ICPRAM 2012

1st International Conference on Pattern Recognition Applications and Methods

Proceedings

Volume 2

Vilamoura, Algarve, Portugal
6 - 8 February, 2012

Sponsored by:

INSTICC

Technically Co-sponsored by:

AERFAI
Asociación Española de Reconocimiento de Formas y Análisis de Imágenes

APRP

IEEE Signal Processing Society

IEEE

In Cooperation with:

aaai

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

VISUAL NAVIGATION FOR THE BLIND
Path and Obstacle Detection

J. José, J. M. H. du Buf and J. M. F. Rodrigues
Vision Laboratory, Institute for Systems and Robotics (ISR/IST)
University of the Algarve (FCT and ISE), 8005-139 Faro, Portugal
{*jjose, dubuf, jrodrig*}*@ualg.pt*

Keywords: Path detection, Obstacle detection, Obstacle avoidance.

Abstract: We present a real-time vision system to assist blind and visually impaired persons. This system complements the white cane, and it can be used both indoor and outdoor. It detects borders of paths and corridors, obstacles within the borders, and it provides guidance for centering and obstacle avoidance. Typical obstacles are backpacks, trash cans, trees, light poles, holes, branches, stones and other objects at a distance of 2 to 5 meters from the camera position. Walkable paths are detected by edges and an adapted Hough transform. Obstacles are detected by a combination of three algorithms: zero crossings of derivatives, histograms of binary edges, and Laws' texture masks.

1 INTRODUCTION

Blind and visually impaired persons need to navigate using the cane or, at best, an ultrasonic obstacle detector. There are an estimated 180 million persons with severe visual impairments of which 40-50 million are completely blind, and every year 2 million more become blind. The Portuguese project "Blavigator: a cheap and reliable navigation aid for the blind" aims at developing a cheap portable GIS and GPS-based navigation aid for the blind, for both outdoor and indoor navigation, with vision modules for obstacle avoidance and object recognition.

There are a few recent systems for visually impaired users which may assist them in navigating, with and without obstacle detection and avoidance. One system integrates outdoor navigation with obstacle detection (Lee et al., 2008). Another is the electronic travel aid called iSONIC (Kim et al., 2009). The latter complements the conventional cane by detecting obstacles at head height. (CASBliP, 2009) was an EU-funded project. The main aim was to develop a system capable of interpreting and managing real-world information from different sources in order to improve autonomous mobility. For a detailed description of the state-of-the-art see (du Buf et al., 2011).

The work presented here concerns one of the vision modules of the Blavigator project. It uses a stereo camera fixed to the chest of the blind, at a height of about 1.5 m from the ground. Results presented are obtained by using only the right-side camera, and the system performs equally well using a normal, inexpensive webcam with about the same resolution. The resolution must be sufficient to resolve textures of pavements and potential obstacles like holes with a minimum size of about 10 cm at a distance of 2 to 5 meters from the camera. Our prototype works in real-time on a normal netbook computer and is able to detect borders of paths like outdoor sidewalks and indoor corridors, and it can assist the user in navigating inside the borders. It also detects obstacles, both indoor (trash cans, plant pots, backpacks and furniture) and outdoor (light poles, tree branches, holes and loose stones) inside the borders for alerting the user and avoiding them.

2 PATH DETECTION

Paths are the areas where the user can walk in a corridor or on a sidewalk. These are delimited by left and right border lines. The position where these lines intersect is called the Vanishing Point (VP). Using the Canny edge detector (Canny, 1986) in combination with an adapted version of the Hough transform (Duda and Hart, 1972), it is possible to detect the borders and the VP.

Let $I_{\mathrm{I}}(x,y)$ be an input frame with fixed width W_{I} and height H_{I}. Because of perspective projection, the left and right path borders intersect at the vanishing

