# The SmartVision local navigation aid for blind and visually impaired persons

João José, Miguel Farrajota, João M.F. Rodrigues, J.M. Hans 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***

*The SmartVision prototype is a small, cheap and easily wearable navigation aid for blind and visually impaired persons. Its functionality addresses global navigation for guiding the user to some destiny, and local navigation for negotiating paths, sidewalks and corridors, with avoidance of static as well as moving obstacles. Local navigation applies to both in- and outdoor situations. In this article we focus on local navigation: the detection of path borders and obstacles in front of the user and just beyond the reach of the white cane, such that the user can be assisted in centering on the path and alerted to looming hazards. Using a stereo camera worn at chest height, a portable computer in a shoulder-strapped pouch or pocket and only one earphone or small speaker, the system is inconspicuous, it is no hindrence while walking with the cane, and it does not block normal surround sounds. The vision algorithms are optimised such that the system can work at a few frames per second.*

**Keywords***: vision aid, path detection, obstacle avoidance*

## 1. 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 metres 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 [22]. It can detect obstacles in front and unimpeded directions are obtained via range values at consecutive times. The system comprises an embedded computer, the sensor array, an orientation tracker and a set of pager motors. Talking Points is an urban orientation system [24] 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 [16] 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.

GuideCane [26] is a computerised travel aid for blind pedestrians. It consists of a long handle attached to a sensor unit on a small, lightweight and steerable device with two wheels. While walking, the user holds the handle and pushes the GuideCane in front. Ultrasonic sensors detect obstacles and steer the device around them. The user feels the steering direction through the handle and can follow the device easily and without conscious effort. Drishti [20] is an in- and outdoor navigation system. Outdoor it uses DGPS localisation to keep the user as close as

possible to the central line of sidewalks. It provides the user with an optimal route by means of its dynamic routing facility. The user can switch the system from out- to indoor operation with a simple vocal command which activates a precise ultrasound positioning system. In both cases the user gets vocal prompts which alert to possible obstacles and which provide guidance while walking about.

CASBliP or Cognitive Aid System for Blind People [13] was a European Union-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. Environmental information from various sensors is acquired and transformed into enhanced images for visually impaired users or into acoustic maps via headphones for blind users. Two prototypes were developed for the validation of the concepts. The first was an acoustic prototype containing a novel time-of-flight CMOS range-image sensor mounted on a helmet, in combination with an audio interface for conveying distance information through a spatialised sound map. The second was a real-time mobility-assistance prototype equipped with several environmental and user interfaces for safe in- and outdoor navigation.

SWAN or System for Wearable Audio Navigation is a project of the Sonification Lab at Georgia Institute of Technology [27]. The core system is a wearable computer with a variety of location- and orientation-tracking technologies, including GPS, inertial sensors, pedometer, RFID tags, RF sensors and a compass. Sophisticated sensor fusion is used to determine the best estimate of the user's actual location and orientation. Tyflos-Navigator is a system which consists of dark glasses with two cameras, a portable computer, microphone, earphones and a 2D vibration array [3]. It captures stereo images and converts them into a 3D representation. The latter is used to generate vibration patterns on the user's chest, conveying distances of the user's head to obstacles in the vicinity. The same authors presented a detailed discussion of other relevant projects concerning navigation capabilities [2].

Similar initiatives exploited other sensor solutions, for example an IR-multisensor array with smart signal processing for obstacle avoidance [1] and a multi-sonar system with vibro-tactile feedback [5]. One system is devoted to blind persons in a wheelchair [7]. Information of the area around the wheelchair is collected by means of cameras mounted rigidly to it. Hazards such as obstacles, drop-offs ahead of or alongside the chair, veering paths and curb cuts can be detected for finding a clear path and maintaining a straight course [18]. All camera information can be combined with input from other sensors in order to alert the user by synthesised speech, audible tones and tactile cues.

From the overview presented above we can conclude that technologically there are many possibilities which can be exploited. Some are very sophisticated, but also very complex and likely too expensive for most blind persons who, in addition to having to deal with their handicap, must make both ends meet financially. Moreover, ergonomically most may prefer not to wear a helmet or to use other visually conspicuous devices which set them apart. Many previous initiatives were very ambitious in the sense that information from many sensors was integrated for solving most problems one can imagine. An additional aspect is that complex systems are difficult to assemble and integrate, and they require maintenance by professional technicians. For these reasons the project "SmartVision: active vision for the blind," funded by the Portuguese Foundation for Science and Technology, is developing two separate modules for global and local navigation which can be integrated if the user desires this.

An initiative similar to the Portuguese SmartVision project is the Greek SmartEyes project [25]. It also addresses global navigation using GPS with a GIS. Vision by two chest-mounted cameras is used to obtain a disparity map for detecting open space and obstacles. This is complemented by two ultrasound sensors mounted next to the cameras.

