USING LINE SCAN CAMERA FOR MATERIAL CUTTING

NICK ANDREI IVANESCU, THEODOR BORANGIU
Dept. of Computer-Based Process Control
University Politehnica of Bucharest
313, Spl. Independentei, Bucharest
ROMANIA

Abstract: The paper describes an intelligent manufacturing control system, designed to optimise flat material cutting from visually inspected scenes. A vision processor reconstructs the 2D image of the material, moving on a linear conveyor, from a dual line scan camera. A number of image features are extracted to identify internal defects. A skeletonization algorithm produces the binary representation of the useful surface to be manufactured. The vision system generates the maximal number of user-defined patterns that optimally cover the useful surface. The vision pattern data is converted to path specification to command the pattern cutting trajectories for a SCARA robot.

Keywords: Manufacturing Systems, Robot Vision, Backtracking, Image Processing, Computer Integrated Manufacturing.

1 Introduction

The current trend in nowadays development of artificial vision systems is their increasing dedication to applications. This trend was a logic consequence of accepting that, as the complete understanding of the mechanisms of human and artificial vision might still necessitate an important number of years, designing specialised, application-oriented machine vision represents the for the moment the best solution to develop the field.

The most important direction in the area of potential applications of artificial vision in industry is actually provided by real-time inspection and measurement tasks. Examples of such applications are: checking the presence of objects or object features, estimating their location, inspecting relative poses of assembly components, examining part geometry and material surfaces.

The generic name of such tasks is Automated Visual Inspection (AVI). To be implemented, they impose a feature-based description of materials and parts, which became an attribute of flexibility in computer manufacturing. In high-technology automotive or microelectronics industries, CIM architectures integrate visually guided robots, leading thus to another class of generic tasks: Robot Vision Guidance (RV).

To perform connected AVI and RV tasks, visual servo architectures were designed, by classifying robot-vision systems according to the type of *structure* of the closed-loop motion controller and to its type of *control law*:

- *Hierarchical motion control* structures (pipelined *dynamic look-and-move* schemes) vs. *direct motion control* structures (direct *visual servo* schemes).
- *Position-based control* structures (using error signals defined in task space coordinates) vs.. *image-based control* structures (using error signals defined directly in terms of image features extracted from object's images).

Most of the actual limits of connected AVI-RV systems are due to image acquisition and low level processing. Thus, usually the optics and lighting devices are custom designed almost for any particular application. Then, the parallel image processing hardware cannot be fully exploited for handling the continuously increasing dynamics of industrial applications (600-2000 object/minute in discrete batch processes), because of the limitations in picture snap rates. One envisaged solution to enhance the applicability of artificial vision to manufacturing is the development of specific *vision engineering* concepts and tools.

The class of artificial systems dedicated to *mobile scenes* is frequently used in mixed AVI-RV applications. At the level of the robot – partner of the machine vision processor, such applications are defined either as "pick-on-the-fly" or "consumer of vision data" tasks, executed at run time.

2 System architecture and 2D image reconstruction from Line Scan Camera data

The robot-vision system dedicated for visual inspection of flat materials (leather surfaces, metallic or glass sheets) and motion planning for multi-purpose robotised processing (pattern drawing or cutting) is based on a multiprocessor pipelined architecture with the following resources:

- Image acquisition is performed by a dual-camera system composed by two DALSA Line Scan cameras (LSC) with line resolution of 2048×1 pixels, each pixel being represented on 8 bits.
- Materials are travelling on a conveyor belt, the motion of which is estimated by a LENORD encoder having the resolution of 1024 pulses/rotation. The displacement pulses are input to an ADVANTECH I/O counter-timer board providing the feedback displacement data for the Central Management Subsystem (CMS), which triggers the image line snap compensating for speed variations in the belt motion.
- The scene illumination is provided by linear neon tubes operated at 100 Hz by a MERCRON lighting controller, which equalises the light intensity over the total length of the neon sources. The tubes are placed at 20 cm above the conveyor belt.
- A CORECO frame grabber and image processor stores the image lines from the two cameras and performs low level processing upon the 2D reconstructed image of the inspected material. The image processor, I/O counter-timer unit and CMS processor are integrated in an industrial, dedicated IBM computer providing also the man-machine interface at global application level.
- An ADEPT Cobra 600TT robot system is connected via a serial communication line to the CMS processor, from which it receives the point – and path data to move its end-effector along the closed contours of the patterns optimally generated to cover the maximum useful area of the material's surface (Fig. 1).

