THE SCENE UNDERSTANDING STAGE OF AN ARTIFICIAL VISION SYSTEM FOR MOBILE ROBOT NAVIGATION

T.Tambouratzis [1]
Institute of Informatics & Telecommunications, NRCPS "Demokritos", Greece.

Abstract

The scene understanding stage of a robotic navigation system is presented. The robotic navigation system aims at navigating through an area cluttered with objects from a known domain until it locates and reaches a known target. The scene understanding stage receives a line-drawing representation of the 3-D scene, which is initially encoded in an accurate, compact and uniform manner. Subsequently, region segmentation, interior/exterior line categorisation and inside/outside separation along contours of exterior lines are performed. Finally, an artificial neural network implementation of a combination of line-labelling schemes is executed; this outputs salient characteristics of the 3-D scene (surface background/foreground characterisation, orientation, depth, edge tilt) which constitute the key factors for correct robot navigation.

1. Introduction

This piece of research presents the scene understanding stage of an *artificial vision system* (AVS) for mobile robot navigation. The AVS constitutes part of the *robotic navigation system* (RNS), which is depicted in Figure 1.

2. The Robotic Navigation System (RNS)

The RNS aims at navigating in an intelligent, non-random manner through an area cluttered with objects from a known domain (e.g. objects found in a well-defined and constrained office or factory environment as in [1]) until it locates and reaches a known target.

The RNS consists of four stages, namely, the *image extraction* stage, the *image pre-processing* stage, the *scene understanding* stage and the *robot navigation* stage. These constitute a cycle which is repeated until the navigation task is complete, i.e. until the target has been reached.

2.1. The Robot Navigation Stage

The last stage of the RNS is focused entirely on the ***motion of the robot in search of the target.*** The robot is directed either through areas of the scene which are unoccupied by objects (if the target is not yet visible) or towards the target (if it is visible).

3. The Artificial Vision System (AVS)

The first three stages of the RNS constitute the AVS, which is responsible for capturing, analysing and understanding the current instance of the 3-D scene that the RNS is navigating through.

3.1. The Image Extraction Stage

A camera and a frame-grabber, both of which are attached to the robot, are employed in order to ***extract a 2-D grey-scale image of the 3-D viewed scene*** that the RNS needs to understand and navigate through. Good lighting conditions are required for an accurate image of the 3-D scene.

3.2. The Image Pre-Processing Stage

Edge extraction is initially performed; it is possible to use gradient-based oriented edge operators and subsequent thinning techniques [2-3]. Edge extraction not only reduces the captured 2-D grey-scale image into a binary image - whose black pixels probably correspond to parts of the edges - but also accomplishes an early segmentation of the image in terms of edges and regions. The edges can be *real* (if they arise from a discontinuity in depth between adjacent but non-abutting faces or from a discontinuity in orientation between adjacent abutting faces), or *virtual* (if they arise from a discontinuity in the intensity values of parts of the same face with different illuminations). Good lighting conditions promote the detection of real edges and eliminate virtual edges.

1 *Address for Correspondence*: Dr. T.Tambouratzis, Department of Mathematics, Agricultural University of Athens, Iera Odos 75, Athens 118 55, Greece, email tambourt@cyclades.nrcps.ariadne-t.gr.

