Multiview 3D Pose Estimation of a Wand for Human-Computer Interaction

X. Zabulis, P. Koutlemanis, H. Baltzakis, and D. Grammenos

Institute of Computer Science - FORTH, Herakleion, Crete, Greece

Abstract. A method is proposed that visually estimates the 3D pose and endpoints of a thin cylindrical physical object, such as a wand, a baton, or a stylus, that is manipulated by a user. The method utilizes multiple synchronous images of the object to cover wide spatial ranges, increase accuracy and deal with occlusions. Experiments demonstrate that the method can be applied in real-time using modest and conventional hardware and that the outcome suits the purposes of employing the approach for human computer interaction.

1 Introduction

Ubiquitous computing and ambient intelligence have introduced more natural ways of interacting with computers than the conventional keyboard and mouse. Recent trends in human computer interaction indicate the plausibility of tangible and natural interaction through modest hardware. The *Nintendo Wii* was the first popular system to provide this functionality, based on acceleration measurements and visual tracking of LEDs. The *Sony PlayStation Move* visually tracks the 3D location of luminous spheres, while the *Microsoft Kinect* sensor employs depth maps to infer user limb locations. Though used in everyday life, pointing objects such as a wand or a baton have not pervaded in such interfaces despite the fact that, aside location, they also convey orientation information.

This work aims to provide a means of explicit interaction that is based on visually tracking a thin cylindrical rod manipulated by a user in 3D space, by estimating its location and orientation (pose) in real-time; henceforth, we call this object a wand. To estimate its pose, without any assumptions on its size, at least two synchronous views are required. More views can be utilized to increase accuracy, treat occlusions, and cover wider areas. To deal with various environments two wand detection approaches are proposed, one based on color and another on luminous intensity; in the latter case the wand is a light source. Intensity based detection is simpler, but requires instrumentation of the wand.

The remainder of this paper is organized as follows. In Sec. 2 related work is reviewed. In Sec. 3 an overview of the proposed method is provided, which is analytically formulated in Sec. 4. In Sec. 5, experiments which evaluate the accuracy, performance and usability of the approach are presented. In Sec. 6, conclusions are provided.

2 Related Work

In the domains of ubiquitous computing and ambient intelligence, physical objects are blended in interaction comprising “tangible interfaces” [1]. In this work, the user-interface item is decoupled from the services that it may provide. This study concerns the applicability of a wand as an item of explicit, real-time interaction. Having such a means, dedicated platforms can be then employed to employ such items in system interaction [2,3].

The need for a pointing device as an interaction item is underscored in [4], where a PDA is visually tracked to emulate a wand in an augmented reality environment. Multiview tracking of markers on the PDA provides pose information for augmenting the virtual wand, whereas this work employs a physical wand. This need is also found in efforts to capture human pointing gestures, i.e. [5], a task that, to date, has not been fully achieved. To the best of our knowledge a visually tracked wand in 3D has not been proposed as a means of interaction. The most relevant work is [6], where a pipette is tracked in from a single view to indicate points on a plane. Markers are utilized to track the wand, at a relatively slow rate ($4\,Hz$).

The geometrical basis of the proposed approach is the reconstruction of a straight 3D line segments from multiple images. We follow conventional approaches [7] in multiview geometry, to combine its multiple observations. An early approach to the problem is [8], formulated for 3 views. In [9] lines are only affinely reconstructed. More pertinent is the approach in [10], but which is iterative manner, as the goal is to match multiple segments. The approach in [11] is also relevant but assumes the existence of multiple interconnected line segments to detect endpoints, information which is not available in our case. In contrast to [12], we cannot assume a short baseline as the user may be performing rapid motions. We do not employ stereo [13], as it yields inaccuracies on thin objects.

3 Method Overview

A wand of unknown size is imaged from multiple views and may be occluded totally or partially in some. In each view, it is segmented and modeled as a line segment that approximates[1] the projection of its major axis on the image plane. When segmentation is successful in 2 views, the object’s size and pose can be estimated. If more views are available, they are combined to increase accuracy.