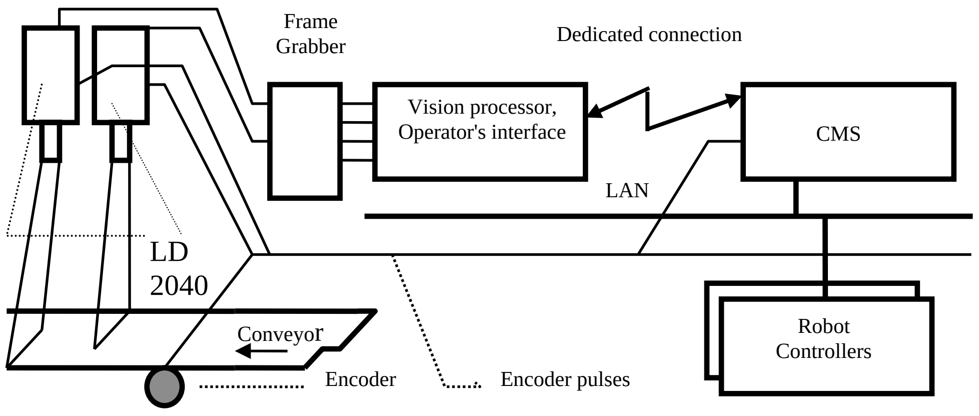


Fig. 1 Dual-LSC robot-vision system for inspecting materials on moving scenes.

The dual-camera configuration was adopted for two main reasons: high-precision inspection requirements and significant widths of materials along the direction of the image lines (belt width). One important aspect considered is that, due to the big opening of the cameras objectives, the acquired images present important distortions along the direction of line acquisition from the sensors.

The control system was designed as a *multi-robot* structure. This characteristic might be useful when either the workspace of one single robot manipulator is insufficient to cover the entire belt width, or the subsequent pattern cutting cycle of one single robot is incompatible with the high rate of material arrival on the conveyor belt. The vision processor performs high-level image processing for surface inspection, defect detection, skeletonization, and optimal placement of patterns on the valid surface of the material. The mapping of motion data in robot base coordinates, and the communication with the robot controller are performed by the CMS unit. All this computation is executed while the material is moving in front of the LSC, and is finished before the material reaches the downstream placed robot workstation.

The image acquisition model for the dual-LSC system, shown in Fig. 2, has three particularities:

- image *distortions* are present only along the direction of the camera acquisition line [3];
- there is a zone of *superposed* images for the two cameras: the end of the acquisition line from camera 1 is significantly superposed over the beginning of the acquisition line from camera 2;
- there is a zone of *spatially shifted* images for the two cameras: the image from camera 2 is anticipated with a fixed number of lines with respect to the image from camera 2.

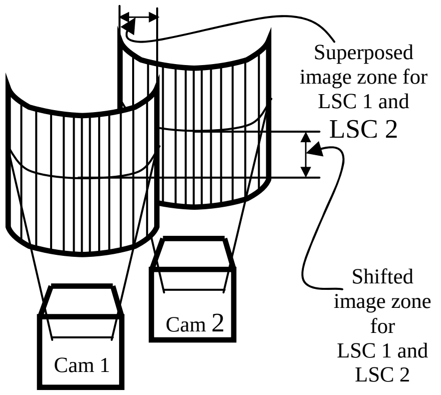

Superposed image zone for LSC 1 and LSC 2
Shifted image zone for LSC 1 and LSC 2
Cam 2
Cam 1

Fig. 2 The image acquisition model

To extract and further compensate for these model particularities, a *camera-scene calibration* procedure was designed, which uses a calibration device. This device is a pattern represented by a set of elements of known shapes, dimensions and positions in the real world of millimetres. The usage of this calibration device during the camera installation stage allows also an exact *parallelism* of acquisition lines of all cameras, also *perpendicular* to the belt direction.

The calibration pattern is symmetric along the direction of the conveyor belt, and contains a number of dark rectangular *blobs* on a light background, to estimate the model of the scene along the distortion direction.