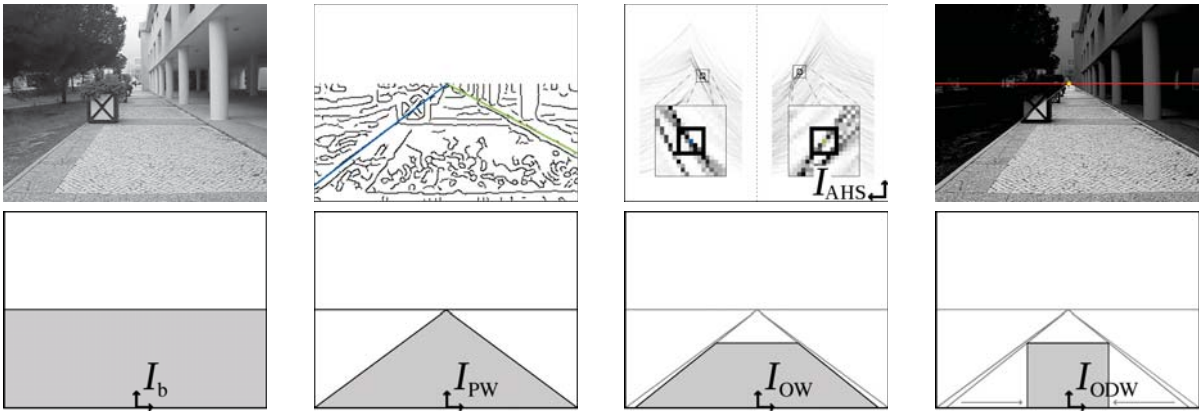

Figure 1: Top row, from left: one input frame, its edge map I_b with left and right borders in blue and green, the corresponding AHS with magnified regions, and detected borders highlighted with the vanishing point VP (yellow circle) and the horizon line HL (red). Bottom row: the different path and obstacle windows I_b, PW, OW and ODW.

point; see the yellow circle in the top-right image of Fig. 1. If the camera plane is exactly in the vertical position, the horizon line HL (the red line in the same image) is close to the middle of the frame. Since vertical camera alignment is not fixed but varies over time when the user walks, we use the VP in order to determine the line: $y_{HL} = y_{VP}$. During an initialization phase, i.e., the first 5 frames which last less than one second, $y_{HL} = H_I/2$ is taken, after which the height of HL is dynamically computed for each new frame by averaging the heights of the previous 5 frames of which the VP is known. The lower part of the image, below HL, will contain path borders and obstacles on the ground. This part, which is illustrated in Fig. 1 (bottom-left), will be analyzed.

In order to reduce CPU time, only grayscale information is processed after resizing the lower part to a width of 300 pixels while maintaining the aspect ratio. Then two iterations of a smoothing filter are applied in order to suppress noise. The Canny edge detector is applied with $\sigma = 1.0$, $T_l = 0.25$ and $T_h = 0.5$. The result is a binary edge image $I_b(x,y)$ of width $W_b = 300$ and height $H_b = y_{VP}(W_b/W_I)$, with $x \in [-W_b/2, W_b/2-1]$ and $y \in [0, H_b-1]$, and the origin of the coordinate system at the bottom-center of the image. One path frame and I_b are shown in the leftmost images of Fig. 1.

We then use the Hough transform to search for border candidates in the left and right halves of image I_b. As we want to check straight lines in both halves using polar coordinates, the Hough transform is applied to I_b, yielding the adapted Hough space $I_{AHS}(\rho,\theta)$: we restrict the Hough space to $\theta \in [20^o, 69^o] \cup [111^o, 160^o]$ such that vertical and almost vertical lines in the intervals $\theta \in [0^o, 20^o] \cup [160^o, 180^o]$ are ignored. The same is done for horizontal and almost horizontal lines ($\theta \in [70^o, 110^o]$).

For maximizing the number of searched lines and minimizing the computation time, we use $\Delta\theta = 0.5^o$. We use $\Delta\rho = \cos\theta$ for $\theta \in [20^o, 44^o]$ and $\Delta\rho = \sin\theta$ for $\theta \in [45^o, 69^o]$. Lines outside the image can be discarded, so the biggest value of ρ occurs for the straight lines that pass through the opposite corners relative to the bottom-center of the image ($\rho \in [0, (W_b/2)\cos\theta + H_b \sin\theta]$).