SmartVision’s functionality for local navigation is very restricted: (1) only path tracking and obstacle detection, and (2) only the space a few metres in front of the user is covered, which is best done by using one or two miniature cameras. Ideally, the cameras – but also a CPU and earphone – could be mounted in dark glasses as in the Tyflos-Navigator system [3]. However, many blind persons are continuously and unconsciously turning their head while focusing on different sound sources. As a consequence, the user should learn to control his head, which may be very difficult and imposes yet another physical and even perceptual limitation on the user, or image processing becomes very complicated because of sudden and unpredictable camera

motions. For these reasons the camera will be attached at chest height, as is done in the SmartEyes project [25], also taking into account that blind persons have learned not to sway much with their body while walking and swaying the white cane in front of them.

As mentioned above, the SmartVision system has two modes of operation. The first, global navigation, employs a GIS with GPS and other localisation devices like active RFID tags for going from some location to a certain destiny [9]. Here we concentrate on local navigation, for centering on paths and in corridors while negotiating both static and moving obstacles. The area covered is in front of the user and just beyond the reach of the white cane, such that the system can alert the user to looming obstacles before his white cane will touch – or miss – them. To this purpose the user is equipped with a stereo camera attached at chest height, a portable computer, and only one earphone such that normal ambient sounds are not blocked; see Fig. 1. Instead of using a blocking earplug, a miniature speaker can be worn behind one ear. The cameras can be cheap webcams which are mounted in a very small tube, and the computer can be worn in a shoulder-strapped pouch or pocket. Both tube and pouch can be made of or covered by a material or fabric which matches the user's clothes.

**Figure 1.** Illustration of the prototype with stereo camera, portable computer and earphone.

The processing chain is depicted in Fig. 2. Although blind users have learned not to sway much their body while walking and swaying the white cane in front of them, the camera attached at chest height will not be very stable over time, i.e., there are cyclic pan and tilt oscillations. Therefore, after a few initial frames the optical flow will be clustered into overall frame motion and object motions. Frame motion will be filtered for motion prediction in order to stabilise new frames such that path detection (in a path-detection window) and detection of static obstacles in front on the path (in an obstacle-detection window) can be adapted in order to free CPU time.

Until here all processing is done by using only one of the two cameras, for example the left one. Then, stereo disparity can be used to estimate distances of static and moving obstacles on the path, as indicated by the left red arrow in Fig. 2. The left frame has already been processed for optical flow on the basis of a compact image representation for solving the correspondence problem of successive frames in time. Since solving the correspondence problem in stereo can be done using the same image representation, the additional processing for distance estimation only involves computing the image representation of the right frame, but only within the path-detection or even the obstacle-detection window in order to limit CPU time. In addition, distance estimation is only required when an obstacle has been detected, and this information is used to modulate the signals of the user interface: the right red arrow in Fig. 2.

Basically, the user interface can create three alerts: alert P for centering on the path, and alerts SO and MO for static and moving obstacles. One solution is to use sound synthesis, for example a pure A tone of 440 Hz for alert P which may increase or decrease in frequency and in volume when the system advises to correct the heading direction to the left or to the right. The spectrum and volume of the sound can also be modulated in the case of detected obstacles, or static and moving obstacles may be indicated by different chirps or beeps. An alternative is to use text-to-speech synthesis with a limited set of small messages. Different solutions are being tested by blind persons in order to find the best one.

It should be stressed that we assume sufficient ambient illumination for image processing. In the case of very low light levels, for example outdoor during the night, special histogram equalisation is required [23]. Also, algorithms for path and obstacle detection are similar to

those used for robot navigation in corridors [28], although our algorithms are optimised for running on a small portable computer.

The rest of this article is organised as follows. In the next section we describe path detection, the path detection window, the adapted Hough space and border detection. In Section 3 the detection of static obstacles within the obstacle detection window is explained. Section 4 deals with optical flow and detection of moving objects. Final conclusions are presented in the last Section 5.

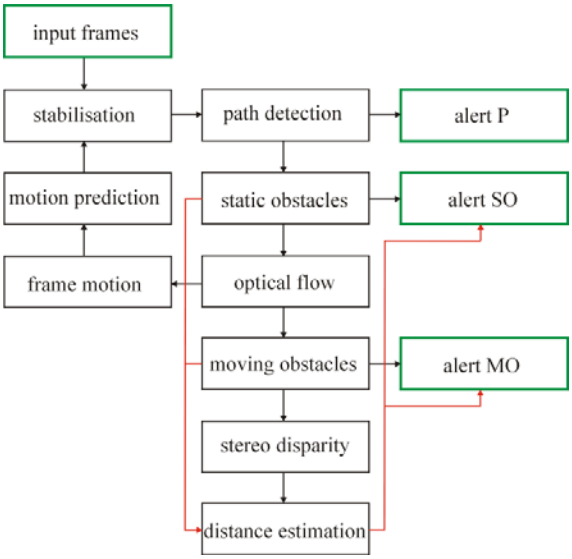


**Figure 2.** Block scheme of the processing. At right the user interface with sounds and/or speech.

## 2. 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 [6] 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 most important edges and to build an Adapted Hough Space (AHS); and (c) the highest values in the AHS yield the borders.