The two cameras have their acquisition lines reciprocally offset along the belt's direction. LSC 1 has its acquisition line positioned on the upper edge of the inferior *blob row*, whereas LSC 2 has its acquisition line positioned on the lower edge of the superior *blob row*. The following steps are carried out in the camera-scene calibration sequence:

1. The calibration pattern is placed in the plane of the conveyor belt, with its two blob rows normally to the direction of motion of the belt.
2. The position of each camera is adjusted with respect to the conveyor, by help of its 3-d.o.f rotation mobility (see Fig. 3):
   - The *roll* angle about the *Z* axis, normal to the belt plane, is first adjusted; this provides the parallelism (and horizontality) of the LSC acquisition lines. The proper roll orientation is achieved when only odd blobs are identified in the image.
   - The *pitch* angle is next adjusted about the *X* axis, normal to the belt's motion direction, to position the acquisition lines. The proper pitch orientation is achieved when the offset between the acquisition lines of the two LSC equals the dimension of the free space between the superior and inferior row of blobs, and all even and odd blobs are observed.
   - The *rotation about the belt motion axis*, *Y*, adjusts the value of the superposition zone between the two LSC images. The angles are adjusted until the first (last) blob, at the left limit (right limit) of the belt, is identified at the start (end) of its line in the LSC 1 (2) image
3. The *focus* and the *aperture* are adjusted for each objective, until the best *contrast* (computed from black-white transitions in the grey level histogram of an image line) and *luminosity* are obtained from the image of a calibration sheet.
4. The calibration function of the vision system is finally activated, for the automated computation of the *scene parameters* from the measured values of the calibration pattern.

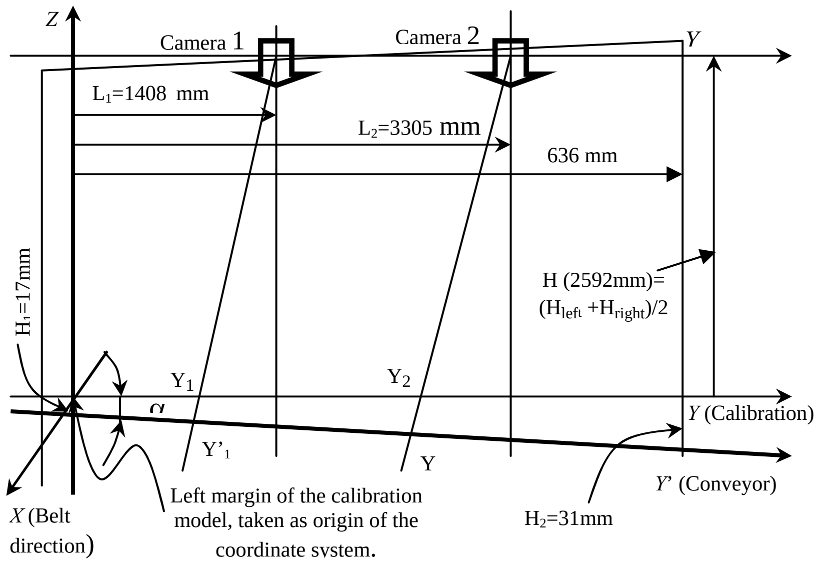


Fig. 3 Geometric features of the image acquisition system.

As can be observed from Fig. 3, there might exist a horizontality error in the placement of the calibration device, whenever the belt's width has important values. Due to this fact, the parallelism errors between the *OY* axis (the axis of the calibration device along the belt's width) and the *O'Y'* axis, corresponding to the conveyor width, must be considered [5].

To estimate the parameters of the scene (of the conveyor), the system uses the mapping equations from the calibration coordinates, *OY*, to the scene coordinates, *O'Y'*:

$$tg\alpha = \frac{H_2 - H_1}{L} \qquad (1)$$

For line scan camera number 1:

$$Y'_1 = \frac{Y_1 \cdot H\cos\alpha - L_1(H_1 + Y_1\sin\alpha)}{H - H_1 - Y_1\sin\alpha} \qquad (2)$$

For line scan camera number 2:

$$Y'_2 = \frac{Y_2 \cdot H\cos\alpha - L_2(H_1 + Y_1\sin\alpha)}{H - H_1 - Y_1\sin\alpha} \qquad (3)$$

The scene parameters, obtained with the off line camera-scene calibration sequence, were used to map the location of the covering patterns, optimally generated by the vision system at run time, in the robot base coordinates $(x_{0,} y_0, z_0)$.