For optimization purposes, we can compute the lines for the Hough transform by "mirroring" the left and right lines about the y axis: we calculate the lines for $\theta \in [20^o, 69^o]$ and, at the same time, the lines for $\theta \in [111^o, 160^o]$, as explained below. The right border is denoted by $L_{\rho,\theta}(x_r, y_r)$ and the left one by $L_{\rho,\pi-\theta}(x_l, y_l)$. For $L_{\rho,\theta}$ we use $x_r = (\rho - y_r \sin\theta)/\cos\theta$ and $y_r = (\rho - x_r \cos\theta)/\sin\theta$. For $L_{\rho,\pi-\theta}$ we use $x_l = -x_r - 1$ and $y_l = y_r$. This results in a reduction of CPU time of about 50%, as we need little more than half of all computations involved in building the normal Hough space.

The I_{AHS} space is filled by checking the pixels in I_b from top to bottom: left-to-right for the right border ($L_{\rho,\theta}$) and right-to-left for the left border ($L_{\rho,\pi-\theta}$). As for the normal Hough space, I_{AHS} is a histogram which is used to count the co-occurrences of aligned pixels in the binary edge map I_b. However, longer sequences of edge pixels count more than short sequences or not-connected edge pixels. A run of n connected edge pixels is counted by applying $P_n = P_{n-1} + 2$ with $P_1 = 1$, and will contribute n^2 to the relevant I_{AHS} bin. The final value of an $I_{AHS}(\rho,\theta)$ bin is the sum of the P_n values of all sequences of ON pixels, each sequence having at least one ON pixel. The AHS is shown in Fig. 1, 1st row 3rd column, with the detected maxima in blue and green. The corresponding borders are also shown colored in the edge map (Fig. 1, 2nd column at top).

For selecting the path borders we analyze the successive highest values of I_{AHS} for each frame, in each half, starting with the highest ones. This results in an intersection point. If the intersection point of the next left and/or right value(s) has a smaller Euclidian distance to the VP of the previous frame, we continue checking the next value(s) on the corresponding side(s). If there is no pair with a smaller distance, the current value is selected. After both values are selected, their intersection corresponds to the frame's VP. In this search, all combinations of left and right border candidates are considered. If in the left or right regions where the I_{AHS} values are checked there is no maximum which corresponds to at least one sequence of not less than 10 connected ON pixels, it is checked again without region restrictions. If still no correspondence can be found, the border is considered not found for that side. In this case, the average of the last 5 borders found is used. If after 5 consecutive frames one or both borders are not found, the user is warned. The HL value remains the same, and the scanned window expands to the left- and/or rightmost column according to the missing border(s). In such a case the user knows that the vision module temporarily cannot detect a valid path, but he can continue walking using the white cane, and the module will still check for obstacles in front. Below, the walkable path is called the Path Window (PW). It consists of the triangle delimited by the left and right borders in I_{b}, as shown in the second-from-left images in Fig. 1.

3 OBSTACLE DETECTION

The PW is wider than the area in front where the blind person will walk and where obstacles must be detected. Hence, the PW is narrowed by drawing new lines through the VP: the positions of the original borders of the PW at the bottom line are shifted right (left border) and left (right border), by 5% of W_{b}, which results in a narrower triangle. In addition, the height of the window can be reduced because the top of the triangle (near VP) is too far away. Hence, the height is redefined for a corresponding range of at least 5 meters from the user, i.e., $H_{\mathrm{OW}} = 2/3 \times y_{\mathrm{VP}}$, which yields a trapezoid with parallel top and bottom lines. This is called the Obstacle Window (OW, I_{OW}); see Fig. 1, 2nd row 3rd column, and Fig. 2, the bright area in the top-left image.

For obstacles in the immediate neighborhood beyond the reach of the white cane we have to consider distances between 2 and 5 m in front of the user, taking into account the height of the camera and perspective projection of the lens. However, the resolution at the bottom of the image is higher than that at the top or even at the VP. Therefore, we define a new Obstacle Detection Window (ODW) and use interpolation to correct image resolution; see Fig. 1 (bottom-right); also Fig. 2, the 2nd and 4th images from left on the top row in the case of the OW in the frames to their left. Hence, the trapezoidal OW is converted to the rectangular ODW by maintaining the resolution at the top of OW but reducing it at the bottom.

At this point we must stress that our system will not detect obstacles at a distance of less than 2 m from the user, because of two reasons: (i) The user has already been alerted to a looming obstacle at a larger distance and advised to adapt path trajectory. (ii) The user will always check a detected obstacle using the white cane at short distance.