### 2.1. Path Detection Window PDW

Input frames have a fixed width W and height H. 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/2. If the camera points lower or higher, HL will be higher or lower, respectively; see Fig. 3. 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. At the start, HL will be assumed to be at H/2, but after five frames the height of HL is dynamically computed on the basis of previous camera frames after detection of the path borders and the corresponding vanishing points; see below.

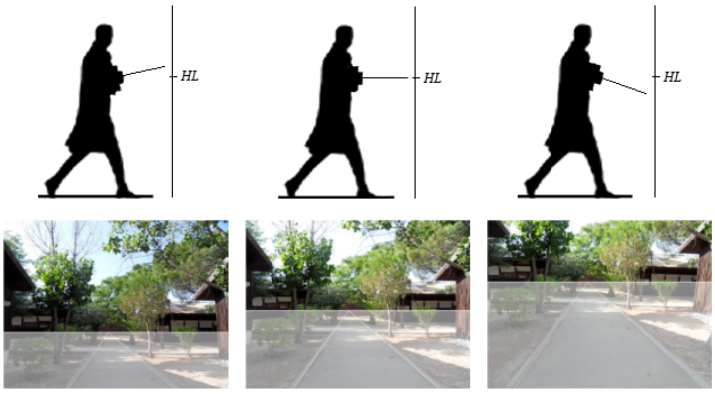


**Figure 3.** From left to right: camera pointing up, horizontally aligned, and pointing down. The Path Detection Window is highlighted in the images.

## 2.2. Adapted Hough Space AHS

The Canny edge detector and an adapted version of the Hough transform are used 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 of the lower part of the frame delimited by HL. Then two iterations of a 3x3 smoothing filter are applied in order to suppress noise.

The Canny edge detector [4] is applied with , which defines the size of the Gaussian filter, in combination with and which are the low and high thresholds for hysteresis edge tracking. The result is a binary edge image with a width of 300 pixels and variable height around 225 pixels in case of the Bumblebee 2 camera. The left part of Fig. 4 shows one original frame together with the resized and lowpass-filtered PDW and detected edges below.

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 [10] 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 and , we use a different reference point. Instead of using the origin at the top-left corner, we use a new origin at the bottom-centre; see the right part of Fig. 4. This simplifies the processing of the left and right image halves and results in the adapted Hough space AHS in polar coordinates as shown in the bottom-right part of Fig. 4. As for the normal Hough space, AHS is a histogram which is used to count co-occurrences of aligned pixels in the binary edge map ($x$, $y$). However, there are two differences. First, almost vertical and horizontal edges cannot be path borders. Hence, the Hough space is restricted to and to 110 on the corresponding sides. This yields a reduction of CPU time of about 30%.

Second, longer sequences of edge pixels count more than short sequences and not-connected edge pixels. To this purpose we use a counter P which can be increased or reset. When checking each pixel of the edge map for a projected line with a certain angle and distance to the new origin and find the 1st ON pixel, P=1 and the corresponding AHS bin will be incremented by 1. If the 2nd pixel is also ON, P increments by 2 to 3 and the bin will increment by 3, and so on. If a next pixel is OFF, P is reset to 0 and the bin is not modified. In other words, a run of connected edge pixels has P values of 1, 3, 5, 7, etc., or with , and will contribute to the AHS bin. An example of an AHS is shown in the right part of Fig. 4 together with magnified regions (bottom). The maxima belonging to the left and right borders are marked in red and green, respectively, also the detected borders in the edge map (top).

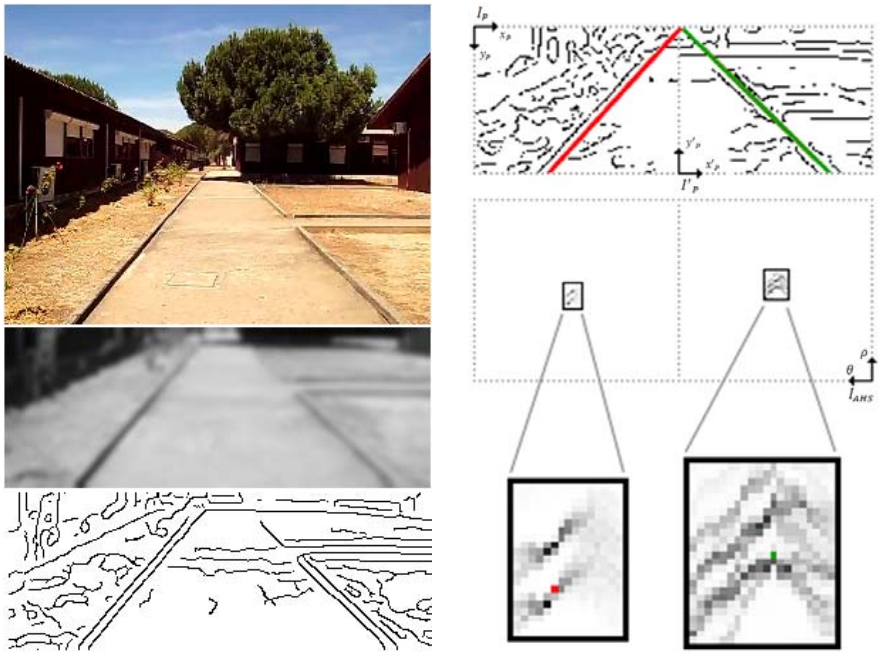

**Figure 4.** Left part: one original frame (top) with resized PDW after low-pass filtering and the binary edge image. Right part, top to bottom: PDW with detected edges and coordinate systems, AHS and zoomed areas. The left and right borders are marked in red and green, respectively.