The final step of the camera-scene calibration consisted into adjusting the acquisition line scan rate according to the belt's speed. This is a 2-stage process:

- First, a *circular calibration disk* of known radius was placed in the field of view of the camera, and the scan rate was varied in a loop-program, until the error between the maximal dimensions in pixels along the *X* and *Y* axes was about zero. The rations in [pixel/millimetre] along the *X* and *Y* directions were also computed for later coordinate conversions.
- Then, the number of corresponding *encoder pulses* n_{enc} was read for the obtained acquisition rate. This value was stored to be further used at run time as feedback signal for line acquisition triggering, irrespective of the belt speed variations.

3 Pattern generation for maximal covering of skeletonized material surfaces

Once the 2D grey level image of materials reconstructed by successive line acquisition, and stored in the computer's memory, several transformation processes are performed upon this image.

The material's image is first *binarised* with two thresholds, thr_L, thr_H, $thr_L < thr_H$. The binary value $f^b_{i,j}$ for each pixel (i, j) of the 2D image with grey levels $f^*_{i,j}$ is computed as follows

$$f^{\mathrm{b}}_{i,j} = \begin{cases} 0, & f^*_{i,j} \in [0, thr_{\mathrm{L}}) \cup (thr_{\mathrm{H}}, 255] \\ 1, & f^*_{i,j} \in [thr_{\mathrm{L}} \cup thr_{\mathrm{H}}] \end{cases} \quad (4)$$

The two thresholds are either automatically computed (the system uses the maximal contrast curve in the grey level image), or user defined, according to a try-and-use preliminary search for each particular class of materials (metallic, leather etc). Hence, any pixel on the material's surface having the colour outside the range of levels $[thr_{\mathrm{L}}, thr_{\mathrm{H}}]$ will be represented as a black, *foreground* pixel, whereas all remaining pixels of the material, with colours inside the specified range, will create the white, *background* representation of the material's surface to be optimally covered by patterns [2].

Internal defects of materials (holes, dark areas and tree-shaped scratches or incisions) are identified as *blobs*, i.e. connected zones of pixels having the same colour (the 3×3 neighbourhood of 8 surrounding pixels is taken for each one). Very small blobs are filtered out; the remaining ones are queued in a *list of blobs* of the *segmented image*. Blob detection is performed by a "fill" type algorithm, which marks foreground pixels after their inspection column-by-column and line-by-line.

Whenever an unmarked (background) pixel is found, the program starts to explore a new blob. Every found "defect"- blob is assigned a new ID value in the blob queue of the image. During the blob search process, other blob features are calculated in parallel: area, perimeter, centre of mass, orientation, minimum rectangle box, which may be further used for analysis and statistics reports [4].

Usually, peripheral material zones (less than d_{ext} from its outer contour and less than d_{int} from the blob contours) must be eliminated from further processing, due to quality problems. This reduction of material surface is obtained by *expanding* the contours of the defect blobs (marking its exterior pixels closer than d_{int} with the blob's colour), respectively by *contracting* the perimeter of the material (marking its interior pixels closer than d_{ext} with the background colour).

The next algorithm was used:

1. Save the coordinates of the *first pixel found* during blob detection, part of its frontier.
2. Create and store a *list of all contour pixels,* starting from this pixel and visiting its neighbours
3. Repeat steps 1 and 2 d_{ext}, respectively d_{int} times for *material contracting* and *defects expanding*.
4. *Mark the pixels* from the list respectively with the background/defect blob colour (black).
5. Create a *new list of contour pixels* from the new frontiers.

The efficiency of the method is visible; only the pixels that have to be modified are traversed and the dimension of the object's frontier only determines the supplementary memory to be used [1].