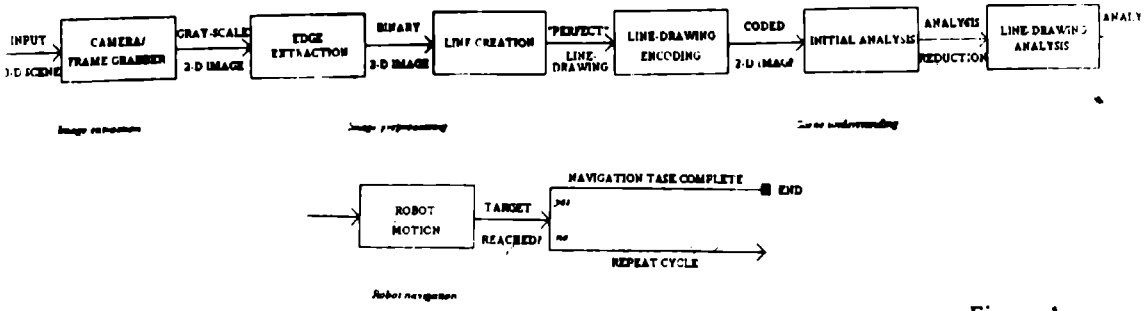


Figure 1

The binary image is then transformed into a line-drawing by *line creation*. The line-drawing representation of the 2-D image is required since it constitutes the starting point of many AI scene analysis labelling schemes; a combination of labelling schemes is used during the ensuing *scene understanding stage* of the AVS. Line segments of varying lengths are initially formed by joining black pixels (*linearity criterion*). Subsequently, these line segments are either extended and joined to create lines (*colinearity criterion*) and vertices (*intersection and angularity criteria*), or deleted (if *no evidence for their existence* is given from other line segments, e.g. if they appear to have arisen by noise).

Although generally inaccurate, line creation can be successfully performed for well-defined and constrained domains, since the distortions that occur during line-segment extension, joining and deletion are relatively limited and can, therefore, be rectified to conform with the geometric characteristics of the vertex-classes in the line-drawing (which correspond to the possible appearances of 3-D corners). For instance, line segments meeting at L-junctions with obtuse angles close to 180^0 can be smoothed to longer lines, accomplishing thus *economy of representation* and *noise resistance*.

Not all created lines coincide with edges of the 3-D scene; missing and superfluous lines occur. However, the end-product of line creation is a *"perfect" line-drawing*, i.e. a line-drawing where *each line terminates at a pair of vertices* (which belong to one of the possible vertex-classes) and *each vertex constitutes the end-point of at least two lines.*

3.3. The Scene Understanding Stage

Classification of the vertices to one of the vertex-classes of the domain to which the objects belong and *discrimination between the lines connected to each vertex* are initially performed. These two processes accomplish ***line-drawing encoding***, i.e. the accurate, compact and uniform representation of the main elements (lines and vertices) of the line-drawing of the 3-D scene. In more detail:

Vertex classification categorises the vertices of the line-drawing into one of the vertex-classes which are possible by the *domain specifications* and by the *visibility of the faces* forming the corresponding 3-D corner. The *geometrical characteristics* of the vertex-classes, i.e. the *angular relations between lines* as well as the *total number of connected lines*, are utilised for categorisation (for an application to the blocks-world domain see Figure 2a). Owing to the inherent inaccuracy of the image pre-processing stage, categorisation may be ambiguous (e.g. T-junctions and very obtuse Arrow-junctions and Fork-junctions). In such cases, ambiguity is offset by multiple categorisation which assigns more than one vertex-class to vertices which are easily confused.

Line discrimination assigns a uniform format to the lines connected to each vertex so as to render them distinguishable from each other. The *clockwise format* is employed here (for an application to the blocks-world domain see Figure 2a), which is *uniform* for all vertex-classes (no or little line reordering is required for multiple vertex categorisation) and *compatible with* the techniques of the ensuing *initial analysis*.

The two processes of vertex classification and line discrimination are concurrently performed during line-drawing encoding. A three-level hierarchy is employed which consists of:

(i) Two basic geometrical operators at the *lower level*, which provide information about line orientation.

The *segment operator* (shown in Figure 3a) is applied to every vertex, in turn, by creating a new co-ordinate system and assigning a number - from 0 to 7 - to each connected line, depending on the absolute orientation of the line. A 1-1 correspondence exists between the pair of new co-ordinate values of the line and its segment value (see Figure 3b).

- The *gradient operator*, which is equivalent to the slope of the line, is applied to each line.