A synchronized and calibrated multicamera system is assumed. Each camera i is located at $\boldsymbol{\kappa}_i$ and has a projection matrix P_i. The image from each camera is compensated for lens distortion directly after its acquisition, forming image I_i. The output is the 3D line segment, represented by its endpoints $\boldsymbol{e}_{1,2}$. The main steps of the proposed method are the following (see Fig. 1):

[1] Due to perspective distortion, this projection does not coincide with the medial axis of the 2D contour, but for thin objects we assume it is a reasonable approximation.

1. Segmentation. Each image I_i is binarized into image M_i to extract the wand. Segmentation may contain errors, such as spurious blobs, while the wand may not be fully visible due to occlusions.
2. 2D modeling. The wand is sought in each M_i, using the Hough Transform (HT) [14], which yields 2D line l_i. Input to HT is provided by a thinning of M_i. A line segment grouping process upon l_i determines the endpoints of the wand in I_i.
3. 3D pose estimation. The line L where the segment lies is estimated, as the one minimizing reprojection error to the observed line segments. Endpoint estimates are obtained from their 2D observations. In this process, outlier elimination is crucial as, due to occlusions and segmentation errors, the object may not be fully visible in all views.
4. Motion estimation, improves accuracy and robustness.

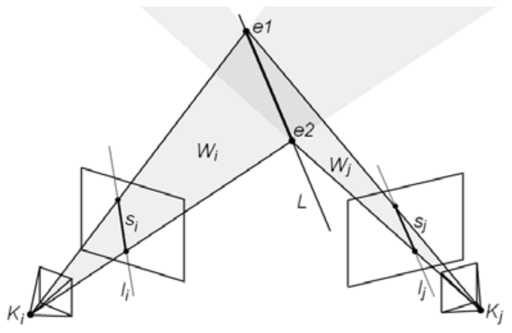

Fig. 1. Method overview (see text)

As the method aims to support real-time interaction, computational performance is of significance. Due to the large amount of data provided by multiple views, we strive for massive parallelization. Thus, techniques are formulated to be executed in parallel on data that are granuously partitioned. For the same reason, single-pass techniques are preferred over iterative ones. The CPU is pipelined at the end of operations to perform sequential processing, which is applied on very few data, in the tasks of outlier elimination and motion estimation.

4 Method Implementation

4.1 Image Acquisition and Segmentation

Acquired images are uploaded from RAM to the GPU. For color images an additional byte per pixel is added to achieve coalesced memory access and efficient use of the texture cache. This byte also facilitates optimal execution of image interpolation on GPU hardware, which is employed to compensate for lens distortion and is applied immediately after acquisition to provide image I_i.

Next, I_i is segmented into binary image M_i where, ideally, pixels have the value of 1 if they image the wand and 0 otherwise. Segmentation errors which may occur are treated in subsequent steps. Depending on setup, segmentation can be based on color or intensity; Fig. 2 demonstrates two such results. Both segmentation versions are parallelized per pixel.

Using a wand of characteristic *color*, selected to be scarcely encountered in the scene, M_i is obtained as follows. A color similarity metric [15], robust to variations of illumination conditions, yields a similarity score per pixel. Each result is thresholded to generate binary image M_i. Using a bi-colored wand (see Table 1) the direction of the wand (besides orientation) can be disambiguated. In this case, the color similarity metric is applied twice, once for each color, and the results are merged in M_i.

A luminous wand is segmented by *intensity* thresholding I_i to obtain M_i. This approach is more robust to illumination artifacts and accidental color similarity. Practically, a brief shutter time, i.e. Fig. 2(right), suffices for accurate segmentation of moderately luminous objects.

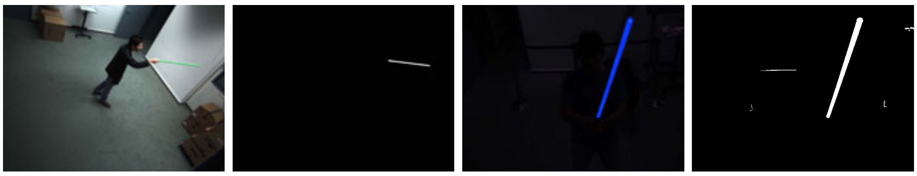
Fig. 2. Image segmentation. Examples of images I_i and segmentations M_i, for color (left) and intensity (right).

