View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Sapientia

A vision system for detecting paths and moving obstacles for the blind

J. José, M. Farrajota, J.M.F. Rodrigues, J.M.H. du Buf

Vision Laboratory, Institute for Systems and Robotics (ISR),
University of the Algarve (FCT and ISE), Faro, Portugal

{jjose, jrodrig, dubuf}@ualg.pt and elsio_farrajota@hotmail.com

Abstract — **In this paper we present a monocular vision system for a navigation aid. The system assists blind persons in following paths and sidewalks, and it alerts the user to moving obstacles which may be on collision course. Path borders and the vanishing point are detected by edges and an adapted Hough transform. Optical flow is detected by using a hierarchical, multi-scale tree structure with annotated keypoints. The tree structure also allows to segregate moving objects, indicating where on the path the objects are. Moreover, the centre of the object relative to the vanishing point indicates whether an object is approaching or not.**

I. Introduction

Navigation of blind people is very arduous because they must use the white cane for obstacle detection while following the front sides of houses and shops, meanwhile memorising all locations they are becoming familiar with. In a new, unfamiliar setting they completely depend on people passing by to ask for a certain shop or the closest post office. Crossing a street is a challenge, after which they may be again disoriented. In a society in which very sophisticated technology is available, from tracking GPS-RFID equipped containers in an area of hundreds of meters to GPS-GIS car navigation to Bluetooth emitting the sound of movie trailers to mobile phones in front of cinemas, one can question what it may cost to provide the blind with the most elementary technology to make life a little bit easier. This technology may not replace the cane, but should complement it: alert the user to obstacles a few metres away and provide guidance for going to a specific location in town or in a shopping centre.

Different approaches exist to help the visually impaired. One system for obstacle avoidance is based on a hemispherical ultrasound sensor array [1]. It can detect obstacles in front and unimpeded directions are obtained via range values at consecutive times. The system comprises an ARM9 embedded system, the sensor array, an orientation tracker and a set of pager motors. Talking Points is an urban orientation system [5] based on electronic tags with spoken (voice) messages. These tags can be attached to many landmarks like entrances of buildings, elevators, but also bus stops and busses. A push-button on a hand-held device is used to activate a tag, after which the spoken message is made audible by the device's small loudspeaker. iSONIC [10] is a travel aid complementing the cane. It detects obstacles at head-height and alerts by vibration or sound to dangerous situations, with an algorithm to reduce confusing and unnecessary detections. iSONIC can also give information about object colour and environmental brightness.

The Portuguese project “SmartVision: active vision for the blind,” financed by the Portuguese Foundation for Science and Technology, combines several technologies, GPS, GIS, Wi-Fi and computer vision, to create a system which assists the visually impaired navigate in- and outdoor [7]. One of its modules serves to help the blind navigate outdoors on paths and sidewalks. It must alert the user to possible obstacles, both fixed objects and moving ones like persons and animals which may be on collision course, and how to avoid them.

There exist some methods to detect the borders of paths and sidewalks, see e.g. [9]. In a previous paper [2] we presented a detection method for paths with fixed obstacles. The system first detects the path borders, using edge information in combination with a tracking mask, to obtain straight lines with their slopes and the vanishing point. Once the borders are found, a rectangular window is defined within which two obstacle detection methods are applied. The first determines the variation of the maxima and minima of the gray levels of the pixels. The second uses the binary edge image and searches in the vertical and horizontal histograms for discrepancies in the number of edge points. Together, these methods allow to detect possible obstacles with their position and size, such that the user can be alerted and informed about the best way to avoid them.

In this paper we present an improved method for border detection, which is faster and more robust, and focus on moving objects on the path which may be on collision course.

In Section 2 we present path detection and in Section 3 the detection of moving objects. Section 4 is devoted to tracking moving objects on the path which may be on collision course, and we conclude with a discussion in Section 5.

Acknowledgements: This research was supported by the Portuguese Foundation for Science and Technology (FCT), through the pluri-annual funding of the Institute for Systems and Robotics (ISR-Lisbon/IST) by the POS_Conhecimento Program which includes FEDER funds, and by the FCT project SmartVision: active vision for the blind (PTDC/EIA/73633/2006).

II. Path Detection

In the SmartVision project, a stereo camera (Bumblebee 2 from Point Grey Research Inc.) is fixed to the chest of the blind, at a height of about 1.5 m from the ground. Results presented here were 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 the pavements related to possible obstacles like holes and loose stones [2] with a minimum size of about 10 cm at a distance of 3 to 5 m from the camera (the first metres are not covered because of the height of the camera; this area is covered by the cane swayed by the user).

Detection of path borders is based on: (A) defining a Path Detection Window (PDW) where we will search for the borders in each frame; (B) some pre-processing of the frame to detect the important edges and to build an Adapted Hough Space (AHS); and (C) the highest values in the AHS yield the borders.

A. Path Detection Window PDW

Let $I(x,y)$ denote an input frame with fixed width W_I and height H_I. Let *HL* denote the horizon line close to the middle of the frame. If the camera is exactly in the horizontal position, then *HL* = $H_I/2$. If the camera points lower or higher, *HL* will be higher or lower, respectively; see Fig. 1. The borders of the path or sidewalk are normally the most continuous and straight lines in the lower half of the frame, delimited by *HL*.

