Robust Head Pose Estimation by Fusing Time-of-Flight Depth and Color

Amit Bleiweiss [1], Michael Werman [2]

School of Computer Science, Hebrew University of Jerusalem

Jerusalem 91904, Israel

[1] amitbl@cs.huji.ac.il

[2] werman@cs.huji.ac.il

***Abstract*—We present a new solution for real-time head pose estimation. The key to our method is a model-based approach based on the fusion of color and time-of-flight depth data. Our method has several advantages over existing head-pose estimation solutions. It requires no initial setup or knowledge of a pre-built model or training data. The use of additional depth data leads to a robust solution, while maintaining real-time performance. The method outperforms the state-of-the art in several experiments using extreme situations such as sudden changes in lighting, large rotations, and fast motion.**

I. INTRODUCTION

Head pose estimation is a widely researched problem in computer vision. The accurate computation of a person's head orientation and position is becoming increasingly important as computing power increases, and as cameras are being incorporated into new sectors. For example, head pose is critical in driving applications, allowing the driver to receive cues based on his or her awareness and visual attention. In addition, head pose can be used as input to face recognition systems, a new means of user interaction, or controlling industrial machinery from a distance. Although many different systems have been proposed, this problem still does not have a robust solution which works in real-time. In addition, most of the solutions require either a training phase, or a prepared 3D model in order to provide correct results.

Recently, much research has been conducted using time-of-flight (TOF) depth cameras. The use of their resultant depth map has been used to solve hard problems in computer vision. Furthermore, it has been shown that this depth data can be used to complement other sensor data in order to improve existing results. In this paper we show that the natural fusion of color and depth data can yield much improved results for head pose estimation. In section 2 we give an overview of related work in the field. In section 3 we give a brief description of the TOF sensor and its output. In section 4 we describe our algorithm in detail. In section 5 we state our results and compare to existing methods. In section 6 we conclude and offer some future directions.

II. RELATED WORK

Head-pose estimation is an active research field with many proposed solutions [15]. We classify existing solutions under the following categories:

Statistical Learning Methods A popular method for estimating head pose uses training data in order to classify different poses. Lia et al. [20] used manifold learning to track both head position and facial expressions. De la Torre et al. [7] trained Hidden Markov Models to track different features of the face. Fujimura et al [21] used unsupervised learning and SVR to train a system for detecting different head pose states. Vatahska et al. [29] trained Haar-like features using a neural network scheme to estimate the head pose at each frame. Lepetit et al. [19] trained feature point descriptors for general object pose estimation, including faces, but required a carefully designed classifier based on training images and a 3D face model. Wu et al. [34] used a probabilistic model for each possible pose, and achieved the best results according to a recent survey [6]. While these methods work in many cases, they typically require extensive manual preprocessing in preparing the training data, and large amounts of training data in order to perform well.

Motion-Based Methods In these methods, feature points on the face are tracked at each frame. Ohayon et al [25] detected a number of 3D feature points on an initial model and used those points to track subsequent head rotations. However, this method reports performance of about 3 fps, and thus is not suitable for real-time applications. Wang et al [33] use optical flow to track facial feature points. Several other papers [24,14,37,26] track feature points throughout a sequence. The disadvantage of this method is that it depends on a set of feature points which may be occluded as the head rotates. As the number of tracked features decreases, so does the performance of these algorithms. In addition, drifting typically occurs in all motion-based tracking algorithms, causing recovery to be quite difficult.

Model-Based Methods Another possible solution is using a model-based approach [9,10,8,5] which requires an initial 3D model similar to the person's face shape and texture. This approach uses a synthesis-analysis loop which renders the 3D model and tries to match the rendered result to the real image by manipulating its vertices and texture coordinates. The problem with this approach is that it requires a textured 3D model of the person in the scene, making it hard to scale for practical applications. The model either has to be created by

MMSP'10, October 4-6, 2010, Saint-Malo, France.
978-1-4244-8112-5/10/$26.00 ©2010 IEEE

an artist or generated automatically by an algorithm [2,17,35]. In addition, depending on the size of the model, these methods may require a high-end GPU for rendering the textured synthetic face model.