4.2 2D Wand Modeling

The output of this step is line segment s_i, approximating the projection in I_i of the wand's major axis. This is achieved, first, by thinning M_i to obtain image T_i. Then, the HT on T_i estimates the line l_i in I_i, that this projection lies. Finally, a grouping process upon l_i determines s_i. All tasks are performed in the GPU.

Thinning. This process performs a thinning of M_i so that an elongated foreground blob, such as the wand's segmentation, is reduced to (an approximation of) its major axis image projection. Due to perspective projection, the wand does not exhibit constant thickness in I_i and, thus, a different amount of thinning is required at each image locus. To parallelize computation, a single-pass operation is employed, which estimates wand thickness at each image point and applies the proportional amount of thinning (see Fig. 3).

First, M_i is convolved with a disk kernel D of a diameter large enough to be "thicker" than the wand in M_i. This results in image Q_i. Pixels of M_i that were 0 are set to be 0 also in Q_i. A priori, the convolution response of D for a range of line widths is computed and stored in a lookup table. Using this table, each pixel in Q_i is converted to an estimate of wand thickness, at that locus. In essence, this is an optimization of scale-selection [16], outputting the distance of each pixel to its closest boundary.

Next, convolution of M_i with a wide Gaussian kernel provides image S_i. In S_i, the wand appears as a smooth intensity ridge. Gradient ∇S_i is then calculated. The thinned image is obtained through a non-maximum suppression process along the direction of ∇S_i, that is applied on image $|\nabla S_i|$. For each pixel $\boldsymbol{p}$ that $M_i(\boldsymbol{p}) \neq 0$, the values of S_i along the direction $\nabla S_i(\boldsymbol{p})$ are suppressed if $M_i(\boldsymbol{p})$ is a local intensity maximum along the direction of $\nabla S_i(\boldsymbol{p})$ in S_i. The spatial extent of this suppression is equal to the local width of the wand, as provided by $Q_i(\boldsymbol{p})$. That is, pixel $T(\boldsymbol{p})$ is 1 if $S(\boldsymbol{p}) > S(\boldsymbol{p} + \alpha \cdot \boldsymbol{v})$ holds for all α and 0 otherwise, where $\alpha \in [-Q_i(\boldsymbol{p}), ..., -Q_i(\boldsymbol{p})] - \{0\}$.

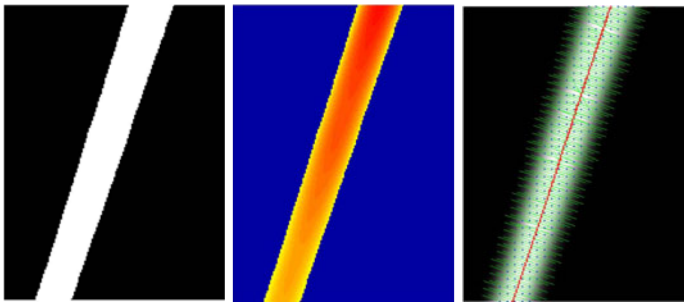

Fig. 3. Image thinning on a detail of the image in Fig. 2(right). Left to right: M_i, Q_i (warmer colors indicate greater sizes), and $|\nabla S_i|$ with gradient direction vectors and the thinning result superimposed. The length of the plotted (green) vectors matches the corresponding size-estimate in Q_i and indicates the spatial extent of nonmax suppression. The resulting "1" pixels of T_i are superimposed as red dots.

Line estimation. Pixels marked as 1 in T_i are passed to the HT, to estimate l_i. For each pixel $\boldsymbol{p}$ in T_i that $T(\boldsymbol{p}) = 1$, corresponding locations in Hough-space are incremented by 1. This is performed in parallel for each pixel, but since concurrent threads may access the same pixel in Hough space, operations are serialized through atomic operations. The global maximum $\boldsymbol{p}_m$ in Hough-space determines l_i and is passed to the next processing step.