## 2.3. Path borders

Until here we explained the computation of AHS, but only during the initialisation phase of the first 5 frames. After the initialisation phase, for optimisation and accuracy purposes, we do not check the entire space. Each border $(\rho, \theta)$, 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 AHS. These regions indicate where the next border positions are expected. The two regions are limited by the minimum and a minimum values of $\rho$ and $\theta$ in $M$.

In frames $i \geq 6$, we look for the highest value(s) in AHS in the regions as defined above, but these regions are enlarged in order to cope with camera motion. Hence, the maxima and minima are enlarged by $\pm 10$ for $\rho$ and $\pm 5°$ for $\theta$. This procedure is applied for all $i \geq 6$, always considering the borders found in the previous five frames.

In the two enlarged regions in AHS we look for the highest values. 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 in 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 in 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 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 5 shows results of path and border detection in the case of two image sequences.

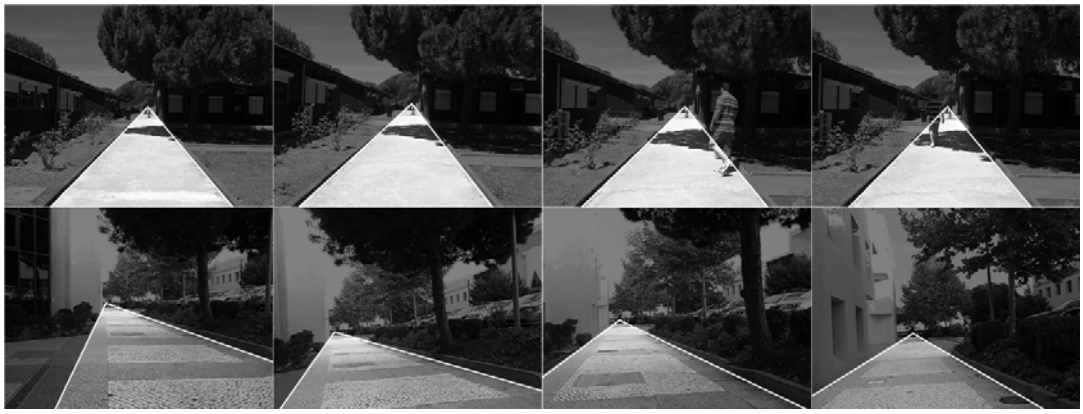

**Figure 5.** Frames of two sequences with detected borders (white lines), the path being highlighted.

## 3. Detection of Static Obstacles

The path window PW defined by the detected path borders and the VP is wider than the area in front where the blind person will walk and where obstacles must be detected. Hence, the PW is first 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), using on both sides 5% of the original width. If a border does not intersect the bottom line, as illustrated in the second line in Fig. 5, then the left or right border of the frame is taken. This results in a narrower triangle. The height of the window can also be reduced because the top of the triangle, near VP, is too far away. Hence, the new height $H'$ is set lower than the height of the VP: $H' = 2/3\, H_{VP}$. This yields a trapezoid with parallel top and bottom lines called the obstacle window OW.

For obstacles in the immediate neighborhood beyond 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. In addition, the resolution at the bottom of the window is higher than at the top. This aspect is important for texture analysis. Therefore, inside the OW we define the obstacle detection window ODW, and use interpolation to correct image resolution. The latter is achieved by mapping the trapezoid (OW) onto a rectangular window with the same width as the OW, line by line, using linear interpolation of the closest pixels.

Then, a resampling to half the width and height is applied to the new rectangular window, after which a Gaussian lowpass filter with a 3x3 kernel is used to reduce noise. This pre-processing is common to all obstacle detection algorithms. Below we explain three algorithms. If at least two of these detect an anomaly in the same region, an obstacle is assumed. At least 3 successive frames are required to confirm the presence of an obstacle, before alerting the user.

### 3.1 Zero Crossing Algorithm

As in [6], we compute derivatives in x and y inside the smoothed rectangle, using a large kernel K= [-1 , -1 , -1 , 0 , 1 , 1 , 1]. Then we sum the amplitudes of all maxima and minima near every zero-crossing: each time the pixel value changes sign, we look for the minimum and maximum value on both sides and sum the absolute values. For analysing 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 window. The resulting two arrays are then smoothed twice with a 7x1 filter 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.