A solution combining the *Greedy* and *Backtracking* techniques, rather than a polynomial algorithm was chosen to solve for the optimal covering of the reduced material surface with rectangular patterns, of user defined dimensions. Backtracking was used as one of the most known methods for solving problems for which the solution is represented as an array $x = (x1 \;\; ... \;\; xn) \in A1 \times ... \times An$, where Ai are finite sets. Moreover, it is necessary that the solution $(x1,..., xn)$ satisfy some internal constraints, which in this case apply to the rectangle patterns as follows:

- all the points belonging to the rectangles must be not marked (not present in the defect blob list);
- once a rectangle is determined, its area is subtracted from the material's area to be used;
- the edges of the rectangles are parallel to the axes x_{vis}, y_{vis} of the image plane.

The function verifying these conditions returns a Boolean value, which, if true, causes the execution of the following sequence of steps, performed by the ***back*** (***int*** *Level*) function:

1. A matrix called m_modBuf is filled with zeros in the area occupied by the rectangle. Matrix pixels are accessed with m_modBuf+(line-1)*image_length+column.
2. The x, y coordinates of the upper left and lower right corners of each new found rectangle will be stored in an array called *rectangle*.
3. The function ***back*** is called again with incremented argument *Level+1*, trying to find the x, y coordinates of rectangle number *Level+1* (see Fig. 4).

The set $A = A1\times ...\times An$ is the space of all possible solutions. The elements $x \in A$ satisfying the internal constraints are the *result solutions*, as arrays containing the corner's coordinates for the set of n

found rectangles, which cover a greater material surface as compared to the previous solution.

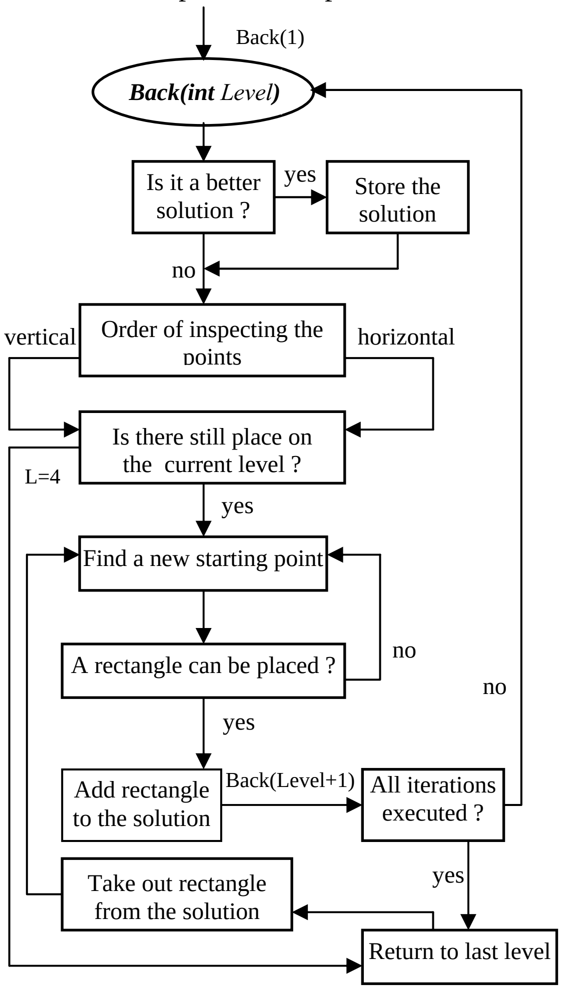

Back(1)
Back(int Level)
Is it a better solution ?
yes
Store the solution
no
vertical
Order of inspecting the points
horizontal
Is there still place on the current level ?
L=4
yes
Find a new starting point
A rectangle can be placed ?
no
no
yes
Add rectangle to the solution
Back(Level+1)
All iterations executed ?
yes
Take out rectangle from the solution
Return to last level

Fig. 5 The Backtrack-Greedy algorithm for optimal surface covering

Normally, when using the backtracking method, the interest is to determine *all* the result solutions, eventually for selecting among them that one which maximizes a *cost function*. For the current application, the cost function is the useful material surface covered by rectangular patterns.