2D line segment detection. Due to occlusions and segmentation errors, the wand does not appear as a single segment along l_i, while spurious blobs may also be encountered. A traversal of M_i is performed along l_i and connected components are labeled. Very small segments are attributed to noise and are disregarded. Size-dominant segments along l_i are grouped if they are separated by a distance smaller than τ_d; the value of τ_d is set by adapting the line grouping metric in [17] for the 1D domain of line l_i. The longest detected segment is selected and its endpoints identified. If a bi-colored wand is employed, the matched color of each point is stored. If the length of the resulting segment is very small, the wand is considered not to be detected. The process is demonstrated in Fig. 4.

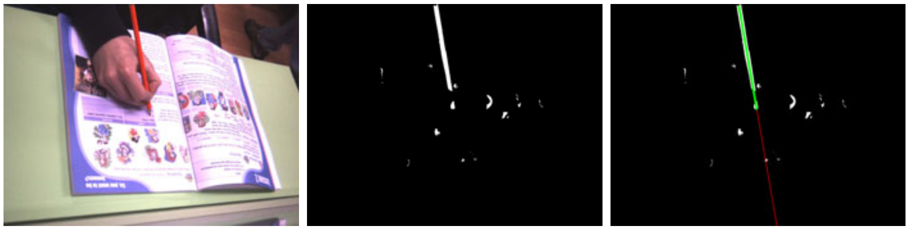

Fig. 4. Line segment detection. Original image I_i (left), M_i (middle), and M_i with l_i and s_i superimposed (right); the red line is l_i and the green segment is s_i.

4.3 3D Line Segment Estimation

First, the line L where the 3D segment lies is estimated and then its endpoints $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ are estimated. When views are more than 2, the problem is overdetermined and an optimization approach is adopted to increase accuracy.

Line estimation. For each view j that a line segment s_j is detected, we define a plane W_j (see Fig. 1). This plane is defined by the camera center $\boldsymbol{\kappa}_j$ and two image points on s_i. The 2D endpoints of the segment can be used for this purpose, as it is of no concern whether the wand is fully imaged. Their 3D world coordinates on the image plane are found, as the intersection of the rays through these points with that plane (see [7], Eq. 6.2.2).

When the wand is detected in 2 views, $j = 1, 2$, L is the intersection of W_1 and W_2. If the wand is detected in $n > 2$ views, plane W_j for each view is considered. Ideally, planes W_j should intersect on the same 3D line, however due to noise and calibration errors this is hardly the case. Hence, the following optimization process is followed. Planes W_j are represented in a $4 \times n$ matrix A. Each row of A represents a W_j plane, containing the parameters for its 4 equation parameters. Let $A = UDV^T$ the Singular Value Decomposition of A. The two columns of V providing the 2 largest singular values span the best rank 2 approximation to A (see [7], p323) and are used to define L. The Maximum Likelihood estimate of L is found by minimizing a geometric image distance between its image projection in image j and the measured line segment s_j, in all I_j. A geometric distance metric for line segments is adapted from [18] and, in our case, formulated as $d = (d_1^2 + d_2^2)^{1/2}$, where $d_{1,2}$ are the 2D point-to-line distances between the endpoints of s_i and the 2D projection of the considered line (candidate L). This provides L, the line minimizing the sum of distance errors, between its projections and line segments s_j.

Endpoint estimation. A pair of views, say (k, j), is required to obtain an estimate of the wand's endpoints, $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. We consider the 2D endpoints in view k and the rays from $\boldsymbol{\kappa}_k$ through these endpoints. The corresponding two intersections of these rays with W_j provide an estimate for $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ each. The task is performed for all pairs where $j \neq k$, providing multiple point estimates.