Thresholds are applied to the histograms in order to remove “noise” caused by the texture of the pavement, which can be smooth or structured. The upper and lower thresholds are determined dynamically on the basis of the maximum and minimum values of the computed histograms. During system initialisation, i.e., the first five frames, we must assume that no obstacle is present, but then the system adapts to the actual pavement type and the histograms

reflect local deviations from the pavement structure. For further details we refer to [14]. After thresholding the histograms in x and y, the bins are back-projected into the 2D window for obtaining the size and position of the obstacle.

### 3.2 Histograms of Binary Edges

Canny's edge detector has already been applied for path detection. Here we apply it again, but now to the small ODW, before computing first derivatives in x and y with edge magnitudes and orientations. As in the previous method, these are filtered and histograms are computed. During the first five frames, the maximum value of the edge magnitudes in the entire ODW is used as the high-hysteresis threshold in Canny's edge tracking. After the fifth frame, if the maximum magnitude in only the bottom half of the ODW is higher than the maximum threshold in the previous frame, then this new maximum is taken as the new threshold. Otherwise the average of the old threshold and the new maximum is used. This way the algorithm adapts to the pavement type and insignificant edges are removed.

As we want to determine the region where the obstacle is, we construct an orientation histogram with only two bins using the computed edge orientations: horizontal edges in the intervals from -67.5 to 67.5° and from 112.5 to 247.5°. Vertical edges are in the intervals from 22.5 to 157.5° and from 202.5 to 337.5°. This improves obstacle detection relative to using intervals of multiples of 45°. For locating the region where an obstacle might be, the edge histograms are filled for every line and column in the ODW, i.e., horizontal edges are summed over *y* and vertical ones over *x*. All bins with values below 2 are discarded for a better localisation with clear left-right and top-bottom limits. As in the previous method, thresholded histograms in x and y are back-projected into the 2D window for obtaining the size and position of the obstacle.

### 3.3 Laws' Texture Masks

The third algorithm is based on Laws' texture energy masks [17] applied to the ODW. Again, the idea is to detect changes of the textures. Our tests with real objects and pavements showed that the best masks are E5L5, R5R5, E5S5 and L5S5. After filtering with the masks, a quadratic energy measure with size 11x11 is applied to the result of each mask. The four energy images are then normalised using the maximum energy response which each mask can achieve theoretically, such that each mask contributes equally to final detection. The four normalised energy images are then summed and the result is normalised to the interval 0-255. All values above a threshold of 10 are considered to be due to a possible obstacle. The rest of the processing is equal to the processing as described in the zero-crossing algorithm.

### 3.4. Obstacle Avoidance

If an obstacle is detected (a) in at least 3 consecutive frames, (b) by at least two of the three algorithms in each frame, and (c) with obstacle regions in the ODW whose intersections are not empty, the user will be alerted. Figure 6 shows a sequence in a corridor with the detected path borders and the obstacle window OW. At left, the obstacle approaches the OW, and after entering the OW at least two algorithms have detected it; the white region on the obstacle is a combination of the regions detected by the algorithms. In order to avoid the obstacle, the user is instructed to turn a bit left or right. This is done by comparing the obstacle's region with the open spaces to the left and to the right in the path window. Hence, the user can adapt his heading direction when approaching the obstacle. It should be stressed that the user will always use his white cane in order to check the space in front. Still under development is the interaction between obstacle avoidance and correct centering on the path, such that avoidance does not lead to leaving the correct path.

## 4. Detection of Moving Objects

Apart from detecting path borders and static obstacles on the path, 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 [21], and keypoints provide important image information because they 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 detecting and 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. It should be stressed that step (a) requires much more CPU time than steps (b) and (c). The reason is that many filters (64) are applied. On a normal CPU this takes about one second. However, when using a GPU (Nvidia's CUDA API) this reduces to a fraction of a second. In addition, although in Figs 7, 8 and 9 entire frames are shown for illustrating the algorithms, the processing can be limited to a much smaller part of the frames after a moving object has been detected.

**Figure 6.** A sequence with detected path borders, the obstacle detection window, and a looming obstacle which has been detected.

### 4.1. Keypoint Detection and Annotation

Gabor quadrature filters provide a model of cortical simple cells [21]. 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 the 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 $N_\Theta$ 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 [21]. 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 $R^E_{s,j}$ 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 ($\lambda$). For further details we refer to [12].

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 7 shows keypoint detection and annotation results together with optical flow. At top-left it shows one frame from the sequence shown in Fig. 4, and to the right a combination of two successive frames. The top-right images show keypoints detected at two scales, $\lambda = 6$ (left) and 15 (right). The second row shows, left-to-right annotated keypoints at the two scales, and the displacement vectors between the successive frames at those scales.