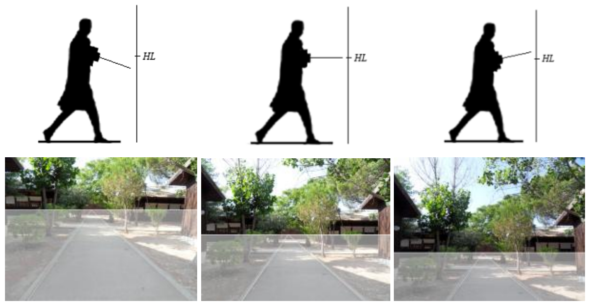

Figure 1. From left to right: camera pointing down, horizontally aligned, and pointing up. The Path Detection Window is highlighted in the images.

Because of perspective projection, the left and right borders of the path and many other straight structures intersect at the vanishing point *VP*. Since the horizontal camera alignment is not fixed but varies over time when the user walks, we use the *VP* in order to determine the line: $y_{HL} = y_{VP}$. Consequently, the path detection window is defined by $I_{PDW}(x,y)$ with $x = [1, W_I]$ and $y = [y_{VP}, H_I]$, if the top-left pixel of each frame is the origin of the coordinate system. Different PDWs are illustrated in Fig. 1.

The value of *HL* is computed dynamically, by averaging the values of the previous five frames: for HL_i, i.e., frame number i which still must be analysed, $y_{VP,i} = (\sum_{j=i-5}^{i-1} y_{VP,j})/5$. This cannot be done in the case of the first five frames, for which we use $y_{VP} = H_I/2$. This is not a problem because the first frames are mainly used for system initialisation and a frame rate of 5 fps implies only one second.

B. Adapted Hough Space AHS

The Canny edge detector and an adapted version of the Hough transform are applied to I_{PDW} for the detection of the borders and the vanishing point. In order to reduce CPU time, only gray-scale information is processed after resizing the window to a width of 300 pixels using bilinear interpolation, maintaining the aspect ratio. Then two iterations of a 3x3 smoothing filter are applied in order to suppress noise.

The Canny [3] edge detector is applied with $\sigma = 1.0$, which defines the size of the Gaussian filter, in combination with $T_L = 0.25$ and $T_H = 0.5$ which are the low and high thresholds for hysteresis edge tracking. The result is a binary edge image $I_P(x_P, y_P)$, of width $W_P = 300$ and height $H_P = H_I(W_P/W_I)$, with $x_P = [1, W_P]$ and $y_P = [1, H_P]$, and with the extrapolated horizontal line (vanishing point) at $y_P = y_{VP}(W_P/W_I)$. Figure 2 (left) shows one original frame together with the resized and lowpass-filtered PDW (top-right) and detected edges (bottom-right).

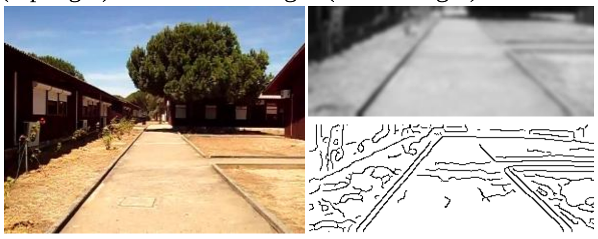
Figure 2. Left: one original frame. Right: resized PDW after low-pass filtering (top) and the binary edge image I_P (bottom).

The borders of paths and sidewalks are usually found to the left and to the right, assuming that the path or sidewalk is in the camera's field of view; see e.g. Fig. 4. We use the Hough transform [12], where $\rho = x * \cos\theta + y * \sin\theta$, to search for lines in the left and right halves of the PDW for border candidates, also assuming that candidates intersect at a vanishing point.

As we want to check straight lines in the two halves of the window using polar coordinates, we use a different reference point. Let $I'_P(x'_P, y'_P)$, see Fig. 3 (top), be the new origin at the bottom-centre of image I_P, where y'_P is related to y_P by $y'_P = H_P - (y_P - 1)$ and x'_P is related to x_P by $x'_P = (x_P - 1) - W_P/2$, with $y'_P = [1, H_{I_P}]$ and $x'_P = [-W_{I_P}/2, W_{I_P}/2 - 1]$.

The Hough transform is applied to I'_P, yielding the Adapted Hough Space $I_{AHS}(\rho, \theta)$ with $\theta = [0°, 180°]$ and $\Delta\theta = 0.5°$, and $\rho = [0, W_P/2 \,.\cos\theta + H_P \,.\sin\theta]$. For $\theta = [0°, 44°]$ we apply $\Delta\rho = \cos\theta$, and for $\theta = [45°, 90°]$ we apply $\Delta\rho = \sin\theta$, in order to adjust the interval to polar coordinates such that no lines are repeated or missed.