To the ODW region we also apply a lowpass filter with a 3×3 kernel in order to suppress noise. An example is shown in the top-right image of Fig. 2. This is the filtered ODW of the image to its left, and this will be used for obstacle detection.

Below we explain the three obstacle detection algorithms: (a) Zero-Crossing Counting, (b) Histograms of Binary Edges and (c) Laws' Texture Energy Masks. If at least two of these detect an anomaly at about the same position, an obstacle is assumed. At least 3 successive frames are required to confirm the presence of an obstacle and before alerting the user.

(a) In Zero-Crossing Counting, we compute derivatives in x and y in I_{ODW}, using a large kernel $K = (-1,-1,-1,0,1,1,1)$. Then we sum the amplitudes of the maxima and minima near every zero-crossing (ZC): each time the derivative changes sign, we look for the minimum and maximum value on both sides and sum the absolute values. For analyzing variations on lines we use the x derivative, and for columns we use the y derivative. This is done for every line and column in the ODW. These arrays are then filtered twice with a 7×1 smoothing kernel. Filtered values below 3 in the histograms are due to noise and are removed. All parameters and kernels were determined experimentally using different test sequences. Examples of x and y derivatives are shown in the 1st and 2nd images on the 2nd row of Fig. 2, with the two histograms in black.

Thresholds are applied to the histograms in order to remove "noise" caused by the texture of the pavement. Let $T_{\max/\min,dx/dy}$ denote the maximum and minimum thresholds of the derivatives in x and y, and i be the frame index of a sequence. Values in the interval $[T_{\min,dx/dy}, T_{\max,dx/dy}]$ are set to zero, as these are caused by a textured pavement. In the first frames during initialization ($i \leq 5$), we must assume that no obstacle is present and $T_{\max,dx/dy}$ will be set to the

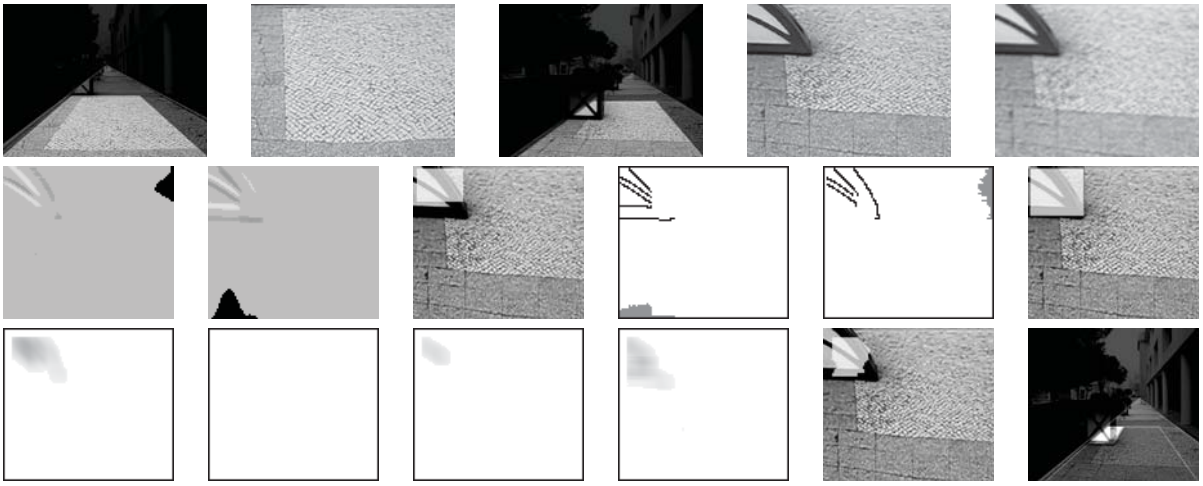

Figure 2: Top row, from left: two frames with the OW highlighted, their rectangular ODWs to their right, and the result of lowpass filtering. The latter is used for obstacle detection. Middle row, from left: x and y derivatives with histograms in black and detected obstacle region; the histograms of horizontal and vertical edges in gray and detected obstacle region. Bottom row, from left: the local energy images of Laws' masks E5L5, R5R5, E5S5 and L5S5, the detected obstacle region, and the final result, after combining the three methods, in the input frame.

corresponding maximum values of $I_{\mathrm{ODW},dx/dy}$. Similarly, $T_{\mathrm{min},dx/dy}$ are determined. Here the entire ODW is used.

