	Edge Vect.
Fig 7b	-	-	-	-	-	20	25	-	Dir. Corner Calc.
Fig 7c	2,0	400	300	2	2D	20	25	-	Dir. Corner and Edge Vect.
Fig 8b	2,0	400	300	2	2D	-	-	7	Ind. Corner Calc. and Edge Vect.

The EV case consists on edge detection based on 1D and 2D filters. Results of this processes are seen in figure (7). The figure shows the contrast in the use of 2D filters vs 1D filters for edge detection and how different levels of smoothing (σ = 0,0 and σ = 2,0) affect the resulting edges. It can be seen in figure (7-d) that edge detection

based on 1D filters is best at keeping corners, but too much fine detail is present in final edges. Conversely, edges obtained by the application of 2D filters have less fine detail, but corners are lost in the process of smoothing.



(a) Edge vectorization based on 2D filters. **No** gaussian convolution was applied to image ($\sigma = 0$)

(b) Edge vectorization based on 1D filters. **No** gaussian convolution was applied to image ($\sigma = 0$)

(c) Edge vectorization based on 2D filters. Gaussian convolution with $\sigma = 2.0$

(d) Edge vectorization based on 1D filters. Gaussian convolution with $\sigma = 2.0$

Figure 7: Edge Vectorization (EV) with varying filters. **No** corner generation applied

Results shown in figure (7-c) are the edges that best fit the needs of this work, because they characterize the important features of the original building and they are free of fine detail. Therefore these have been used as input for the remaining two cases of study, i.e. DCE an ICE. Quantitative results of the EV process shown in table (2) are also taken from figure (7-c).

Table 2: Quantitative results for the cases of study

	False positives (%)	False negatives (%)	Good localization (%)
Edges EV (fig 7c)	8,33	21	79
Corners DCE (fig 8c)	1,30	71	29
Corners ICE (fig 9)	1,30	40	60

Figure (8) shows results of the DCE case. In this case two additional processes are implemented: Corner extraction and edge-corner alignment. In (8-a) the edges convey the most important information of the building, i.e. the straight lines. The missing

corners of this figure are going to be obtained by aligning the corners found in (8-b) to the edges. Figure (8-c) shows results of the Edge-corner Alignment process. It can be seen that even though many edges appear joined by a common corner there are still many missing corners. In order to improve the results the following method was implemented.

(a) Edge vectorization

(b) Direct corner extraction

(c) Edge-corner alignment from data shown in (8a) and (8b)

Figure 8: Edge-corner Generation process using the Direct Corner Extraction algorithm (DCE)

Localization is one of the parameters to measure good quality in feature detectors. Good localization means that the reported corner or edge position is as close as possible to the correct position [22, page 5]. Results of the ICE case figure (9) are superimposed on original image as shown in figure (10). Note the good localization of edges and corners. In table (2) results of the detection show that 79% of the edges were localized correctly, 60% of the reported corners have good localization in the ICE case, whereas 29% of the corners have good localization in the DCE case.

Figure 9: Edge-corner generation process using Indirect Corner Extraction algorithm (ICE)

Figure 10: Original image with superimposed edges and calculated corners

Another parameter that measures quality in feature detectors is good detection. Good detection is measured by the number of false negatives and false positives reported [22, page 4]. The arrows on the left side of figure (10) are pointing to two cases of false positives. These are edges that are incorrect and should not have been reported. The percentage of false positives for all cases of study is low. For edges $8,33\%$ and for corners $1,30\%$. False negatives are features that should have been reported. Elimination of fine detail causes false negatives. False negatives in the EV case is 21%, corner false negatives in the DCE case is 71%, whereas corner false negatives for DCE is 40%.

5 Discussion and conclusions

This article has addressed the detection of straight edges and corners present in architectural images. Applications are discussed in the sub-sequent matching of computer-generated vs. camera sampled pictures of historical buildings for Computer Assisted Tourist Information Systems (specifically, the GEIST project).

1. In this work the straight lines are defined as principal features because they convey the most important information about urban buildings. Directional derivative filters were used in order to maximize the detection of straight lines in the original image. Besides the use of Gaussian convolution favored the fine detail absence from final edges.

2. The smoothing achieved by the gaussian convolution gives as a result the lost of fine detail, noise and corners. Even though the first two are good consequences, the lack of corners is an undesired consequence, because corners are indispensable in the workflow in order to obtain the user's pose. This situation gave rise to the need to implement an additional process: the corner generation.

3. Two methods of corner generation were developed: DCE and ICE. The method ICE improved the amount of edges joined by a common corner respect to the DCE algorithm. Time response was also improved by corner generation based on ICE.