The 3D estimates are then clustered by a connected component labeling process: two points are assigned with the same label if they are closer than τ_a. We assume that the two clusters with the greatest cardinality correspond to the endpoints and the, potentially, remainder points to the outliers; besides noise, an outlier may be due to the fact that the wand is not fully imaged in some view. The images of the inliers from each cluster are triangulated, using Maximum Likelihood Triangulation, to obtain the reconstruction of each endpoint. The endpoint estimates $\boldsymbol{e}_{1,2}$ are the projections of these points on L. For a bi-colored wand each point is associated with a color (see Sec. 4.2) and, thus, 3D estimates are associated to physical endpoints.

4.4 Motion Estimation

Tracking improves the accuracy of pose estimation and corrects errors, i.e. when the wand is transiently lost or pose estimation is extremely inaccurate. The trajectory of the wand is tracked over time in the 6D space using a Kalman filter. To implement the filter we have assumed a 12D state vector $\mathbf{x}(t)$ given as:

$$\mathbf{x}(t) = [\mathbf{p}(t); \mathbf{a}(t); \mathbf{p}'(t); \mathbf{a}'(t)]^T \tag{1}$$

where $\mathbf{p}(t) = [p_x(t),\ p_y(t),\ p_y(t)]$ is the wand's first end-point, $\mathbf{a}(t) = [a_x(t),\ a_y(t),\ a_y(t)]$ is the normalized direction vector pointing to the second point and $\mathbf{p}'(t)$ and $\mathbf{a}'(t)$ are the corresponding derivatives with respect to time (the speed components). The state vector $\mathbf{x}(t)$ is not directly observable. Instead, at each time instant t, we observe vector $\mathbf{y}(t) = [\mathbf{p_1}(t); \mathbf{p_2}(t)]^T$ which is our 6D measurement vector and which consists of the Cartesian coordinates of the two endpoints of the wand $\mathbf{p_1}(t)$ and $\mathbf{p_2}(t)$. The resulting state-space model is described by the following equations:

$$\mathbf{x}(t) = \mathbf{F}\mathbf{x}(t-1) + \mathbf{w}(t) \tag{2}$$

$$\mathbf{w}(t) \sim N(0, U(t)) \tag{3}$$

$$\mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{v}(t) \tag{4}$$

$$\mathbf{v}(t) \sim N(0, C_y(t)) \tag{5}$$

where $\mathbf{w}(t)$, $\mathbf{v}(t)$ are independent, zero-mean Gaussian processes with variances $U(t)$ and $C_y(t)$, representing the transition and the observation noise at time instant t, respectively. $\mathbf{F}$ is the state transition matrix which is used to propagate the current state to the next frame and is selected to satisfy the following:

$$\mathbf{x}(t) = \mathbf{F}\mathbf{x}(t-1) = [\mathbf{p}(t-1)+\mathbf{p}'(t-1); \mathbf{a}(t-1)+\mathbf{a}'(t-1); \mathbf{p}'(t-1); \mathbf{a}'(t-1)]^T \tag{6}$$

$\mathbf{H}$ is the observation matrix which implements the relation of the hidden state with the measurement vector:

$$\mathbf{y}(t) = H\mathbf{x}(t) = [\mathbf{p}(t); d\mathbf{a}(t)]^T \tag{7}$$

where d is the is the length of the wand, estimated from previous frames. The state vector $\mathbf{x}(t)$ and its 12×12 covariance matrix $\mathbf{C_x}(t)$ are estimated recursively using the Kalman Filter equations [19].

5 Experiments

To evaluate accuracy in different spatial scales, as well as, scalability with respect to number of views, the method has been tested in the following setups:

1. A 7-camera cluster in a $5 \times 5\, m^2$ room. Cameras are mounted at the ceiling, viewing it peripherally from a height of $\approx 2.5\, m$, through a $66° \times 51°$ FOV.
2. A trinocular system installed $40\, cm$ above an enhanced school desk. The maximum baseline is $71.5\, cm$ and cameras verge at the center of the table, configured at a FOV of $43° \times 33°$.