3D Methods In the past few years, these techniques have been extended to incorporate measured 3D data. Gokturk et al. [11] described a top-view head-tracking system based on a training set of time-of-flight depth images. Walder et al. [32] used a complex dynamic 3D scanner setup to track the head's pose and non-rigid shape, though the algorithm took over 20 seconds to compute per frame. Malassiotis et al. [22] estimated head pose from CLA range data by tracking the nose, but state their technique is highly dependent on the person's face shape and does not perform well for large rotations. Meers et al. [23] used a similar technique on TOF data. Although these techniques have improved previous results, they still lack the robustness necessary for practical real-world applications, mostly due to the noisy nature and low resolution of the recorded depth data.

Recently, research has been focused on fusing TOF depth with color [3,27], stereo [36,13] and IR [16] in order to improve the results. We propose a model-based approach which fuses RGB and time-of-flight depth data in order to provide further improvement. Our method is unique in that the model is based on real 3D data rather than estimated values. In addition, although we are using a model-based approach, there is no need for a prepared model but rather the model is created on the fly per frame.

III. Sensor Description

For our data, we used 3DV Systems' ZCam camera, which simultaneously captures both RGB and depth data using the time-of-flight principle. This gives us a 4-channel image with 8 bits per channel. Although the data is recorded using 2 separate lenses, the RGB image is already warped to match the depth image. The ZCam camera has several options for resolution and frequency, and we used 320x240 resolution captured at 30 fps. The camera has a range of 0.5-2.5m, a 60^o diagonal field of view, a depth accuracy of a few centimeters, and RGB data comparable to a standard webcam.

Unlike other TOF depth cameras (Canesta, PMDTec, Mesa), which use the intensity modulation approach [18], the ZCam has a unique fast optical shutter technology. This approach gives the camera the ability to set the window limits freely, selecting both the front and back limits of the sensor. Intuitively, the process can be imagined as an optical signal that is reflected by objects in the scene and generates a distorted light wall [1]. Other TOF cameras only set the front limit. This allows us to adjust the camera parameters in such a way that we get maximum depth resolution, capturing a high quality face located at 60-70cm (see Fig. 1). Another advantage of the ZCam is that it has larger resolution and does not exhibit the aliasing artifacts inherent in the intensity modulation scheme. Furthermore, the noise in the depth data at such a close distance was constant in space, thus removing the need for a complex noise model. We verified this with various clothing, people, and hairstyles.

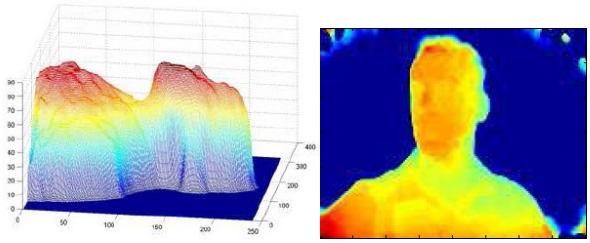


Fig. 1. Example of face depth data resolution acquired by TOF sensor

IV. Head Pose Estimation

We estimate the head pose using an iterative scheme [10,12] which solves for the relative 3D transformation between each pair of frames. First, we find an initial head window *w* using a face classifier [31] and expand the window. Then, for each image in the sequence, we perform the following process (see Fig. 2):

1. Given RGB and depth data from two subsequent frames I_1 and I_2, we estimate their relative translation Δt and rotation ΔR by solving the linear system of equations presented in section 4B, using *w* as a region of interest for deciding which pixels to use in the equations. The resulting solution vector contains the translation and rotation deltas for the current iteration.
2. Transform the 3D face mesh corresponding to I_1 by result translation Δt and rotation ΔR and render a synthetic 2D image $\hat{I}_1$.
3. Compute the residual of the synthetic 2D image $\hat{I}_1$ and actual video image I_2 (as described in Section 4C). As long as the residual error decreases, we continue to iterate, going back to step 1 and swapping the original image I_1 with the latest transformed synthetic image $\hat{I}_1$. Once the error starts to increase, we end the iteration and apply the procedure to the next pair of frames in the sequence.