   Advantages of ICE:

   (a) More efficient.

   (b) More edges related to a common corner.

   Disadvantages of ICE:

   (a) Localization of corners may not be correct. Two neighboring edges should not be joined through a corner. But, because of their proximity, the algorithm does join them. Therefore a false positive is obtained.

   (b) In the presence of fine detail, results of ICE are very poor. Edges become misshaped after applying the corner generation process.

4. The results of the corner generation processes may be improved by finding a directly extracted corner (DCE) in the neighborhood of an indirectly calculated corner (ICE).

5. Results of this work show that a video based approach for the Detection of the Line of Sight problem is feasible. The processes implemented in this work extract principal features from video images. Through these features it is possible the calculation of the line of sight in a unexpensive and accurate way.

5.1 Future work

Continuation of this work is possible in several ways: (i) to produce, from the urbanistic data base for a particular landmark, not raster images but vectorized information. In this way, the raster image of the tourist camera would be matched against a vectorized image, therefore improving the efficiency of the process. (ii) once a landmark has been identified as visited by the tourist, and the positions and trajectory of his/hers have been indentified, only neighboring landmarks need to be displayed and searched through. Remote ones are not accessible to the viewer, as a non-continuous trajectory would be required to approach them. (iii) curved lines are to be identified, for the case in which architectural landmarks present circular or elliptical features.

A Definitions

In this section basic terms for the comprehension of this article are detailed.

1. Image $I()$
   Given $I : Z \times Z \longrightarrow N$
   Domain $[0, X_c] \times [0, Y_r]$, where X_c : Column pixels, usually 639; Y_r : Row pixels, usually 479.
   Range of *I( )* function is $[B_L, W_L]$, where B_L : Black level (usually 0); W_L : White level (usually 255).

2. 8-Neighborhood of a pixel (i, j) $N_8(i, j)$
   The relationship between pairs of pixels (i, j) and (m, n) that share one or two end-points.
   $N_8(i, j) = \{(m, n) | (|m - i| = 1) \vee (|n - j| = 1)\}$.

3. Convolution
   Convolution is a linear operation that calculates a resulting value as a linear combination of the neighborhood of the input pixel and a convolution mask [25, page 68-69]. In equation (1) the output pixel -*Final(i,j)*- is calculated as a linear combination of the neighborhood of the input pixel -*Inital(i,j)*- and the coefficients of the two dimensional (2D) convolution filter *M*.

$$Final(i, j) = Initial(i, j) * M = \sum_{k=-1}^{+1} \sum_{l=-1}^{+1} Initial(i + k, j + l) M(k, l). \quad (1)$$

   Final(i,j) in equation (1) is the sum of the products of pixels in the neighborhood of (i, j) and filter M. In the present work two steps apply filters, they are: Gaussian and First derivative convolution.

4. Path $p(v_0, v_f)$
   A path is a non-selfintersecting, unit-width sequence of vertices, starting at v_0 and ending at v_f, made up of orthogonal or diagonal steps. No assumptions are made on $I(v_i)$

$$p(v_0, v_f) = [v_0, v_1, ..., v_f]\, s.t.$$
$$(v_i \in [0, X_c] \times [0, Y_r]) \wedge (D_8(v_i, v_{i+1}) \leq 1) \wedge (|N_8(v_i)| \leq 2).$$

5. Generalized edge $E(v_0, v_f)$
   A generalized edge between v_0 and v_f , $E(v_0, v_f)$, is a path $p(v_0, v_f)$ between them, built with high gradient pixels. All pixels of the path have gradient larger than T_2, and at least one pixel in the path has gradient larger than T_1. A generalized edge $E()$ does not have to be straight.
   Given $T_1, T_2 \in N$, where $T_1 > T_2$,

$$E(v_0, v_f) = [v_0, v_1, ..., v_f] s.t. (p(v_0, v_f)) \wedge$$
$$(\forall v_i \in p(v_0, v_f), \nabla I(v)_{v=v_i} > T_2) \wedge$$
$$(\exists v_j \in p(v_0, v_f), \nabla I(v)_{v=v_j} > T_1).$$

6. Straight edge (or simply, *edge*) $e\left(v_0, v_f\right)$
   An *edge* is an approximately straight generalized edge, in which the perpendicular distance from each pixel to the straight segment, $\overline{v_0 v_f}$, joining the extremes is bounded by an ϵ value.

$$e\left(v_0, v_f\right) = \left[v_0, v_1, ..., v_f\right] s.t.$$
$$E(v_0, v_f) \wedge \forall v_i \in E(v_0, v_f), d\left(v_i, \overline{v_0 v_f}\right) < \epsilon\,.$$

   Typical values for ϵ: $[0, 10]$.