After initialization ($i \geq 6$), the same method is applied, but the maxima and minima are calculated using only the bottom half of the window, i.e., $x \in [-W_{\mathrm{ODW}}/2, W_{\mathrm{ODW}}/2-1]$ and $y \in [0, H_{\mathrm{ODW}}/2]$. If the maximum of a derivative of the current frame i is higher than the maximum threshold of the previous frame, $\max_i > T_{\max,i-1}$, or the minimum is lower than the minimum threshold, $\min_i < T_{\min,i-1}$, then $T_{\max/\min,i} = \max_i / \min_i$; otherwise $T_{\max/\min,i} = (\max_i / \min_i + T_{\max/\min,i-1})/2$. This allows the system to adapt to different types of pavements while walking, but it is still sensitive to deviations.

This procedure yields a binary image resulting from the multiplication of the thresholded and back-projected line and column histograms. The 3rd image on the 2nd row of Fig. 2 shows the result as a bright rectangular area in the ODW.

(b) In the Histograms of Binary Edges algorithm Canny's edge detector is applied to I_{ODW}, with the same σ and T_l as used in path detection, but with T_h as described below. Canny's algorithm is used to calculate first derivatives dx and dy, the edge magnitude $I_{\mathrm{ODWc},mag} = (dx^2 + dy^2)^{1/2}$, and the edge orientation $I_{\mathrm{ODWc},\theta} = \arctan(dy/dx)$. This information is used as follows: (b-i) To compute dynamically the high hysteresis threshold T_h. During initialization, $i \leq 5$, the maximum value of $T_h = \max_i[I_{\mathrm{ODWc},mag}]$ is used. For $i \geq 6$, if the maximum in the bottom half of $I_{\mathrm{ODWc},mag}(x,y)$, ($x \in [-W_{\mathrm{ODW}}/2, W_{\mathrm{ODW}}/2-1]$ and $y \in [0, H_{\mathrm{ODW}}/2]$), is higher than the maximum threshold of the previous frame ($\max_i > T_{h,i-1}$), then $T_{h,i} = \max_i$; otherwise $T_{h,i} = (\max_i + T_{h,i-1})/2$. As in algorithm (a), this allows to adapt to different types of pavements.

(b-ii) As we want to determine the region where the obstacle is, we split the resulting Canny edge map I_{ODWc} using the edge orientations. This yields two images $I_{\mathrm{ODWc,V/H}}$ with the vertical and horizontal edges. For this we check the angles in $I_{\mathrm{ODWc},\theta}$. We apply four angle intervals: for horizontal edges $\theta_H \in [-67.5^o, 67.5^o] \cup [112.5^o, 247.5^o]$ and for vertical ones $\theta_V \in [22.5^o, 157.5^o] \cup [202.5^o, 337.5^o]$. The edge pixels in I_{ODWc} for which the angles are in the θ_H and θ_V intervals are stored in $I_{\mathrm{ODWc,H/V}}$.

For locating the region where an obstacle might be, edge histograms are filled along columns and lines in $I_{\mathrm{ODWc,V}}$ and $I_{\mathrm{ODWc,H}}$. The 4th and 5th images on the 2nd row of Fig. 2 show the edge maps in black and the histograms in gray at the borders. All bins in both histograms with values below 2 are discarded for a better localization with left-right and top-bottom limits. The final image results from the multiplication of the back-projected histograms. This is shown by the bright rectangular area in the rightmost image on the 2nd row of Fig. 2.

(c) The third algorithm is based on Laws' Texture Energy Masks (Laws, 1980) applied to I_{ODW}. The main idea is to detect changes of the image's textures in the window. If the frames contain textures before an obstacle enters the ODW, these will not be detected through the use of a threshold value. As in (Laws, 1980), our tests showed that the best masks are E5L5, R5R5, E5S5 and L5S5. Below the masks are denoted