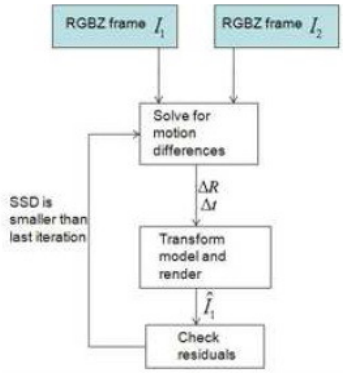


Fig. 2. Algorithm flow chart

A. Face Model

The 3D face model is constructed based on the projected world coordinates supplied from depth data in the rectangular area of the face (see Fig. 3). The rest of the pixels are discarded and treated as noise. The model consists of a set of 3D coordinates $\{ p_0,..., p_n \}$ and a matching set of RGB pixel values $\{ r_0, g_0, b_0, ... r_n, g_n, b_n \}$. The 3D mesh is constructed by creating a triangle for each set of neighboring pixels, and using their respective color data as texture coordinates. A typical generated model consists of an average of 1500-2000 vertices depending on the head's distance from the camera, making it suitable for fast synthesis.

Fig. 3. Textured face mesh used as model for comparison with actual image

B. Proposed Algorithm

We define the 3D motion equations in the following manner:

$$p_1 = R_1 p_0 + t_1$$
$$p_2 = R_2 p_0 + t_2$$

where p_0 is a 3D vertex located on the model, p_1 is that same vertex after a transformation (as seen in image I_1), and p_2 is the same vertex after an additional transformation (as seen in image I_2). As we know the intrinsic parameters of the sensor, applying a perspective projection to the 3D vertex $p_i = [xyz]$ yields the 2D projected coordinate [*XY*]

$$X = -f_x \frac{x}{z}, Y = -f_y \frac{y}{z}$$

where f_x and f_y are the horizontal and vertical focal lengths of the sensor. Using these projected coordinates, we define a 2D displacement between each pixel by

$$d_X = X_2 - \hat{X}_2$$
$$d_Y = Y_2 - \hat{Y}_2$$

In order to combine all these into a single linear system of equations, we include the pixel intensity values using the optical flow equation

$$\frac{\partial I}{\partial X} d_X + \frac{\partial I}{\partial Y} d_Y = \hat{I}_2 - I_2$$

In our case, the input data is 320x240 resolution, and thus computing optical flow for the entire image would result in slow runtime performance [30]. Since our goal was to achieve real-time results, we did not compute the full optical flow, but rather only computed the gradients for selected vertices. The gradients $\frac{\partial I}{\partial X}$ and $\frac{\partial I}{\partial Y}$ are thus computed by using samples from the neighboring pixel's synthetic and actual image value, rather than computing an entire scene flow. Combining the above equations, we end up with a linear system of equations which can be solved in a least squares sense:

$$A[\Delta R_x \Delta R_y \Delta R_z \Delta t_x \Delta t_y \Delta t_z] = b$$

Each pixel contributes 3 equations (one for each of RGB), and *z* is taken from the depth image directly. Once we solve for the deltas, the model is transformed and the next iteration is initiated.

C. Computing the Residual

At each iteration we compute the residual of the input image with the synthetic image. The sum of the result is checked against a threshold value, and then a decision is made whether to continue with an additional iteration, or go to the next frame. We tried several possible options for computing the residual, using different measures combining color and depth sums. In practice, the sum of squared differences of all color channels including z worked best. Thus, at each iteration we compute the following:

$$\hat{\varepsilon} = \sum_{i=0}^{n_w} (\hat{z}_i - z_i)^2 + (\hat{r}_i - r_i)^2 + (\hat{g}_i - g_i)^2 + (\hat{b}_i - b_i)^2$$