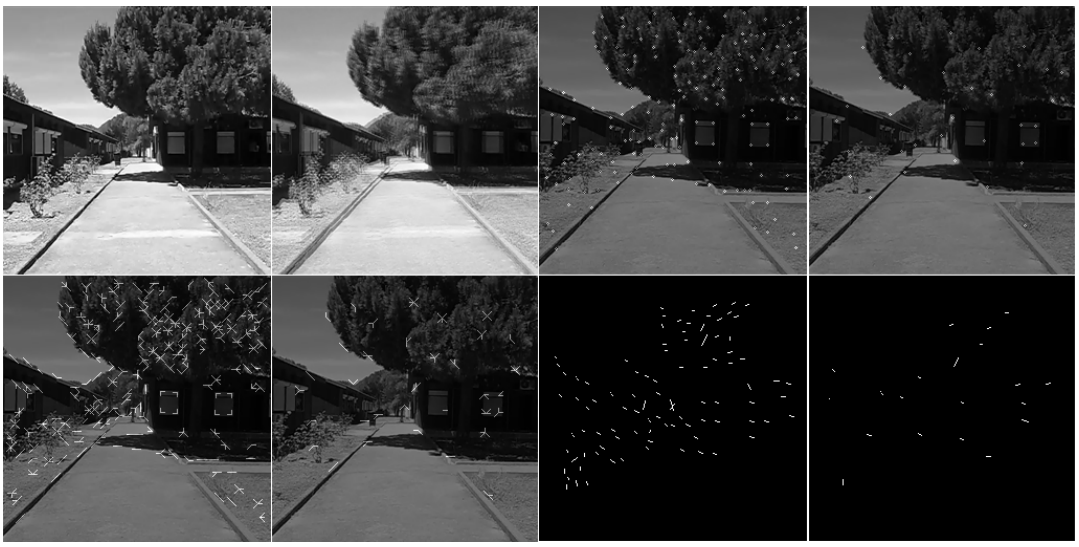

**Figure 7.** Keypoint detection, annotation and matching. Top, from left: one frame of the sequence shown in Fig. 4, two successive frames combined, and detected keypoints at scales $\lambda = 6$ and 15. The bottom row shows annotated keypoints and displacement vectors between the successive frames at the two scales.

### 4.2. 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 $\lambda = 27$, which is a compromise between speed and quality of results. Hence, at the moment all keypoints at $\lambda = 27$ 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 ($\lambda$). 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.

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 $\lambda$ 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, but the matching process is bottom-up: it starts at the finest scale because there the accuracy of the keypoint annotation is better. Keypoint matching is detailed in [12].

Figure 7 shows, at bottom-right, 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. Such outliers can be removed.