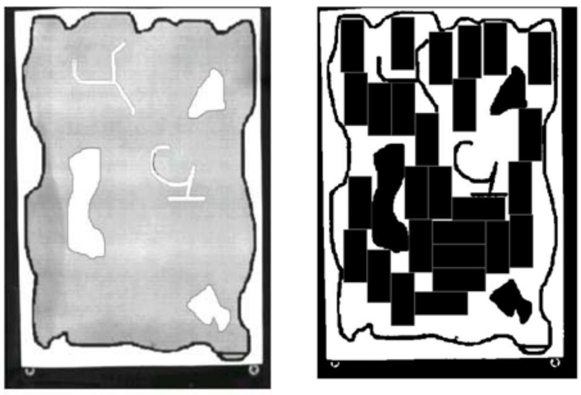
The method avoids the generation of all possible result solutions (elements of the cross product A). A value is assigned to element $xk \in Ak,\ k = n,...,1$ only after values were assigned to $x1 \in A1, \ldots, x(k-1) \in A(k-1)$. The method continues to assign a value to $x(k+1) \in A(k+1)$ only if xk together with $x1,...x(k-1)$ verify the *continuation condition* $Ck(x1,..,xk)$. If this condition is not satisfied, then another choice must be made for $xk \in Ak$, which is possible unless all possible values in the set Ak were exhausted. If the set Ak was exhausted, it is necessary to reduce k to $k-1$ and to select another result solution for the element $x(k-1) \in A(k-1)$. This happens when there is no more space left to place a rectangle, irrespective of its orientation. The search for solutions is terminated when the maximal number of iterations, experimentally defined, is reached.

To optimise the *Backtracking* algorithm, the *Greedy* programming technique was used. This method applies to problems for which a set A of n elements is given, and a subset $B \in A$ must be determined, meeting certain conditions to be accepted. Because more than one solution may exist, a *selection criterion* is specified to provide in such cases the unique, *optimal solution* from the possible ones. If B is a *possible solution* and $C \in B$, then C is also a possible solution.

Using the camera-scene calibration parameters, the rectangle coordinates are mapped to the robot base frame (x_0, y_0, z_0), and sent to the robot controller via a dedicated communication line.

4 Experimental results

The dual-LSC robot-vision system was implemented, calibrated and tested in the Robotics and CIM laboratory of the University Politehnica in Bucharest. The application consisted in inspecting leather pieces of about $0.4\,\text{m}^2$, which randomly shaped and located holes and incisions. Rectangle patterns were placed on the materials after defect detection and surface reduction (Fig. 6).



Grey scale input image of sample leather material

Rectangles patterns 90×40, 50 iterations

Fig. 6. Binary image of sample, after pattern covering.

A series of experiments was carried out, respectively with rectangular patterns of dimensions [cm]:

D_1=30 x 50, D_2=50 x 70, D_3=70 x 90, D_4=110 x 130

For each type of pattern, the Backtracking algorithm was run with $50, 300, 550, 800$ and 1050 iterations. The percentage p[%] of the accepted surface, covered by rectangle patterns, is indicated in Table 1, where

$$p = \frac{100 \cdot D_i \cdot pattern\_no}{\text{Re}duced\_backgrd\_area - Expanded\_blob\_area}$$

Table 1 Percentage of surface covering with the *Backtracking – Greeding* procedure

Pattern dimensions		30×50	50×70	70×90	90×110	110×130
Number of iterations	1050	91.2	80	73.6	66.6	55.5
	800	88.7	77.1	68.4	58.3	55.5
	550	88.7	77.1	68.4	58.3	55.5
	300	88.7	74.2	68.4	50	44.4
	50	87.5	71.4	68.4	50	44.4

As can be observed from Table 1, the number of covering patterns increases as the number of iterations augments, and/or the dimensions of the rectangle's edges become smaller. However, over 300 iterations, the result solutions do not improve significantly. For 300 iterations the execution time was of 4.4 seconds, which did not overrun the 6.2 seconds travelling time of the material from the vision inspection station to the robot workstation.

References

[1] X1. Ahuya, N., A Transform for Multiscale Image Segmentation by Integrated Edge and Region Detection, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 6, 1996, pp. 20-25

[2] X2. Borangiu, Th. and M. Dupas, *Robot Vision*, Bucharest, Academic Press, 2001.

[3] X3. Flusser, J. and T. Suk, Degraded Image Analysis: An Invariant Approach, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, 1998.

[4] X4. Lee, C. and D.A. Landgrebe, Feature Extraction Based on Decision Boundaries, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 2, 1993.

[5] X5. Negahdaripour, S., Integration of Radiometric and Geometric Cues for Dynamic Scene Analysis, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol.20, 1998.