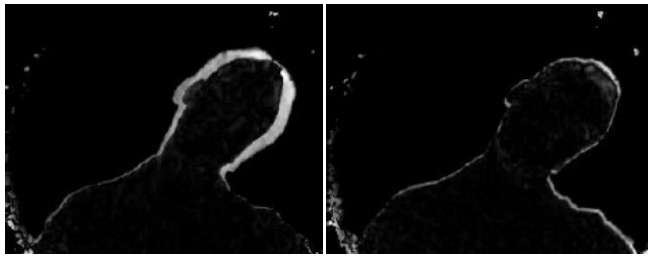


Fig. 4. Examples of residuals computed in between several iterations. It is clear that the right image is closer to convergence than the left image.

D. Outlier Removal

Since least squares is sensitive in terms of outliers, especially for overdetermined systems with a large number of equations, they must be removed in order to provide a robust solution. The majority of outliers are removed automatically as only pixels within the face's window *w* are used in the solution. These outliers are either noisy pixels attributed to the inherent

noise in the depth sensor, or pixels related to non-face regions of the image which do not obey our motion model. In addition, during each iteration of least squares, we remove equations for those pixels which exhibit large motion steps. A rough motion estimation is computed using the following

$$\hat{d}_c(p) = \hat{I}_2(p) - I_2(p)$$

$$\hat{d}_z(p) = \hat{z}(p) - z(p)$$

When these displacements exceed a certain threshold, the pixel's respective equation is removed from the system before the least squares solution is executed.

E. Recovery

The residuals are also used to decide when tracking is lost, and the algorithm needs to recover. When the residual sum is larger than a certain threshold for several frames, it is clear that the algorithm is not converging, and thus we can infer that tracking is lost. In this case, we freeze the algorithm and run the Viola and Jones frontal face classifier [31] until a valid face is identified. The advantage of this approach is that it is easy to identify when the face is looking straight at the camera, at which point it is simple to restart the algorithm and assume a valid state. We use the depth data to optimize performance, as running the classifier on a 320x240 image is slow and prone to false positives as well. Therefore, we use the depth data as a mask and run the classifier only in the area of the color channels corresponding to non-zero pixels in the depth channel. This gives us performance which is 4 times faster on average, and reduces the number of false positives.

V. EXPERIMENTS

It was somewhat difficult to compare with other methods, as our data is unique in that it contains the TOF depth data as well, and we focused on real-time performance. No other head depth and RGB data is made publicly available as far as we have seen, so that limited us to checking color-only trackers and ignore the depth for other methods. The color tracker we used is very similar to other real-time model-based systems [10,12] papers. Also, stereo depth is quite different from the nature of TOF depth data, and thus the same algorithms cannot be compared (stereo had "holes" of unknown data, depth has range problems, etc.). We tested our algorithm on a set of challenging sequences, including extreme rotations, abrupt lighting changes, and fast motion. The algorithm is able to accurately track the head throughout these sequences. In the cases when tracking is lost, a quick recovery is accomplished within a few frames. The depth data is used as a mask for running the face classifier on a smaller region of interest rather than on the entire image, thus enabling us to recover in real-time. These complex sequences exhibited the advantage gained by fusion with time-of-flight depth. In addition to running our tracker on each sequence, we compared the results with model-based trackers using only color or time-of-flight depth independently, which yielded worse results in all cases. The color-only tracker used the algorithm described in Section 4B, but used a pre-built model based on the first frame, as we do not have any real z data for generating the 3D face model on the fly per frame. In addition, the $\hat{z}$ in Equation (5) was estimated from the depth buffer of the rendered result rather than real measured depth data. The depth-only tracker used a simplified form of Equation (5), as the scalar values of the depth data were used instead of color data. This also led to a much smaller linear system, as each pixel contributes a single equation.