The information provided by both operators is independent of the appearance (length or location) of each line in the line-drawing. Although similar, these operators differ dramatically in their degree of resolution, the latter being *more fine-grained* than the former and providing *qualitative rather than quantitative information.*

(ii) Three fundamental procedures at the *intermediate level.* These are composed of logical

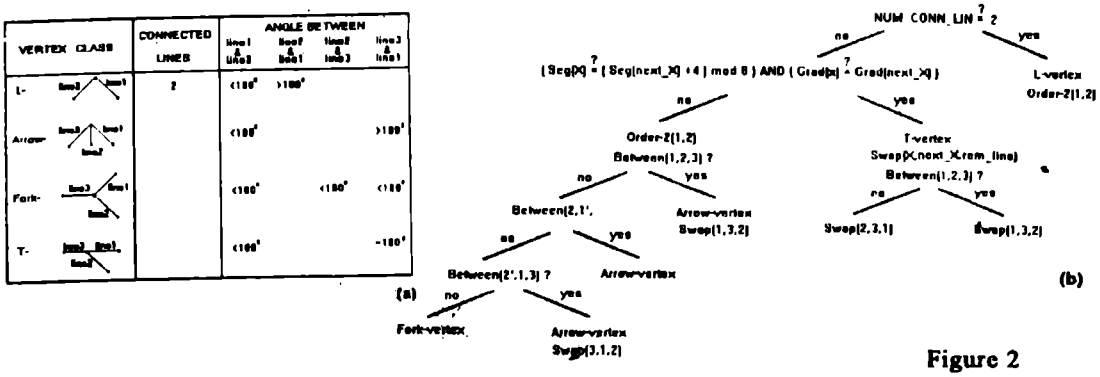


Figure 2

operations between the operators of the lower layer and determine the relative orientation of any number of lines with a common origin as well as the line reordering required to lead to the assignment of the clockwise format to any number of lines with a common origin.

- *SWAP (linei_1,linei_2,...,linei_n)* reorders *n* lines so that *linei_j* becomes ordered *j*th.
- *ORDER-2(line1, line2)* reorders a pair of lines with a common origin, so that the less than 180^0 angle lies between then when scanning clockwise from *line1* to *line2*.
- *BETWEEN(line1,line2,line3)* determines whether *line3* lies between *line1* and *line2* when scanning clockwise from *line1* to *line2*.

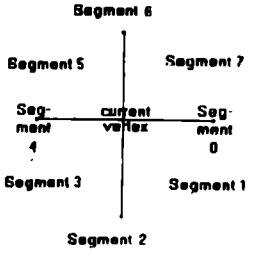
All three procedures are incremental and utilise the operators of the lower level in such a way that coarse-grained (qualitative) information is gathered first and, if this is not enough, more fine-grained (quantitative) measurements are made.

(iii) A *decision tree* at the *upper level*, which outputs the class of each vertex and a uniform order for the lines connected to it (as a means of discriminating between them). The decision tree needs to be constructed only once for each domain, but varies for different domains which allow for distinct vertex classes. However, the construction of the decision tree is straightforward, since the procedures of the intermediate level can be appropriately configured for any domain once a tabulation similar to the one of Figure 2a has been performed. Additionally, the decision tree is characterised by computational economy, as vertex classification and line discrimination are progressively realised, invariably utilising the least amount of information necessary.

Figure 2b demonstrates the decision tree, as this is constructed for the blocks-world domain; the allowable vertex classes and the desired order of the lines connected to each vertex class have been tabulated in Figure 2a. Vertex classification is delineated by the main branches of the decision tree, while the clockwise order of the connected lines is realised in the body of the terminal nodes.

During the ***initial analysis***, *region separation* is performed by employing a variant of the boundary stroll [4]. This is called the *clockwise stroll* and accomplishes full separation of each region of the image as well as of the background. Taking into account that any line of the line-drawing separates either two regions or one region and the background, each oriented line of the line-drawing (there exist two for each line) is directly adjacent to one of the two regions that the associated line separates. Under these assumptions, a number of clockwise strolls can expose all the regions in the line-drawing. The clockwise stroll is performed as follows:

A. Select the first oriented line of the clockwise stroll, add it to the region-separating path and mark its originating vertex (*vertex_origin*).
B. Go to the ending vertex of this oriented line and find the oriented line



(a)

SEGMENT VALUE	X-coordinate	Y-coordinate
0	positive	zero
1	positive	negative
2	zero	negative
3	negative	negative
4	negative	zero
5	negative	positive
6	zero	positive
7	positive	positive

(b)

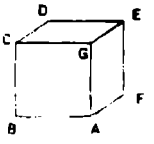
Figure 3

connected to it with **opposite direction** (*temp_line*).