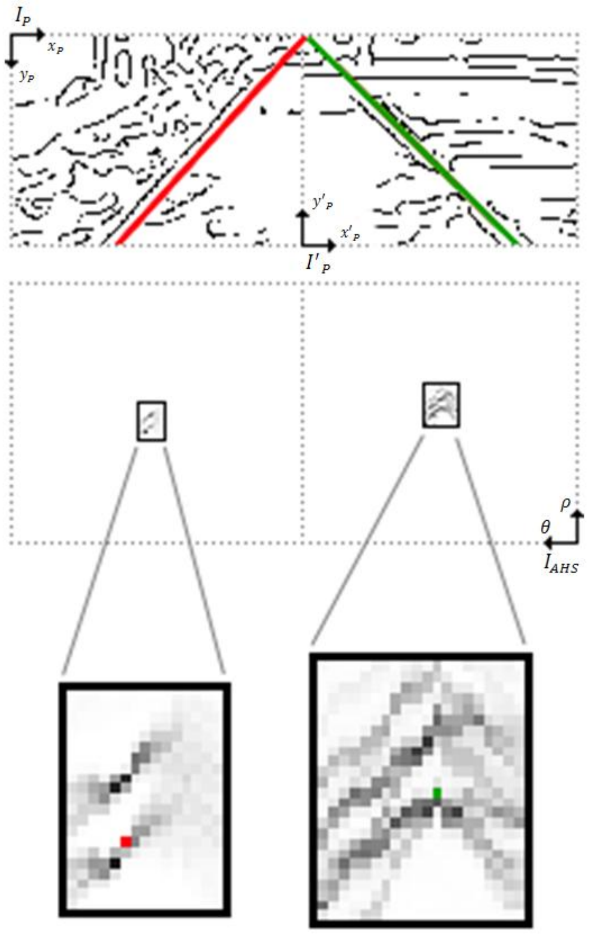

Figure 3. Top: PDW with detected edges and coordinate systems. Middle: AHS with zoomed areas (at bottom). The left and right borders are marked in red and green, respectively.

Since we have "mirrored" lines for θ and $\pi - \theta$ for the same values of ρ, we only calculate the lines for $\theta = [0°, 89°]$ for the right border, which we denote by $L_{\rho,\theta}(x'_{P,r}, y'_P)$. Similarly, the lines for $\theta = [91°, 180°]$ are computed for the left border, $L_{\rho,\pi-\theta}(x'_{P,l}, y'_P)$, because of the y'_P mirror axis.

For $L_{\rho,\theta}$ we use $x'_{P,r} = (\rho - y'_P \sin\theta)/\cos\theta$ and $y'_P = (\rho - x'_{P,r}\cos\theta)/\sin\theta$. For $L_{\rho,\pi-\theta}$ we use $x'_{P,l} = -x'_{P,r} - 1$ and the same y'_P as for $\theta = [0°, 89°]$. This means that $I_{AHS}(0, 0°)$ corresponds to a vertical line with $y'_P = [1, H_P]$ and $x'_{P,r} = 0$, and $I_{AHS}(0, 180°)$ has the same y'_P but $x'_{P,l} = -1$. For obtaining the maximal number of pixels on the projected lines $L_{\rho,\theta}$ and $L_{\rho,\pi-\theta}$, we increment by 1 the y'_P and compute the corresponding $x'_{P,r}$ and $x'_{P,l}$ for $\theta = [0°, 44°]$. For $\theta = [45°, 89°]$ we increment by 1 the $x'_{P,r}$, with $x'_{P,l} = -x'_{P,r} - 1$, and calculate the corresponding y'_P.

The I_{AHS} space is filled by checking the pixels in I'_P 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'_P. However, there are two differences.

First, vertical and horizontal lines in the image cannot be borders of a path or sidewalk (see e.g. Fig. 1). Hence, we restrict the Hough space to $\theta = [20°, 69°] \cup [111°, 160°]$ such that vertical and almost vertical lines in the intervals $\theta = [0°, 20°]$ and $[160°, 180°]$ are ignored; the same is done for horizontal and almost horizontal lines in the interval $[70°, 110°]$. This yields a reduction of CPU time of about 30%.

Second, longer sequences of edge pixels count more than short sequences or not-connected edge-pixels. To this purpose we use a counter P which can be increased or reset. When we check each pixel in I'_P for a projected line $L_{\rho,\theta}$ and find the 1st ON pixel, $P = 1$ and the corresponding $I_{AHS}(\rho, \theta)$ bin will increment by 1. If the 2nd pixel following the first ON pixel is also ON, P will be incremented by 2, and $I_{AHS}(\rho, \theta)$ is incremented by P=3, and so on. If a next pixel is OFF, the variable P is reset to 0 and $I_{AHS}(\rho, \theta)$ is not changed. In other words, a run of n connected edge pixels has P values of 1, 3, 5, 7, etc., or $P_n = P_{n-1} + 2$, with $P_1 = 1$, and the run will contribute n^2 to the relevant I_{AHS} bin.

The final value of the $I_{AHS}(\rho, \theta)$ bin is the sum of the P_n value(s) of all sequences of ON pixel(s): $I_{AHS}(\rho, \theta) = \sum_{l=1}^{k} P_{n,l}$, with k the number of sequence(s) of ON pixel(s) and each sequence having at least one ON pixel.

An I_{AHS} is shown in Fig. 3 (middle) together with magnified regions (bottom). The left and right borders are marked in red and green, respectively, also in the edge map I'_P (top).

C. Path borders

Until here we explained the computation of I_{AHS}, but only during the initialisation phase of the first 5 frames. After the initialisation phase, for optimisation and accuracy purposes, we will not check the entire I_{AHS} space. Each border (ρ, θ), both left and right, is stored during the initialisation in the array $M_i(\rho, \theta)$, with i the frame number.

After the fifth frame ($i = 6$), we already have five pairs of points in M, which define two regions in I_{AHS}. These regions indicate where in $I_{AHS,i}$ the next border positions are expected. The regions are limited by the minima $\rho_{min,l/r}$ and $\theta_{min,l/r}$, and by the maxima $\rho_{max,l/r}$ and $\theta_{max,l/r}$, in the left and the right halves of I_{AHS}.