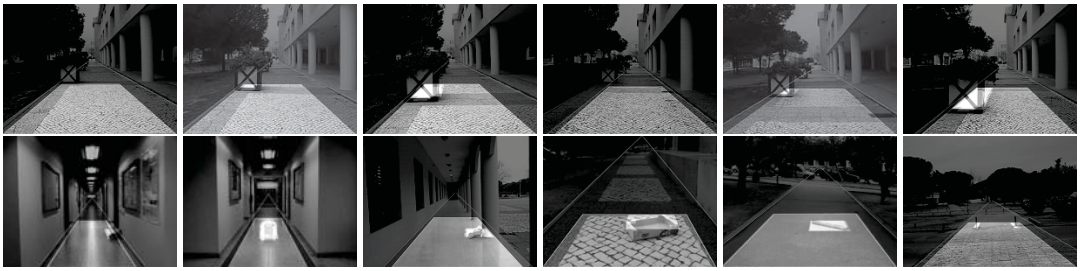

Figure 3: Typical results of in- and outdoor test sequences. Only a few frames are shown.

by subscript lm.

After filtering with the masks, which results in 4 images $I_{\mathrm{ODW,lm}}$, we compute the energy measures $E_{\mathrm{lm}}(x,y) = \sum_{i,j=-5}^{5} I_{\mathrm{ODW,lm}}(x+i,y+j)^2$. The four energy images are then normalized using the maximal energy responses that each mask can achieve, such that each mask contributes equally in final detection. The four normalized energy images are then summed and the maximum value is determined. All values above 4% of the maximum are considered to be due to a possible obstacle. The rest of the processing is equal to the processing as described in Zero-Crossing Counting. The bottom row in Fig. 2 shows, from left, the four E_{lm} and the summed and thresholded result. The final result of the 3 combined methods is also shown in Fig. 2 (bottom-right). The region where the regions detected by at least 2 algorithms overlap is highlighted. More results are shown in Fig. 3.

In summary, if an obstacle is detected (i) in at least 3 consecutive frames, (ii) by at least 2 of the 3 algorithms, and (iii) with regions in the ODW whose intersections are not empty, the user will be alerted.

4 CONCLUSIONS

Computationally fast methods intended to run in real-time are not easy to develop, as we do not want to sacrifice performance. All algorithms presented here can run on an inexpensive netbook at more than 5 fps, with very satisfying results. In all sequences tested so far, some with complex path and pavement structures, paths were correctly detected. Also most simple and complex obstacles were detected, only failing when the obstacles were too similar to the pavement or when multiple textures were present, yielding false negatives and positives. A few examples of sequences (not all frames) are shown in Fig. 3, where the user can be alerted well in advance. The system is now being extended to cope also with sharp turns and other frequent complications, and extensive field tests involving blind users are being prepared with ACAPO, the Portuguese association for blind and amblyopes, in the project coined Blavigator, from Blind Navigator.

It should be stressed once more that the vision system will complement the white cane beyond its reach; it is not intended to replace the cane. In addition, it only serves local navigation for path and obstacle negotiation. Global GPS/GIS-based navigation will complement the vision system (du Buf et al., 2011), leading to improved and autonomous mobility.

ACKNOWLEDGEMENTS

This research was supported by the Portuguese Foundation for Science and Technology (FCT), through the pluriannual funding of the Institute for Systems and Robotics (ISR/IST) through the PIDDAC Programme funds, and by the FCT project Blavigator (RIPD/ADA/109690/2009).

REFERENCES

Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698.

CASBliP (2009). Final activity report of the EU-funded CASBliP project. http://casblipdif.webs.upv.es/.

du Buf, J., Barroso, J., Rodrigues, J., Paredes, H., Farrajota, M., Fernandes, H., José, J., Teixeira, V., and Saleiro, M. (2011). The smartvision navigation prototype for blind users. *JDCTA: International Journal of Digital Content Technology and its Applications*, 5(5):351–361.

Duda, R. and Hart, P. (1972). Use of the hough transform to detect lines and curves in pictures. *Comm. ACM*, 15:11–15.

Kim, L., Park, S., Lee, S., and Ha, S. (2009). An electronic traveler aid for the blind using multiple range sensors. *IEICE Electronics Express*, 11(6):794–799.

Laws, K. (1980). *Textured image segmentation.* PhD thesis, USCIPI Report 940, Image Processing Inst., Univ. of Southern California, Los Angeles (USA).

Lee, S., Kang, S., and Lee, S. (2008). A walking guidance system for the visually impaired. *Int. J. Pattern Recogn. Artif. Intell*, 22(6):1171–1186.