C. Find the oriented line connected to the ending vertex which is (clockwise) ordered **cyclically next** to *temp_line* and add it to the region-separating **path.**

D. Repeat steps **B.** and **C.** until *vertex origin* **is** reached again.

Each region is defined by the path of oriented lines which are directly adjacent to it, while every oriented line is involved in the detection of exactly one region (or the background). The background is uncovered first by a clockwise stroll beginning from an extreme vertex of the line-drawing (e.g. the vertex with the maximum Y coordinate). Each of the other regions is uncovered by a clockwise stroll beginning from the originating vertex of one of the not yet traversed oriented lines; the process is invariable to the choice of the first oriented line. Region separation terminates when all oriented lines have been visited. The total number of performed strolls equals the number of regions in the line-drawing, including the background. It should be mentioned that pairs of opposite-direction oriented lines never belong to the same region. Figure 4 illustrates the process of region separation through four clockwise strolls, as well as the originating oriented lines and the oriented lines traversed during each stroll.



REGION #	0	1	2	3
originating line	FF	EG	DC	GC
	FA	GA	CG	CD
	AD	AF	GE	BA
	BC	FE	ED	AG
	CD			
	DE			

Figure 4

Next, all lines in the line-drawing are partitioned into exterior and interior lines, whereas the inside/outside relations along contours of exterior lines are determined [5-6] using a combination and refinement of the boundary stroll [4] and the enhancement technique [7]. By fully specifying the boundary lines separating the objects from the background, the complexity of the ensuing *line-drawing analysis* is dramatically reduced. For a more detailed description of this part of the initial analysis, the reader is referred to [5] and [6].

Line-drawing analysis receives the line-drawing as well as the product of the initial analysis and employs an *artificial neural network* implementation [8] of a combination of *line-labelling schemes* [9-11] in order to output the basic characteristics of the 3-D scene. Among the *extracted characteristics* are the background/foreground nature of the surfaces, the surface orientation, the surface depth relative to other adjacent surfaces as well as the edges' tilt [11] and nature (boundary/convex/concave) [9-10]. All these characteristics constitute key factors for correct robot navigation. For a more detailed description of the line-drawing analysis, the reader is referred to [8] for the artificial neural network implementation and to [9-10] and [11] for the labelling schemes utilised here.

4. References

1 Tsukiyama, T. Huang, T.S., (1987). Motion stereo for robot navigation of autonomous vehicles in man-made environments, *Pattern Recognition* **20**, pp. 105-113.

2 Rosenfeld, A., Ornelas, J. Hung, Y., (1988). Hough transform algorithms for mesh connected SIMD parallel processors, *Computer Vision, Graphics and Image Processing* **41**, pp. 293-305.

3 Li, Z.N., (1994). Stereo correspondence based on line matching in Hough space using dynamic programming, *IEEE Transactions on System, Man and Cybernetics* **24**, pp. 144-152.

4 Winston, P.H., (1984). **"Artificial Intelligence"**, Addison-Welsey, Cambridge, MA.

5. Tambouratzis, T. (1993). Object segmentation in line- drawings combining approaches from AI and psychophysics, Proceedings of the **"ACCV 1993"**, Osaka, Japan, November 23-25, 1993, pp. 438-441.

6. Tambouratzis, T., (1995). Integrating artificial and psychophysical approaches for boundary finding in line-drawings, *International Journal of Intelligent Systems* **10**, pp. 443-457.

7 Walters, D., (1987). Selection of image primitives for general purpose visual processing, *Computer Vision, Graphics and Image Processing* **37**, pp. 261-298.

8. Tambouratzis, T., (1991). An implementation of a harmony theory network for interpreting line-drawings, *Network: Computation in Neural Systems* **2**, pp. 443-454.

9 Huffman, D.A., (1971). Impossible objects as nonsense sentences, *Machine Intelligence* **6**, pp. 295-323.

10. Clowes, M.B., (1971). On seeing things, *Artificial Intelligence* **2**, pp. 79-116.

11 Tambouratzis, T., (1993). Tilt: a line-labelling scheme for depth and orientation information, in Proceedings of the **"MCPA-93"**, Halmstad, Sweden, June 1-3, 1993, pp. 123-130.