Extreme rotations Most motion-based methods do not perform well on extreme rotations, such as profile views of the face. In this case, many important facial features are occluded, and thus can no longer be tracked. In addition, the side of the face is generally smooth, and thus is not suitable for optical flow type tracking. Finally, as the face rotates, its color changes based on diffuse reflection. Fusing color and depth allows us to overcome these problems. Since the model is updated at each frame, its depth and color reflect its current orientation, and thus it always has the highest quality data based on the camera's current view. We recorded a sequence containing eight extreme head rotations in all directions, and compared the residual errors of our method with tracker based on color only and TOF depth only. Both of these trackers yielded worse results, and lost tracking quickly as soon as the face rotated to a profile view. Compared to our method, the color tracker's error rate was five times higher and the depth tracker's error rate was four times higher on average. This is most likely due to lost features in the color tracker, and noisy pixels in the depth tracker. By fusing color and depth, our method was able to achieve a constant low error rate and maintain tracking throughout the entire sequence (see Fig. 5).

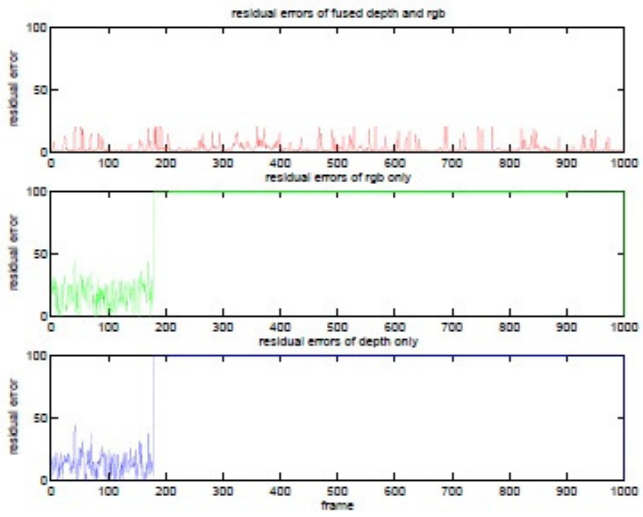


Fig. 5. Comparison graphs of residual error for a sequence with extreme rotations, during which the head is turned to a complete profile view. The top graph shows our method, in which the tracking maintains a constant error rate despite several rotations throughout the sequence. The second graph shows the results of a color-only method, which has larger errors, and loses the tracking at the first extreme rotation (frame 180). The third graph shows the results of a depth-only tracker, which also loses the tracking at the first extreme rotation, and exhibits significantly noisier results than our method.

Lighting changes We ran our algorithm on sequences where a sudden extreme change in ambient lighting caused the entire scene to change colors. Using a model-based method with color only causes tracking to be lost as soon as the lighting changes (see Fig. 6). This is due to the fact that the model is created a priori and does not take into account future color changes. In addition, tracking features by color only works well if the color does not change throughout the scene, due to the optical flow assumption. In our method, we generate a new model for each frame, and thus it adjusts quickly to any change in lighting. Once a significant change in lighting occurs, we get a large residual error. We then recover the tracking using the method described in Section 4E, and create a new model based on the current lighting in the scene. The change in lighting is reflected in the new model's texture, and thus we are able to continue the estimation process as before. In addition, the fusion of time-of-flight depth adds additional information, increasing the overall robustness of the motion tracking phase. The final residual error using the color-only tracker was approximately 2-3 times higher than our method. We did not test the depth-only tracker in these experiments, as lighting changes do not affect the output of the time-of-flight sensor.