Image resolution was 960×1280 pixels, except when modulated to measure effects in accuracy and computational performance. The computer hosting these cameras employed an *nVidia GeForce GTX 260* $1.2\, GHz$ GPU.

5.1 Accuracy and Computational Performance

To the best of our knowledge, there is currently no publicly available multiview dataset for pose estimation of a wand, annotated with high-precision ground truth data. Thus, such a dataset was created [20]. The dataset was collected using an $58\, cm$ wand, mounted on a tripod with 2 degrees of freedom (pitch, yaw) and marked rotation gratings. The dataset sampled a wide range of poses, consisting of 36, 360°-yaw rotations, in steps of 10°. The pitch angles of these rotations ranged from $-70°$ to $80°$, in steps of 10°. Occlusions were present, as in some views the wand was occluded by the tripod. To study the effects of resolution and number of views in the accuracy of estimates, they were modulated as shown in Table 1. We conclude that the method is sufficiently accurate for purposes of indicating points in space and that accuracy gracefully degrades to the reduction of input data. We observe that the results for 7 views are marginally more accurate than those for 4 views. Thus, in this setup, utilization of more than 4 views provides an advantage only in the presence of further occlusions.

Table 1. Left: Indicative image from the dataset used for accuracy estimation. Right: Mean error and standard deviation results.

	480 × 640		960 × 1280	
Views	Yaw	Pitch	Yaw	Pitch
2	2.0° (3.1°)	1.0° (1.4°)	1.4° (4.1°)	0.8° (1.9°)
3	1.2° (1.1°)	0.7° (0.7°)	1.2° (1.1°)	0.7° (0.7°)
4	1.4° (1.6°)	0.9° (1.2°)	0.9° (1.0°)	0.6° (0.7°)
7	1.2° (1.2°)	0.8° (0.9°)	0.9° (1.0°)	0.6° (0.6°)

We performed two experiments to measure the performance of the method. First, for each step of the method, GPU execution time was measured and averaged over a time period of 1000 frames and compared to a reference CPU implementation, for 4 views of 960×1280 resolution; see Table 2, (left). Second,

Table 2. Performance measurements. Left: Execution time for each computational step. Right: Total execution time for different number of views and image resolutions.

Computational Step	CPU	GPU	Speedup
Lens distortion compensation	17.3 ms	1.3 ms	13.3
Image segmentation	230.8 ms	1.2 ms	192.3
Smoothing	20.6 ms	2.5 ms	8.2
Thickness estimation	44.5 ms	4.0 ms	11.1
Non-max suppression	8.1 ms	1.1 ms	7.4
Line detection	38.8 ms	7.0 ms	5.5

Views	480×640	960×1280
2	30 Hz	15 Hz
3	30 Hz	10 Hz
4	22 Hz	7 Hz

we measured performance while modulating the number of views and resolution; see Table 2 (right). We observe that the method is efficiently parallelized in the GPU and that it linearly scales with the amount of input.

5.2 Usability Studies

In order to test the usability, accuracy and response time of the method as perceived by end-users, 3 pilot applications were implemented. Characteristic images from these experiments are shown in Fig. 5.