7. Orthogonal differences, $\Delta_i I, \Delta_j I$
   The orthogonal differences $\Delta_i I$ and $\Delta_j I$ approximate the first directional derivatives of $I()$ in the X and Y directions respectively [25, page 80]. They are applied on functions, and return a scalar value:
   $\Delta_i() : I() \rightarrow R$
   $\Delta_j() : I() \rightarrow R$
   calculated as:
   $\Delta_i I\left(i, j\right) = I\left(i+1, j\right) - I\left(i-1, j\right)$
   $\Delta_j I\left(i, j\right) = I\left(i, j+1\right) - I\left(i, j-1\right).$
   Notation simplification:
   if $v = (i, j) \Rightarrow \Delta_i I\left(i, j\right) = \Delta_i I(v)\,.$

8. Gradient operator $\nabla I(i, j)$
   A gradient operator is the vector formed by the directional derivatives or orthogonal differences (see definitions A7). The gradient operator is applied on functions, and returns a vector:
   $\nabla() : I() \rightarrow R^2$
   $\nabla() : I() = \nabla I(i, j) = (\Delta_i I(i, j), \Delta_j I(i, j))$.
   Collaterally, its direction and magnitude are defined:

   (a) Gradient direction, $\theta(I(i, j))$, is the angle (in radians) from the x axis to the point (i, j):
   $\theta(I(i, j)) = arg(\nabla I(i, j)) = \arctan\left(\frac{\Delta_j I(i,j)}{\Delta_i I(i,j)}\right).$

   (b) Gradient magnitude, $|\nabla(i, j)|$, is the norm of the Gradient operator:
   $|\nabla I(i, j)| = |(\Delta_i I(i, j), \Delta_j I(i, j))| = \sqrt{\left(\Delta_i I\left(i, j\right)\right)^2 + \left(\Delta_j I\left(i, j\right)\right)^2}\,.$

9. Gaussian operator $G_{\Delta x, \Delta y, \sigma}\left(i, j\right)$
   Gaussian operator is a function that reduces the possible number of frequencies at which image intensity function changes take place.
   $G_{\Delta x, \Delta y, \sigma}\left(i, j\right) : Z^2 \rightarrow R$

$$G_{\Delta x, \Delta y, \sigma}\left(i, j\right) = \begin{cases} \frac{1}{2\pi\sigma^2} e^{-\left(\frac{i^2+j^2}{2\sigma^2}\right)}, & \text{if } -\Delta x \leq i \leq \Delta x\,, \\ & \quad -\Delta y \leq j \leq \Delta y\,. \\ 0, & \text{otherwise.} \end{cases}$$

Usually $\Delta x = \Delta y = 3\sigma$. Typical values for σ are in $[0, 10]$.

10. Circular mask $M\,(x, y, R)$
    Set of pixels (usually 37) that make up the interior and boundary of a circular subwindow of an *I()*[22].
    $M\,(x, y, R) = \left\{(i, j)\,|\,(x-i)^2 + (y-j)^2 \leq R^2\right\}$.
    With nucleus v_0 the center of the circle: $v_0 = (x, y)$.
    And v_i any other point in the circle: $v_i = (i, j)$.

11. Intensity difference threshold ΔI_t
    A threshold of the intensity function of an image. It is used in order to define similarity in intensity among the nucleus v_0 and all other pixels of the circular mask *M( )*[22].
    Typical values for ΔI_t are in $[10, 60]$.

12. Delta intensity function $\delta_{v_0,\Delta I_t}\,(v_i)$
    Difference in intensity between pixels v_i and v_0 [22].

$$\delta_{v_0,\Delta I_t}\,(v_i) = \begin{cases} 1, & \text{if } |I(v_i) - I(v_0)| \leq \Delta I_t\,. \\ 0, & \text{otherwise.} \end{cases}$$

    Given an intensity difference threshold ΔI_t, $\delta_{v_0,\Delta I_t}\,(v_i)$ measures the similarity in pixel intensity with respect to a reference pixel v_0.

13. Geometrical threshold g_t
    Geometric threshold is the maximum size allowed for $|USAN()|$. The maximum value of this threshold is bounded by the size of the circular mask *M()*. *Corner* sharpness is determined by this threshold [22].
    Domain
    $[0, |M(\,)|]$
    Typical values for g_t: $[15, 28]$.