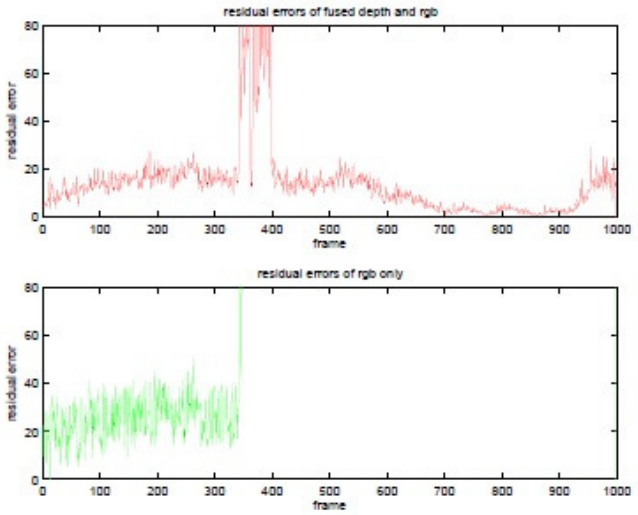
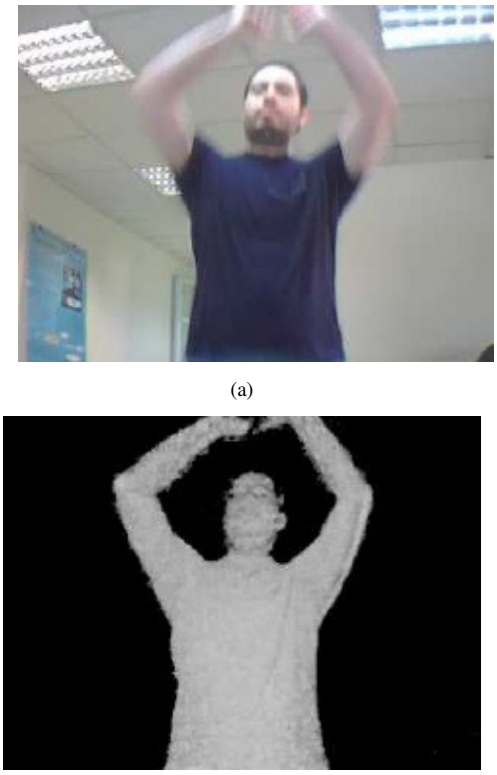


Fig. 6. Comparison graphs of residual error for a sequence with sudden change in lighting. The top graph shows our method, whereas the bottom graph shows the results of model-based method using only RGB color. The lighting change happens around frame 350. Note that our algorithm is able to quickly recover, whereas the color-based method loses tracking. In addition, the above graphs clearly show that the error is much higher throughout the sequence in the case of color-only tracking.

Fast motion Fast motion sequences caused problems when using the color only tracker, as the data appeared to be overly smoothed due to motion blur at the camera's frequency of 30 Hz. This caused the tracking to get lost when the head made a sudden quick rotation. This is due to the fact that color gradients are degraded as the image is smoother (see Fig. 7). When fusing the TOF depth data, the tracking worked fine as the depth sensor does not exhibit the same blurring effects as seen in the color camera (see Fig. 8).



(a)

(b)

Fig. 7. A frame from a fast motion sequence recorded with the ZCam. The color data (a) appears blurred as a result of the 30 Hz frequency. The depth data (b), however, does not exhibit the same effects, and therefore can be fused with color to significantly improve tracking results in fast motion.

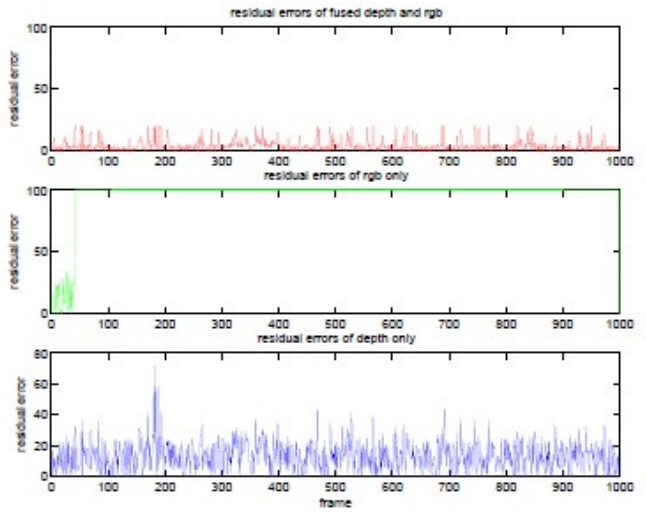


Fig. 8. Comparison graphs of residual error for a sequence with fast rotations. The top graph shows our method, in which the tracking maintains a constant error rate despite several fast rotations throughout the sequence. The second graph shows the results of a color-only method, which has larger errors, and loses the tracking at the beginning (frame 30). The third graph shows the results of a depth-only tracker, which does not lose tracking but does have significant noise compared to our method.