In frames $i \geq 6$, we look for the highest value(s) in $I_{AHS,i}(\rho, \theta)$ in the regions between $\rho_{min,l/r} - T_\rho$ and $\rho_{max,l/r} + T_\rho$, and between $\theta_{min,l/r} - T_\theta$ and $\theta_{max,l/r} + T_\theta$, on the left and on the right side, respectively, with $T_\rho = 10$ and $T_\theta = 5°$. This procedure is applied for all $i \geq 6$, always considering the borders found in the previous five frames.

In the two regions as defined above we look for the highest values in $I_{AHS,i}$. We start by checking the highest value, and then the 2nd highest value. If the 2nd highest value represents a border which is more similar to the border of the previous frame, we still check the 3rd highest value and so on. If a next highest value does not correspond to a border which is more similar to the border of the previous frame, the search is terminated and the best match is selected.

Borders are considered more similar if the intersection of the new candidates (VP_i) and the intersection of the borders of the previous frame (VP_{i-1}) have a smaller distance d, with $d = ((VP_{x,i} - VP_{x,i-1})^2 + (VP_{y,i} - VP_{y,i-1})^2)^{1/2}$.

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 at least 10 connected ON pixels, the border is considered not found for that side. In this case, the average of the last 5 borders found is used: $\rho_i = (\sum_{j=i-5}^{i-1} \rho_j)/5$ and $\theta_i = (\sum_{j=i-5}^{i-1} \theta_j)/5$ on the corresponding side. Figure 4 shows the results of path and border detection in the case of two image sequences.

III. Detection of moving objects

Apart from detecting path borders and possible stationary obstacles on the path, see also [2], it is necessary to detect and track moving obstacles like persons and animals. To this purpose we use multi-scale, annotated, and biologically-inspired keypoints. Keypoint detection is based on Gabor filters [6], and provides important image information because keypoints code local image complexity. Moreover, since keypoints are caused by line and edge junctions, detected keypoints can be classified by the underlying vertex structure, such as K, L, T, + etc. This is very useful for matching problems: object recognition, stereo disparity and optical flow.

The process for tracking moving objects consists of three steps: (A) multi-scale keypoints are detected and annotated; (B) multi-scale optical flow maps are computed and objects are segregated; and (C) the regions that enclose objects allow us to track the objects' movements and their directions.

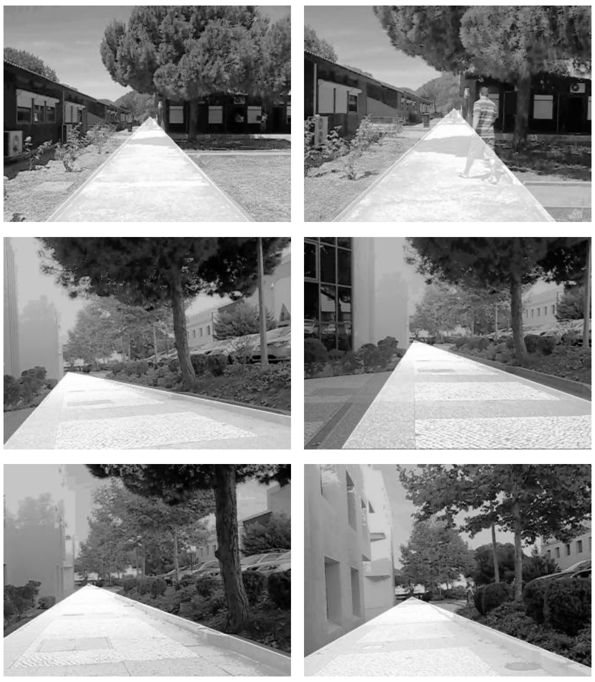
Figure 4. Two sequences with detected borders (white lines), the path being highlighted.

A. Keypoint Detection and Annotation

Gabor quadrature filters provide a model of cortical simple cells [6]. In the spatial domain (x, y) they consist of a real cosine and an imaginary sine, both with a Gaussian envelope.

Responses of even and odd simple cells, which correspond to real and imaginary parts of a Gabor filter, are obtained by convolving the input image with the filter kernels. Responses are denoted by $R^E_{s,j}(x,y)$ and $R^O_{s,j}(x,y)$, s being the scale given by the wavelength λ (λ = 1 corresponds to 1 pixel), and j the orientation $\Theta_j = j\pi/N_\Theta$ with N_Θ the number of orientations. We use 8 orientations, $j = [0, N_\Theta - 1]$ and 8 scales equally spaced on λ = [6, 27] with $\Delta\lambda = 3$.

Responses of complex cells are modelled by the modulus $C_{s,j}(x,y)$, which feed two types of end-stopped cells, single $S_{s,j}(x,y)$ and double $D_{s,j}(x,y)$; for details see [6]. Responses of end-stopped cells in combination with sophisticated inhibition schemes yield keypoint maps $K_s(x,y)$.

In order to classify any detected keypoint, the responses of simple cells $R^E_{s,j}$ and $R^O_{s,j}$ are analysed, but now using $N_\phi = 2 \times N_\Theta$ orientations, $\phi_k = k\pi/N_\Theta$ and $k = [0, N_\phi - 1]$. This means that for each Gabor filter orientation on $[0, \pi]$ there are two opposite keypoint classification orientations on $[0, 2\pi]$, e.g. a Gabor filter at $\Theta_1 = \pi/N_\Theta$ results in $\phi_1 = \pi/N_\Theta$ and $\phi_9 = 9\pi/N_\Theta$.