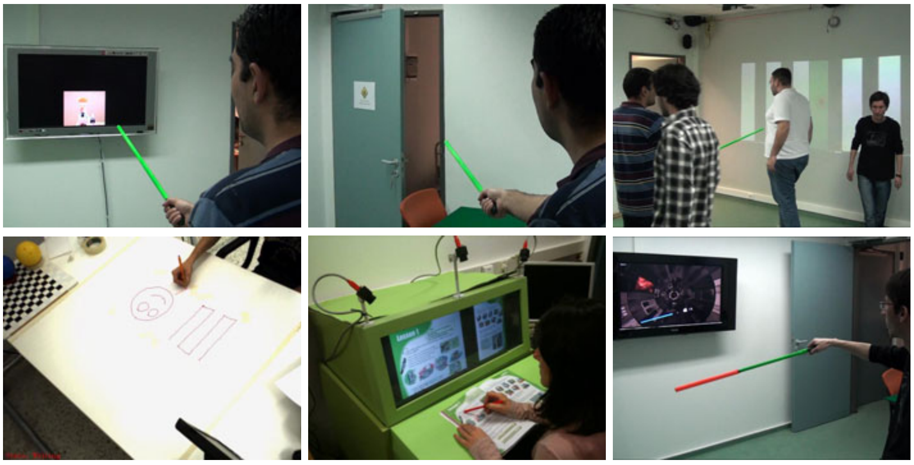

Fig. 5. Images from usability experiments. Top: snapshots from the "room" experiment; right image shows the piano example. Bottom: (i, left) A user draws shapes by bringing the stylus in contact with a desk and dragging it; drawn shapes are virtual and superimposed on an original image from a system's camera, as the projections of the points of contact with the surface. (ii, middle) A user brings a stylus in contact with predefined page regions of a book, to receive content-sensitive information. (iii, right) Image from the "game" experiment, where a player controls a hypothetical saber in the virtual world rendered on the screen.

A **room control** application was created using the first setup. Each wall in the room provides visual output through a projector or a TV screen. On the ceiling, there are 9 computer-controlled RGB spotlights. A computer controlled door (open/close) exists on a wall. Initially, the 4 available screens show a paused video, spotlights are off, and the door is closed. A test participant and an observer enter the room through another door. The observer picks up a 2-color wand ($\approx .5\,m$ long) from the floor and showcases its use. When the wand is pointed at a screen, the respective video starts playing. When the wand stops pointing at it, the video pauses. If the wand points at any of the spotlights, then it turns green and if another was previously lit, it turns off. When the wand points at the door, a knocking sound is heard. If the wand remains pointing the door for $1\,s$, it opens if closed, and vice versa. After the demonstration, the wand is given to the test participant who freely experiments with it. To test more fine-grained actions, a piano keyboard is projected on a wall that can be played by pointing at its keys. A colored dot is projected on the wall at the position where the wand is (estimated to be) pointing at, to provide feedback.

Game. Using the first setup, a game was developed. The goal was to determine whether the system's accuracy was sufficient and its latency small enough to support dexterous and rapid interaction. The user stands in front of a TV screen, using a $58\,cm$ wand as a saber. The system captures the pose of the wand and reconstructs it within a 3D gaming environment, which is rendered on the screen. Using the wand, the user controls the virtual saber to "hit" incoming targets in the form of spheres coming towards him/her.

Desk. Using the second setup we employed the method to track a $14\,cm$ stylus in order to provide interaction of a user with a (i) planar surface and a (ii) physical book. The, corresponding, goals of the experiment were to determine whether the system is sensitive enough to detect the contact of the stylus with the surface and whether the system could be used to indicate regions of interest within pages of the book. In the second case (ii), an additional system [21] recognizes book pages and provides the 3D structure of the book page. The 3D endpoints of the stylus are monitored and when one is approximately in contact with the (i) planar surface or (ii) the book, a pertinent event is triggered.

Discussion. After running several sessions with more than 20 users a number of positive and negative aspects of the system started emerging, which will be more formally tested in subsequent evaluation sessions.

Positive aspects of the system included the following. First, accuracy and response were considered to be adequate for the type of tasks that the participants experimented with. The "desk" experiment yielded error $< 3\,mm$, as to the detection of contact with a surface. Also, employing a non-technological object for interacting with the environment made a very positive impression. The ease of use was deemed high, as it was intuitive and obvious. Finally, participants liked that a single (yet simple) object was be used to control diverse technologies.

On the negative side, it was realized that the wand suffers from the "Midas Touch" problem [22]. The user may accidentally issue commands in the environment while moving it towards the intended interaction target. Typically, this problem is overcome through the use of "dwell" time, additional explicit commands (e.g., buttons, switches, speech), or gestures. Also, since the wand should be visible by at least two views, there were room regions (i.e. corners) which were not covered by the system.