VI. Conclusion

We presented a robust system for real-time pose estimation by using data from a single TOF sensor. The system provides a better solution than previous methods by exploiting the depth information and fusing it with matching color data. This allows for improvement of current model-based methods in that the model can be updated often, and that the tracking actually occurs in 3D rather than 2D. Furthermore, the system does not rely on specific feature points, such that tracking works robustly from all angles of the head. By generating a new model per frame, we avoid the drifting problem which is common in all tracking applications, and also avoid problems when lighting changes in the middle of a sequence. Our method runs comfortably at 30fps on a single-core 2.4GHz PC with an on-board GPU.

This method could be further improved by adding a weighing scheme, such as to decide how much color and TOF depth to use respectively per frame based on some heuristic. It would also be of value to experiment with the fusion of TOF depth and IR data. Unlike the ZCam, color is not available in other TOF cameras whereas IR data is inherent in their functionality.

Acknowledgment

This work was supported by the Israeli Ministry of Science Grant 3-5795.

References

[1] 3DV Systems, *ZCam Practical Guide,* edition 1.03, January 2008.

[2] V. Blanz and T. Vetter. "Face recognition based on fitting a 3d morphable model. *IEEE Trans, Pattern Anal. Mach. Intell.,* 25(9):1063-1074, 2003.

[3] A. Bleiweiss and M. Werman. "Fusing time-of-flight depth and color for real-time segmentation and tracking". In *Dyn3D '09: Proceedings of DAGM 2009 Workshop on Dynamic 3D Imaging,* pages 58-69, 2009.

[4] C. Bregler and J. Malik, "Tracking People with Twists and Exponential Maps", IEEE Conf. Computer Vision and Pattern Recognition, 1998, pp. 8-15.

[5] C.M.J. Brolly and X.L.C. Stratelos. "Model-based head pose estimation for air-traffic controllers". In *ICIP 2003: Proceedings of the International Conference on Image Processing,* pages 113-116, 2003.

[6] L.M. Brown and Y.L. Tran. "Comparative study of coarse head pose estimation. In *MOTION '02: Proceedings of the Workshop on Motion and Video Computing,* page 125, 2002.

[7] F. de la Torre, Y. Yacoob, L. Davis. "A probabilistic framework for rigid and non-rigid appearance based tracking recognition. *IEEE International Conference on Automatic Face and Gesture Recognition*, 0:491-499,2000.

[8] D. Decarlo and D. Metaxas. "Optical flow constraints on deformable models with applications to face tracking. *Int. J. Comput. Vision,* 38(2):99-127, 2000.

[9] F. Dornaika and J. Ahlberg. "Model-based head and facial motion tracking", In *ECCV Workshop on HCI,* volume 3058 of *Lecture Notes in Computer Science,* pages 221-232, 2004.

[10] P. Eisert, "Low Bit-Rate Video Coding using 3D Models". Phd thesis, Friedrich-Alexander-University of Erlangen-Nuremberg, 2000.

[11] S.B. Gokturk and C. Tomasi. "3d head tracking based on recognition and interpolation using a time-of-flight depth sensor". In *CVPR*, pages 211-217, 2004.

[12] S.B. Gokturk, J.Y Bouguet, C. Tomasi, B. Girod, "Model-Based Face tracking for View-Independent Facial Expression Recognition", *Face and Gesture Recognition*, 2002.

[13] U. Hahne and M. Alexa. "Depth Imaging by combining time-of-flight and on-demand stereo". In *Dyn3d '09: Proceedings of the DAGM 2009 Workshop on Dynamic 3D Imaging*, pages 70-83, 2009.

[14] T. Horprasert, Y. Yacoob, L.S. Davis. "Computing 3d head orientation from a monocular image sequence". In *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition,* page 242, 1996.