Classifying keypoints is not trivial, because responses of simple and complex cells, which code the underlying lines and edges at the vertices, are unreliable due to response interference effects [8]. This implies that responses must be analysed in a neighbourhood around each keypoint, and the size of the neighbourhood must be proportional to the scale of the cells, i.e., the size of the filter kernels.

The validation of line and edge orientations which contribute to the vertex structure is based on an analysis of the responses, both $R^E_{s,j}$ and $R^O_{s,j}$, and consists of three steps: (1) only responses with small variations at three distances are considered, (2) local maxima of the responses over orientations are filtered and the remaining orientations are discarded, and (3) even and odd responses are matched in order to filter the orientations which are common to both. The same processing is applied at any scale s (λ).

In step (1), for each orientation Φ_k the responses of the simple cells on three circles around the keypoint positions, with radii $\lambda/2$, λ and 2λ, are compared, considering also orientation intervals $\Phi_k \pm \pi/N_\phi$. The three maximum responses $R_{k,r}$ in the orientation interval around k and at the three radii r are detected, and their maximum $\hat{R}_k = \max_r R_{k,r}$. Only responses with small variations at the three radii are considered, i.e., $R > 0.6\hat{R}_k$.

In step (2), the average $\bar{R} = (1/N_\phi)\sum_k R_k$ is computed and, for validation purposes, all $\hat{R}_k$ below $0.95\bar{R}$ are suppressed. If there exist maximum responses $\hat{R}_k$ for the two neighbouring orientations ϕ_{k-1} and ϕ_{k+1}, they will be removed if $\hat{R}_{k-1} < 0.95\hat{R}_k$ or $\hat{R}_{k+1} < 0.95\hat{R}_k$. The above values were determined by analysing simple objects like triangles, squares and polygons.

The analysis in step (3) only concerns the matching of equal orientations, discarding any orientation which is not detected in the responses of both even ($R^E_{s,j}$) and odd ($R^O_{s,j}$) simple cells. Remaining orientations Φ_k are attributed to the keypoint, plus the junction type L, T, + etc.

In the above procedure there is only one exception: keypoints at isolated points and blobs, especially at very coarse scales, are also detected but they are not caused by line/edge junctions. Such keypoints are labeled “blob” without attributed orientations.

Figure 5 illustrates responses of simple cells in the case of a black square in a noisy background. It shows two scales, λ = 6 (column 1 and 2) and λ = 15 (column 3 and 4), only three orientations of all 8, even cells in columns 1 and 3 and odd cells in columns 2 and 4. Dark levels are negative and bright ones are positive. Also shown is one detected keypoint at each scale with (in red) the three distances λ/2, λ and 2λ at which the responses are tested.

Figure 6 shows keypoint detection and annotation results together with optical flow. At top-left it shows one frame from the sequence shown in Fig. 1, and at top-right a combination of two successive frames. The second row shows keypoints detected at two scales, λ = 6 (left) and 15 (right). The third row shows annotated keypoints at the two scales, and the fourth displacement vectors between the successive frames.

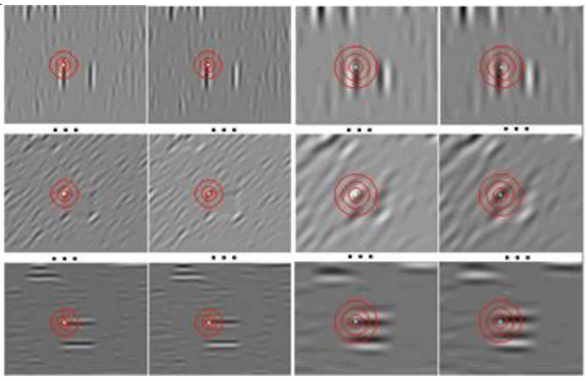
Figure 5. Responses of simple cells. Top to bottom: 3 orientations at two scales λ = 6 (left) and 15 (right). Left to right: responses of even and odd simple cells, plus one detected keypoint with (in red) the 3 distances at which the responses are analysed.

B. Optical Flow

To compute the optical flow, we do not consider each scale independently for two reasons: (1) non-relevant areas of the image can be skipped because of the hierarchical scale structure, and (2) by applying a multi-scale strategy, the accuracy of keypoint matching can be increased, thus increasing the accuracy of the overall optical flow. Therefore we apply a multi-scale tree structure in which at the coarsest scale a root keypoint defines a single object, and finer scales add more keypoints which constitute the object’s parts and details. As stated before, at a very coarse level a keypoint may correspond to one big object. However, because of limited CPU time the coarsest scale applied will be λ = 27, which is a compromise between speed and quality of results. Hence, at the moment all keypoints at λ = 27 are considered to represent individual objects, although we know that several of those may belong to the same object.

Each keypoint at the coarsest scale can be linked to one or more keypoints at one finer scale, which can be slightly displaced. This link is created by down-projection using an area with the size of the filter (λ). This linking is repeated until the finest scale is reached. Hence, keypoints at a finer scale which are outside the “circle of influence” of keypoints at a coarser scale will not be relevant, thus avoiding unnecessary computations. Figure 7 illustrates the linking principle by cones.

At any scale, each annotated keypoint of frame *i* can be compared with all annotated keypoints in frame *i-1*. However, this comparison is restricted to an area of radius λ in order to save time, because (1) at fine scales many keypoints outside the area can be skipped since they are not likely to match over large distances, and (2) at coarse scales there are less keypoints, the radius λ is bigger and therefore larger distances (motions) are represented there. The tree structure is built top-down (Fig. 7), but the matching process is bottom-up: it starts at the finest scale because there the accuracy of the keypoint annotation is better.