6 Conclusion

A method that estimates the 3D pose of a wand, despite occlusions and partial wand appearances, has been described and evaluated. Evaluation of the proposed method demonstrates that it is accurate, robust and intuitive in use and that it can be employed in a variety of user applications. Additionally, a multi-camera dataset annotated with ground truth was compiled and became publicly available, to facilitate the evaluation of similar methods. Future work warrants multiview line segment matching, to support multiuser interaction.

Acknowledgements. This work was supported by the FORTH-ICS internal RTD Programme "Ambient Intelligence and Smart Environments" as well as the European Commission under contract numbers FP7-248258 (First-MM project) and FP7-270435 (JAMES project). Authors thank Manolis I. A. Lourakis and Antonis A. Argyros for fruitful conversations in the formulation of the proposed approach.

References

1. Ishii, H., Ullmer, B.: Tangible bits: towards seamless interfaces between people, bits and atoms. In: CHI, pp. 234–241 (1997)
2. Greenberg, S., Fitchett, C.: Phidgets: easy development of physical interfaces through physical widgets. In: UI Software and Technology, pp. 209–218 (2001)
3. Ballagas, R., Ringel, M., Stone, M., Borchers, J.: istuff: A physical user interface toolkit for ubiquitous computing environments. In: CHI, pp. 537–544 (2003)
4. Simon, A., Dressler, A., Kruger, H., Scholz, S., Wind, J.: Interaction and co-located collaboration in large projection-based virtual environments. In: IFIP Conference on Human-Computer Interaction, pp. 364–376 (2005)
5. Nickel, K., Stiefelhagen, R.: Visual recognition of pointing gestures for human-robot interaction. Image and Vision Computing 25, 1875–1884 (2007)
6. Hile, H., Kim, J., Borriello, G.: Microbiology tray and pipette tracking as a proactive tangible user interface. In: Pervasive Computing, pp. 323–339 (2004)
7. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision (2004)
8. Ayache, N., Lustman, F.: Fast and reliable passive trinocular stereovision. In: ICCV, pp. 422–427 (1987)
9. Quan, L., Kanade, T.: Affine structure from line correspondences with uncalibrated affine cameras. PAMI 19, 834–845 (1997)
10. Baillard, C., Schmid, C., Zisserman, A., Fitzgibbon, A.: Automatic line matching and 3D reconstruction of buildings from multiple views. In: ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery (1999)

11. Martinec, D., Pajdla, T.: Line reconstruction from many perspective images by factorization. In: CVPR, pp. 497–502 (2003)
12. Moons, T., Frère, D., Vandekerckhove, J., Van Gool, L.: Automatic modelling and 3D reconstruction of urban house roofs from high resolution aerial imagery. In: Burkhardt, H.-J., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1406, pp. 410–425. Springer, Heidelberg (1998)
13. Woo, D., Park, D., Han, S.: Extraction of 3D line segment using disparity map. In: Digital Image Processing, pp. 127–131 (2009)
14. Duda, R., Hart, P.: Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM 15, 11–15 (1972)
15. Smith, R., Chang, S.: VisualSEEk: a fully automated content-based image query system. In: ADM Multimedia, pp. 87–89 (1996)
16. Lindeberg, T.: Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. IJCV 11, 283–318 (1993)
17. Lowe, D.: 3D object recognition from single 2D images. Artificial Intelligence 3, 355–397 (1987)
18. Kang, W., Eiho, S.: 3D tracking using 2D-3D line segment correspondence and 2d point motion. In: Computer Vision and Computer Graphics Theory and Applications, pp. 367–380 (2006)
19. Kalman, R.E.: A new approach to linear flitering and prediction problems. Journal of Basic Engineering 82, 35–42 (1960)
20. Koutlemanis, P., Zabulis, X.: (2011), `http://www.ics.forth.gr/cvrl/wand/`
21. Margetis, G., Koutlemanis, P., Zabulis, X., Antona, M., Stephanidis, C.: A smart environment for augmented learning through physical books (2011)
22. Jacob, R.: The use of eye movements in human-computer interaction techniques: what you look at is what you get. ACM Trans. Inf. Syst. 9, 152–169 (1991)