14. Corner response $R(v_0)$
    Corner response is a number, that determines the capacity of the nucleus (v_0) to be detected as a corner. The higher the value of $R(v_0)$ the greater the possibility for v_0 to be a *corner* [22].

$$R(v_0) = \begin{cases} g_t - |USAN()| & \text{if } |USAN()| < g_t\,. \\ 0 & \text{otherwise.} \end{cases}$$

15. 1D Gaussian operator $G_{\Delta x,\sigma}\,(i)$
    $G_{\Delta x,\sigma}\,(i) : Z \to R$

$$G_{\Delta x,\sigma}\,(i) = \begin{cases} \frac{1}{2\pi\sigma^2} e^{-\left(\frac{i^2}{2\sigma^2}\right)}, & \text{if } -\Delta x \leq i \leq \Delta x\,. \\ 0, & \text{otherwise.} \end{cases}$$

References

[1] D. Holweg, A. Schilling and U. Jasnoch. *Mobile tourism new gis-based applications at the cebit fair*, Computer Graphik Topics, **15**(1), 17–19 (2003).

[2] P. Gros, O. Bournez and E. Bouyer. *Using geometric quasi-invariants to match and model images of line segments*, Technical Report, RR-2608, INRIA, July 1995.

[3] Anil K. Jain. *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[4] J. M. S. Prewitt. *Object enhancement and extraction*, Picture Processing and Psychopictorics, Academic Press, 1970.

[5] I. Sobel. *An isotropic* 3×3 *image gradient operator*, Machine Vision for Three-dimensional Scenes, Academic Press, 1990.

[6] D. Marr and E. C. Hildreth. *Theory of edge detection*, Proc. Roy. Soc. London, B-207:187–217 (nov 1980).

[7] J.F. Canny. *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Inteligence, **8**(6), 679–698 (nov 1986).

[8] R. M. Haralick. *Digital step edges from zero crossing of second directional derivatives*, IEEE Transactions on Pattern Analysis and Machine Inteligence, **6**(1), 58–68 (jan 1984).

[9] J. A. Noble. *Descriptions of Image Surfaces*, PhD thesis, Robotics Research Group, Department of Engineering Science, Oxford University, 1989.

[10] S. Venkatesch. *A study of Energy Based Models for the Detection and Classification of Image Features*, PhD thesis, Department of Computer Science, The University of Western Australia, 1990.

[11] H. Asada and M. Brady. *The curvature primal sketch*, IEEE Transactions on Pattern Analysis and Machine Inteligence, **8**(1), 2–14 (1986).

[12] H. Freeman and L. S. Davis. *A corner finding algorithm for chain code curves*, IEEE Transactions on Computers, 26, 297–303 (1977).

[13] P. R. Beaudet. *Rotational invariant image operators*, Int. Conference on Pattern Recognition, 579–583 (1978).

[14] H. P. Moravec. *Visual mapping by a robot rover*, Joint Conference on Artificial Intelligence, 598–600 (1979).

[15] L. Kitchen and A. Rosenfeld. *Gray-level corner detection*, Pattern Recognition Letters, 1, 95–102 (1982).

[16] C. G. Harris and M. Stephens. *A combined corner and edge detector*, 4th Alvey Vision Conference, 147–151 (1988).

[17] K. Rangarajan, M. Shah and D. V. Brackle. *Optimal corner detector*, Computer Vision, Graphics and Image Processing, 48, 230–245 (1989).

[18] F. Arrebola, A. Bandera, P. Camacho and F. Sandoval. *Corner detection by local histograms of contour chain code*, Electronic Letters, **33**(21), 1769–1771 (1997).

[19] F. Arrebola, P. Camacho, A. Bandera, and F. Sandoval. *Corner detection and curve representation by circular histograms of contour chain code*, Electronic Letters, **35**(13), 1065–1067 (1999).

[20] A. Quddus and M. Fahmy. *Fast wavelet-based corner detection technique*, Electronic Letters, **35**(4), 287–288 (1999).

[21] Z. Zheng, H. Wang, and E. Teoh. *Analysis of gray level corner detection*, Pattern Recognition Letters, 20, 149–162 (1999).

[22] S. M. SMITH and J. M. BRADY. *Susan - a new approach to low level image processing*, International Journal of Computer Vision, **23**(1), 45–78 (1997).

[23] D. Douglas and T. Peucker. *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, The Canadian Cartographer, **10**(2), 112–122 (1973).

[24] James D. Foley, Andries van Dam, Steven K. Feiner and John F. Hughes. *Computer Graphics, Principles and Practice*, Addison-Wesley Publishing Co., 2 edition, 1990.

[25] Milan Sonka, Vaclav Hlavac and Roger Boyle. *Image Processing, Analysis and Machine Vision*, Brooks/Cole Publishing Co., Pacific Grove, 1999.