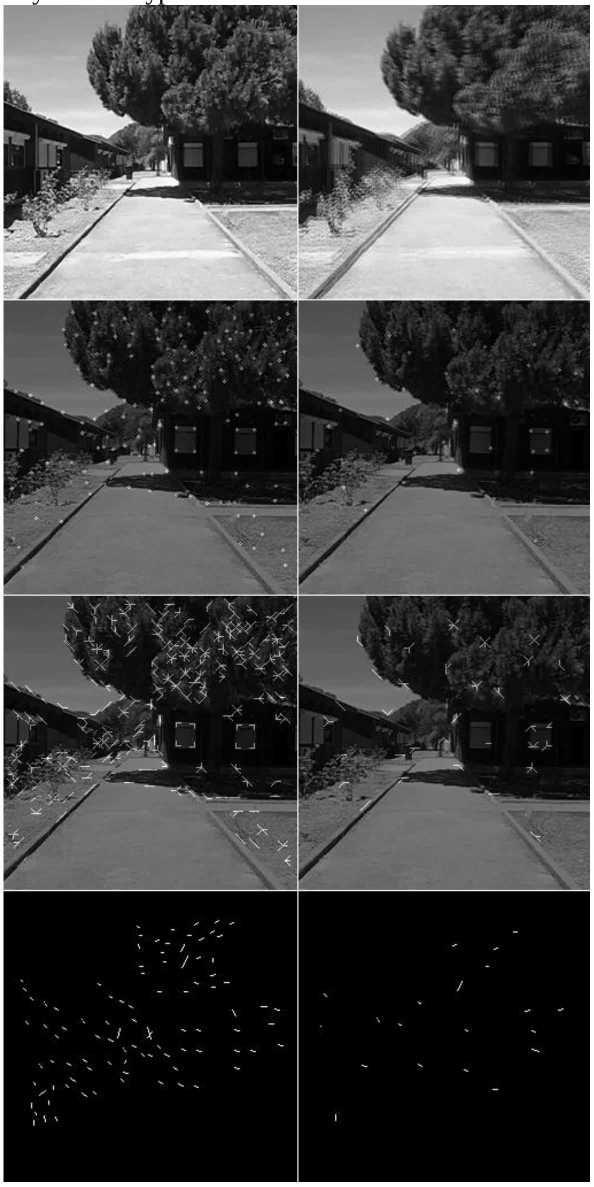

Figure 6. Keypoint annotation and matching. Top: one frame from the sequence shown in Fig. 1 (left) and two successive frames combined (right). The second row shows detected keypoints at scales λ = 6 (left) and 15 (right). The third row shows annotated keypoints and the bottom row displacement vectors between the successive frames (top-right).

Keypoints are matched using three similarity criteria with different weight factors: the distance *D*, the attributed orientations *O*, and the tree correspondence *C*. The distance *D* serves to emphasise keypoints which are closer to the centre of the area. For having *D*=1 at the centre and $D = 0$ at radius λ, we use $D = (\lambda - d)/\lambda$ with d the Euclidean distance. The orientation error O measures the differences of the attributed orientations, but with a relaxation of $\pm\pi/N_\theta$ of all orientations such that also small rotations are allowed. Similar to *D*, the summed differences are combined such that $O = 1$ indicates good correspondence and $O = 0$ a lack of correspondence. Obviously, keypoints marked "blob" do not have orientations and are treated separately. Parameter *C* measures the number of matched keypoints at finer scales, i.e., at any scale coarser than the finest scale. The keypoint candidates to be matched in frame *i* and in the area with radius λ are linked in the tree to localised sets of keypoints at all finer scales. The number of linked keypoints which have been matched is divided by the total number of linked keypoints. Hence, parameter *C* describes the consistency of the matching at a candidate's position at the finer scales, thereby influencing the matching of the candidate at the actual scale.

The three parameters are combined using the similarity measure $S = \alpha(\beta P_O + (1-\beta)P_T) + (1-\alpha)P_D$ with parameters α = 0.7 and β = 0.6. These values were determined empirically. The candidate keypoint with the highest value S in the area (λ) is selected and the vector between the keypoint in frame *i* and the matched one in frame *i*-1 is computed. The remaining candidates in the area can be matched to other keypoints in frame *i*-1, provided they are in their local area. Keypoints which cannot be matched are discarded. Figure 6 shows, at the bottom, the vectors between matched keypoints at λ = 6 (left) and λ =15 (right). Since optical flow in this example is mainly due to movement of the camera, it can be seen that there are some errors. In principle, outliers could be removed, but improvement of the matching process is still subject to ongoing research.

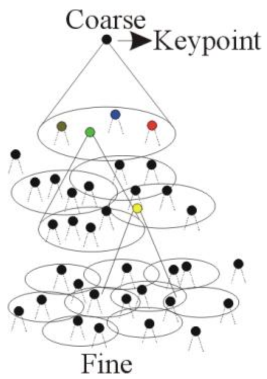

Figure 7. Hierarchical scale structure; see text.

IV. TRACKING OF OBJECTS ON COLLISION COURSE

As mentioned above, at a very coarse scale each keypoint should correspond to an individual object. However, at the coarsest scale applied (λ = 27) this may not be the case and an object may create several keypoints. In order to determine which keypoints may belong to the same object we combine saliency maps with the multi-scale tree structure.