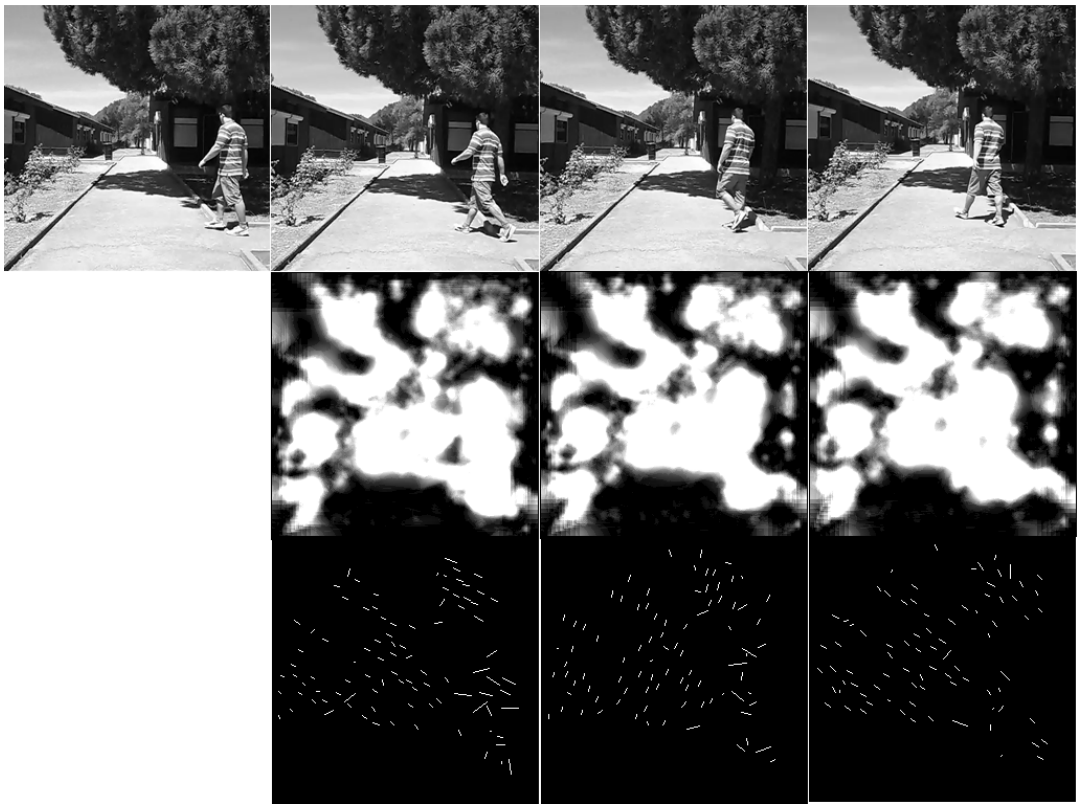

**Figure 8.** A sequence of 4 frames (top) with saliency maps (middle) and optical flow (bottom), both at scale λ=6.

### 4.3. 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.

A saliency map can be based on keypoints as these code local image complexity [21]. 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 already used 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 [21]. Figure 8 (second row) 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 explained above. 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 second row shows saliency maps of the last three frames above, and the bottom row shows optical flow vectors 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 with detected path borders and the bounding boxes. The rightmost images show the tracking of the centre of the bounding box. The tracking 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. As explained in the Introduction (Fig. 2), optical flow can be combined with stereo disparity in order to complement motion with distance estimation.

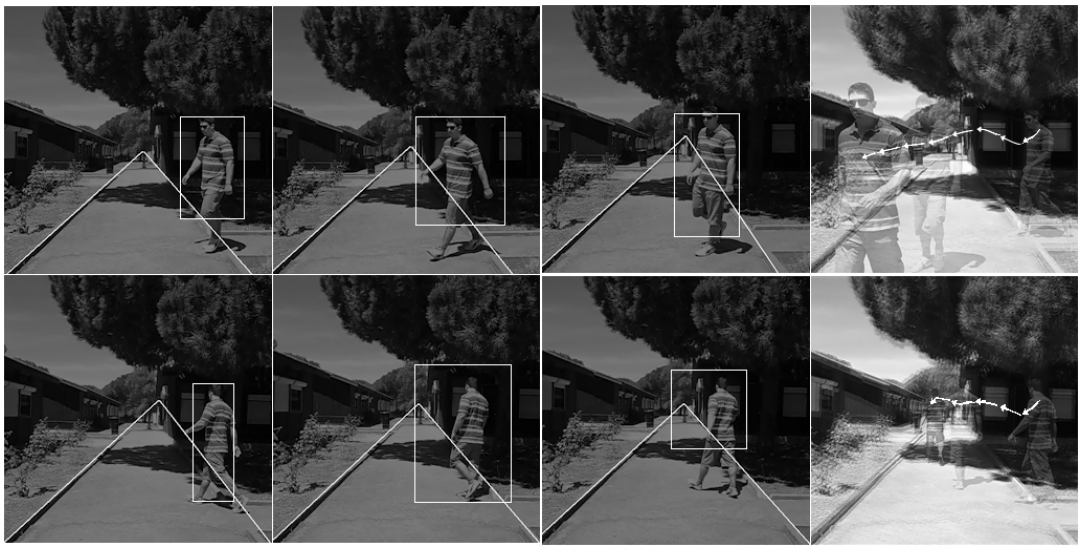

**Figure 9.** Parts of two sequences with tracked moving objects; see text.

## 5. Conclusions

We presented a system for detecting path borders and the vanishing point, such that blind persons can be instructed to correct their heading direction on paths and in corridors. A biologically inspired algorithm for optical flow based on multi-scale keypoint annotation and matching is used. 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. 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 can also employ a GIS with GPS, WiFi and passive as well as active RFID tags [11]. In an already approved follow-up project of two years, algorithms can be further optimised and frame stabilisation, as depicted in Fig. 2, can be implemented. Additional problems can be solved like initial path finding when leaving an office or a building, also path bifurcations and crossing corridors. Extensive field tests are planned with ACAPO, the Portuguese organisation of blind and amblyopes.

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 [19]. 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.