[15] A.K. Jain and S.Z. Li. *Handbook of Face Recognition*. 2005.

[16] O. Kahler, E. Rodner, J. Denzler. "On fusion of range and intensity information using graph-cut for planar patch segmentation. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4):365-373, 2008.

[17] I. Kemelmacher and R. Basri. "Molding face shapes by example". In *ECCV 2006: Proceedings of the European Conference in Computer Vision*, pages 277-288, 2006.

[18] A. Kolb, E. Barth, R. Koch, and R. Larsen. "Time-of-Flight Sensors in Computer Graphics". In *Proc. Eurographics (State-of-the-Art Report), 2009.*

[19] V. Lepetit, J. Pilet, and P. Fua. "Point matching as classification problem for fast and robust object pose estimation". In *CVPR,* pages 244-250, 2004.

[20] W.K. Liao and G.G. Medioni. "3d face tracking and expression inference from a 2d sequence using manifold learning". In *CVPR,* pages 1-8, 2008.

[21] X. Liu, Y. Zhu, and K. Fujimura. "Real-time pose classification for driver monitoring". In *The IEEE 5th International Conference on Intelligent Transportation Systems,* pages 174-178, 2002.

[22] S. Malassiotis and M.G. Strintzis. "Robust real-time 3D head pose estimation from range data", *Pattern Recognition*, 38(8):1153-1165, 2005.

[23] S. Meers and K. Ward. "Head pose tracking with a time-of-flight camera". In *Proceedings of the Australian Conference on Robotics and Automation,* pages 113-116, 2008.

[24] E. Murphy-Chutorian, A. Doshi, and M.M. Trivedi. "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *2007 IEEE Intelligent Transportation Systems Conference,* pages 709-714, 2007.

[25] S. Ohayon and E. Rivlin. "Robust 3d head tracking using camera pose estimation. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition,* pages 1063-1066, 2006.

[26] R. Ruddarraju,A. Haro, and I.A. Essa. "Fast multiple camera head pose tracking". In *Proceedings of Vision Interface 2003*, 2003.

[27] I. Schiller and R. Koch. "Data structures for capturing dynamic scenes with a time-of-flight camera. ". In *Dyn3D '09: Proceedings of DAGM 2009 Workshop on Dynamic 3D Imaging,* pages 42-57, 2009.

[28] H. Spies and J.L. Barron. "Evaluating the range flow motion constraint". *ICPR,* Vol. 3, pages 517-520, 2002.

[29] T. Vatahska, M. Bennewitz, and S. Behnke. "Feature-based head pose estimation from images. In *Proceedings of the IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids),* 2007.

[30] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. "Three-dimensional scene flow". *PAMI*, 27(3):4, 2005.

[31] P. Viola and M. Jones. "Robust real-time face detection. In *ICCV 2001: Proceedings of the 8th IEEE International Conference on Computer Vision,* volume 2, page 747, 2001.

[32] C. Walder, M. Breidt, H. Bulthoff, B. Scholkopf, and C. Curio. "Markerless 3d face tracking". In *Proceedings of the 31st DAGM Symposium on Pattern Recognition,* pages 41-50, 2009.

[33] S.B. Wang, D. Demirdjian, T. Darrell, and H. Kjellstrom. "Multimodal communication error detection for driver-car interaction". In *ICINCO-RA,* pages 365-371, 2007.

[34] Y. Wu and K. Toyama. "Wide-range, person-and-illumination-insensitive head orientation estimation". In *FG '00: Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition,* page 183, 2000.

[35] Z. Zhang, Z. Liu, C. Jacobs, and M. Cohen. "Rapid modeling of animated faces from video". Technical report, Journal of Visualization and Computer Animation, 2000.

[36] J. Zhu, L. Wang, R. Yang, and J. Davis. "Fusion of time-of-flight depth and stereo for high accuracy depth maps". In *CVPR,* 2008.

[37] Y. Zhu and K. Fujimura. "Head pose estimation for driver monitoring". In *Intelligent Vehicles Symposium,* 2004, pages 501-506, 2004.