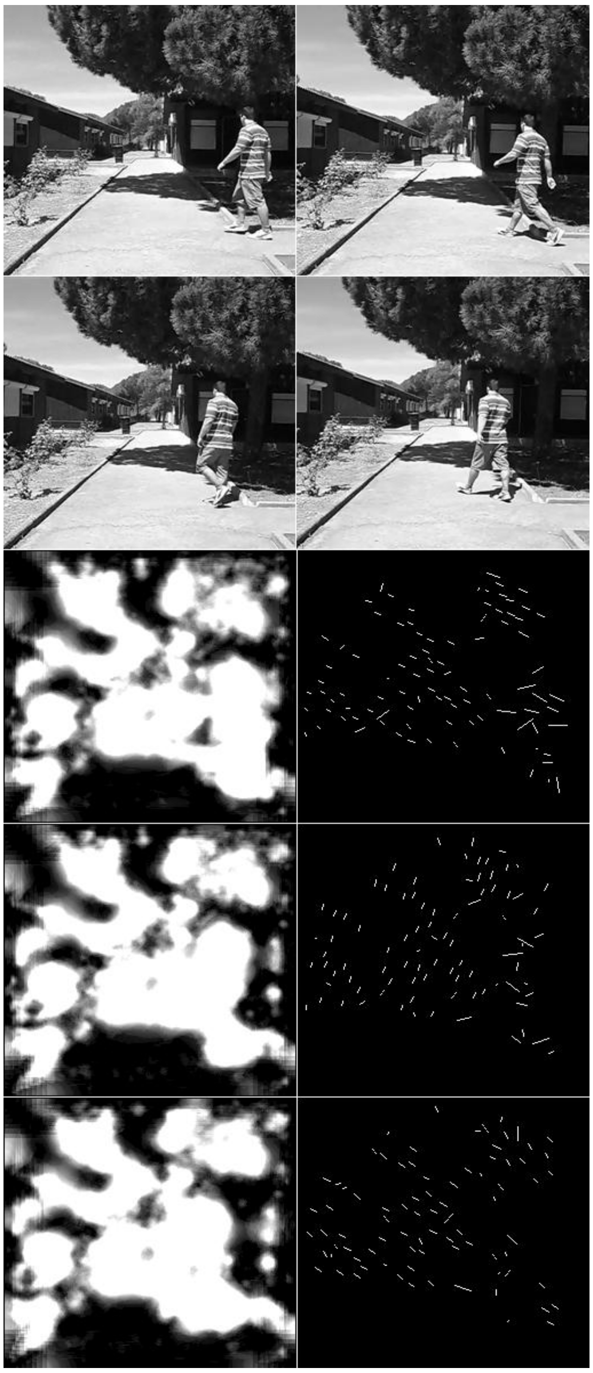

Figure 8. A sequence of 4 frames (top) with saliency maps at left and optical flow at right, both at λ=6.

A saliency map can be based on keypoints as these code local image complexity [6]. Such a map is created by summing detected keypoints over all scales s, such that keypoints which are stable over scale intervals yield high peaks, but in order to connect the individual peaks and yield regions a relaxation area is applied. As applied above, the area is proportional to the scale and has radius λ. Here, in order to save CPU time, the process is simplified and saliency maps are created by summing responses of end-stopped cells [6]. Figure 8 (bottom-left) shows three examples scaled to the interval [0, 255], but only at scale λ =6.

The saliency map of a frame defines, after thresholding, separated regions-of-interest (RoI) and these can be intersected with the regions as defined by the tree structure (Fig. 7). Hence, neighbouring keypoints are grouped together in the RoIs and their displacement vectors after the matching process yield the optical flow of segregated image regions, i.e., where an individual object or a combination of connected (sub)objects is or are moving. In order to discard small optical flow due to camera motion, optical flow vectors are only computed if at at least 4 scales the matched keypoints in successive frames have displacement vectors with a length which is bigger than 1 pixel.

Figure 8 (top) shows four successive frames of a sequence. The bottom part shows saliency maps of the last three frames (at left) together with optical flow vectors (at right), both at one fine scale. By using the intersections of the thresholded saliency maps and the areas defined by the tree structure, the optical flow vectors of the keypoints in segregated regions can be averaged, and the intersected RoIs can be approximated by curve fitting, here simplified by a rectangular bounding box. The centre of the bounding box is used for tracking moving objects over frames, also indicating where exactly the object is on the path. This is illustrated in Fig. 9. It shows parts of two sequences, and in each frame the detected path borders and the bounding boxes plus one combined image which shows the tracking of the centre of the bounding box. The flow vector of the centre allows us to detect the lateral movements left-to-right and right-to-left, and the distance between the centre and the vanishing point of the path borders indicates whether the object is moving towards or away from the camera. This is shown by the colour of the arrows in the top-right corners of the frames: yellow means movement at the same distance, green means going away, but red indicates an approaching object such that the user can be alerted to be even more cautious than in the yellow or green cases.

V. Conclusions

We presented a system for detecting path borders and the vanishing point, together with a biologically inspired algorithm for optical flow based on multi-scale keypoint annotation and matching. Moving obstacles can be detected and tracked, such that the blind user can be alerted and informed about the approximate position on the path and whether the object is approaching or not. Detection of moving obstacles complements detection of static obstacles in front on the path, just beyond the reach of the white cane [2].

Having a reasonably fast algorithm for optical flow, the same algorithm can be applied to stereo disparity in order to also estimate the distance to objects, both moving and static. The algorithms will be integrated in the SmartVision prototype, which also employs a GIS with GPS, WiFi and passive as well as active RFID tags [4].