## 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 SmartVision: active vision for the blind (PTDC/EIA/73633/2006).

## References

[1] Bruno Andò, Salvatore Graziani "Multisensor strategies to assist blind people: a clear-path indicator" IEEE Trans. on Instrumentation and Measurement, Vol. 58, No. 8, pp. 2488-2494, 2009.
[2] Nikolaos Bourbakis "Sensing surrounding 3-D space for navigation of the blind" IEEE Engineering in Medicine and Biology Magazine, Vol. 27, No. 1, pp. 49-55, 2008.
[3] Nikolaos Bourbakis, Despina Kavraki "A 2D vibration array for sensing dynamic changes and 3D space for blinds' navigation" In Proc. 5th IEEE Symp. on Bioinformatics and Bioengineering, pp. 222-226, 2005.
[4] John F. Canny "A computational approach to edge detection" IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679-698, 1986.
[5] Sylvain Cardin, Daniel Thalmann, Frédéric Vexo "Wearable system for mobility improvement of visually impaired people" Visual Computer Journal, Vol. 23, No. 2, pp. 109-118, 2006.
[6] David Castells, João M.F. Rodrigues, J.M. Hans du Buf "Obstacle detection and avoidance on sidewalks" In Proc. Int. Conf. on Computer Vision-Theory and Applications, Vol. 2, pp. 235-240, 2010.
[7] James Coughlan, Roberto Manduchi, Huiying Shen "Computer vision-based terrain sensors for blind wheelchair users" Computers Helping People with Special Needs, Springer, LNCS 4061, pp. 1294-1297, 2006.
[8] J.M. Hans du Buf "Responses of simple cells: events, interferences, and ambiguities." Biological Cybernetics, Vol. 68, No. 4, pp. 321–333, 1993.
[9] J.M. Hans du Buf, João Barroso, João M.F. Rodrigues, Hugo Paredes, Miguel Farrajota, Hugo Fernandes, João José, Victor Teixiera, Mário Saleiro "The SmartVision navigation prototype for the blind" In Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI), pp. 167-174, 2010.
[10] Richard Duda, Peter Hart "Use of the Hough transform to detect lines and curves in pictures" Comm. ACM, Vol. 15, No. 1, pp. 11-15, 1972.
[11] José Faria, Sérgio Lopes, Hugo Fernandes, Paulo Martins, João Barroso "Electronic white cane for blind people navigation assistance" In Proc. World Automation Congress, pp. 1-7, 2010.
[12] Miguel Farrajota, João M.F. Rodrigues, J.M. Hans du Buf "Optical flow by multi-scale annotated keypoints: A biological approach" In Proc. Int. Conf. on Bio-inspired Systems and Signal Processing, pp. 307-315, 2011.
[13] Final Activity Report of the EU-funded CASBliP project. http://casblipdif.webs.upv.es.
[14] João José, Miguel Farrajota, João M.F. Rodrigues, J.M. Hans du Buf "A vision system for detecting paths and moving obstacles for the blind" In Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI), pp. 175-182, 2010.
[15] Kentaro Kayama, Ikuko Yairi, Seiji Igi "Detection of sidewalk border using camera on low-speed buggy" In Proc. Int. Conf. on Artificial Intelligence and Applications, pp. 262–267, 2007.

[16] Laehyun Kim, Sehyung Park, Sooyong Lee, Sungdo Ha “An electronic traveler aid for the blind using multiple range sensors” IEICE Electronics Express Vol. 6, No. 11, pp. 794-799, 2009.
[17] Kenneth Ivan Laws “Textured image segmentation” PhD dissertation, USCIPI Report 940, Image Processing Institute, University of Southern California, Los Angeles, 1980.
[18] Xiaoye Lu, Roberto Manduchi “Detection and localization of curbs and stairways using stereo vision” In Proc. IEEE Int. Conf. on Robotics and Automation, pp. 4648-4654, 2005.
[19] Naveen Onkarappa, Angel D. Sappa “On-board monocular vision system pose estimation through a dense optic flow” In Proc. Int. Conf. on Image Analysis and Recognition, Springer, LNCS 6111, pp. 230-239, 2010.
[20] Lisa Ran, Sumi Helal, Steve Moore “Drishti: an integrated indoor/outdoor blind navigation system and service” In Proc. 2nd IEEE Annual Conf. on Pervasive Computing and Communications, pp. 23-30, 2004.
[21] João M.F. Rodrigues, J.M. Hans du Buf “Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection” BioSystems, Vol. 86, No. 1-3, pp. 75-90, 2006.
[22] Byeong-Seok Shin, Cheol-Su Lim “Obstacle detection and avoidance system for visually impaired people” In Proc. 2nd Int. Workshop on Haptic and Audio Interaction Design, Springer, LNCS 4813, pp. 78-85, 2007.
[23] Yang Shubin, He Xi, Cao Heng, Cui Wanlong “Double-plateaus histogram enhancement algorithm for low-light-level night vision image” JCIT: J. of Convergence Information Technology, Vol. 6, No. 1, pp. 251-256, 2011.
[24] Jason Stewart, Sara Bauman, Michelle Escobar, Jakob Hilden, Kumud Bihani, Mark W. Newman “Accessible contextual information for urban orientation” In Proc. 10th Int. Conf. on Ubiquitous Computing, pp. 332-335, 2008.
[25] Nikolaos Trichakis, Dimitrio Bisias, Styliani Taplidou, Eleni Korkontzila, Leontios Hadjileontiadis “SmartEyes: an enhanced orientation and navigation system for blind or visually impaired people” IEEE Computer Society, CSIDC 2003 Final Report, pp. 1-20, 2003.
[26] Iwan Ulrich, Johann Borenstein “The GuideCane - Applying mobile robot technologies to assist the visually impaired” IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans, Vol. 31, No. 2, pp. 131-136, 2001.
[27] Jeff Wilson, Bruce Walker, Jeffrey Lindsay, Craig Cambias, Frank Dellaert “SWAN: System for wearable audio navigation” In Proc. 11th IEEE Int. Symp. on Wearable Computers, pp. 91-98, 2007.
[28] Zhengyin Zhou, Tianding Chen, Di Wu, Changhong Yu "Corridor navigation and obstacle distance estimation for monocular vision mobile robots" JDCTA: Int. J. of Digital Content Technology and its Applications, Vol. 5, No. 3, pp. 192- 202, 2011.