The developed vision system is not unique. Recently, a similar system has been developed for intelligent cars, for tracking roads and lanes and for detecting possible obstacles like pedestrians [11]. The basic concepts like borders, vanishing point and optical flow are the same, but the implementation is completely different. This is also due to the different requirements: a car may have a speed of 100 km/h, but blind persons with the white cane do not exceed 1 m/s. However, all CPU power of a portable computer will be required because the ultimate goal is to substitute a big part of the functionality of a normal visual system.

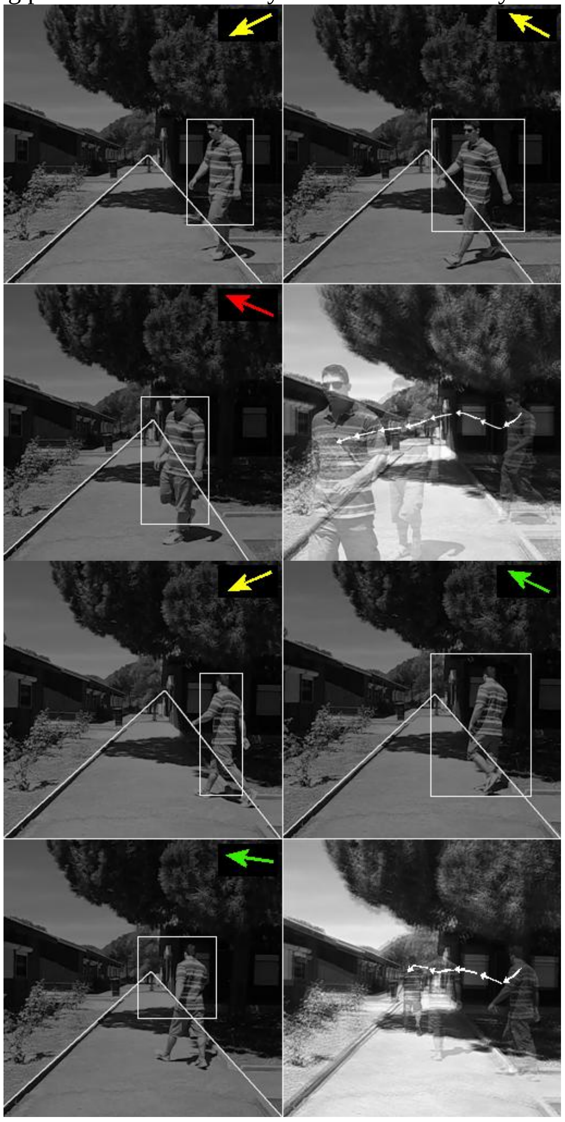
Figure 9. Parts of two sequences with tracked moving objects; see text.

References

[1] B. Shin, C. Lim. "Obstacle detection and avoidance system for visually impaired people", Proc. 2nd Int. Workshop on Haptic and Audio Interaction Design, Springer LNCS 4813, Seoul, South Korea, Nov. 29-30, 2007, pp.78-85.

[2] D. Castells, J.M.F. Rodrigues, J.M.H. du Buf. "Obstacle detection and avoidance on sidewalks", Proc. Int. Conf. on Computer Vision-Theory and Applications, Angers, France, May 17-21, 2010, Vol. 2, pp. 235-240.

[3] J. Canny. "A computational approach to edge detection". IEEE Trans. on Pattern Analysis and Machine Intelligence, 679-698, 1986.

[4] J. Faria, S. Lopes, H. Fernandes, P. Martins, J. Barroso. "Electronic white cane for blind people navigation assistance", Accepted for World Automation Congress, Kobe, Japan, Sept. 19-23, 2010.

[5] J. Stewart, S. Bauman, M. Escobar, J. Hilden, K. Bihani, M. Newman. "Accessible contextual information for urban orientation". Proc. 10th Int. Conf. on Ubiquitous Computing, Seoul, Korea, Sept. 21–24, 2008, Vol. 344, pp. 332-335.

[6] J.M.F. Rodrigues, J.M.H. du Buf. "Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection", BioSystems, Vol. 86, pp. 75-90, 2006. doi:10.1016/ j.biosystems.2006.02.019.

[7] J.M.H. du Buf et al. "The SmartVision navigation prototype for the blind", Subm. to Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion, Oxford, United Kingdom, Nov. 25-26, 2010.

[8] J.M.H. du Buf. "Responses of simple cells: events, interferences, and ambiguities." Biol. Cybern. 68, 321–333, 1993.

[9] K. Kayama, I. Yairi, S. Igi. "Detection of sidewalk border using camera on low-speed buggy". Proc. Int. Conf. on Artificial Intelligence and Applications, Innsbruck, Austria, , Feb. 13-15, 2007, pp. 262–267.

[10] L. Kim, S. Park, S. Lee, S. Ha. "An electronic traveler aid for the blind using multiple range sensors", IEICE Electronics Express 6 (11), 794-799, 2009.

[11] N. Onkarappa, A.D. Sappa. "On-board monocular vision system pose estimation through a dense optic flow", Proc. Int. Conf. on Image Analysis and Recognition, Póvoa do Varzim, Portugal, June 21-23, 2010, Springer LNCS 6111, pp. 230-239.

[12] R. Duda, P. Hart. "Use of the Hough transform to detect lines and curves in pictures". Comm. ACM, Vol. 15 , 11-15, 1972.