The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Real-Time Face and Object Tracking.

Benjamin R. Fransen, Evan V. Herbst, Anthony Harrison, William Adams and J. Gregory Trafton

Abstract— **Tracking people and objects is an enabling technology for many robotic applications. From human-robot-interaction to SLAM, robots must know what a scene contains and how it has changed, and is changing, before they can interact with their environment. In this paper, we focus on the tracking necessary to record the 3D position and pose of objects as they change in real time. We develop a tracking system that is capable of recovering object locations and angles at speeds in excess of 60 frames per second, making it possible to track people and objects undergoing rapid motion and acceleration. Results are demonstrated experimentally using real objects and people and compared against ground truth data.**

I. Introduction

Object and person tracking is a common need for robotic applications. While a considerable amount of research has been performed in the field of tracking, [26], a versatile tracking algorithm capable of modeling and tracking objects in real time remains a highly sought after tool. The system presented here satisfies this need by recovering object models from scene sensor data and then tracking said objects in real time.

Tracking has two stages. First, an object must be learned by the system. To generate object models, a time-of-flight range camera is integrated with a calibrated color image at the pixel level. This produces a dense range and color image for every pixel. Elements of the range and color image can then be tracked in segments or as a whole. It is also important to note that no prior data is necessary for object modeling, all data is generated by sensors on a mobile robot.

Second, an object must be detected and then tracked. We utilize two methods for object detection, a haar based detection system and a detection system that searches for objects using the tracking system itself. To track the scene, a 3D optical flow algorithm is derived that models scene translation and rotation, recovering a full SE(3) transformation of an objects motion.

During tracking, only luminance images are used. By removing the dependency on range images while tracking, object models can be used by robots outfitted with only luminance based cameras. This facilitates tracking at larger distances where range data is less accurate than camera data and facilitates the re-use of data by robots that lack range sensors.

Work presented here has a wide impact including human-robot-interaction, robotic object manipulation, simultaneous localization and mapping (SLAM), and even privacy preservation [23]. Several contributions to the state of the art are made by this paper. An extremely efficient optical flow algorithm is presented for use in real time applications capable of tracking faces and general scene elements. Multi-core processor algorithm design issues and performance characteristics are evaluated. Additionally, a novel hardware integration is demonstrated for capturing combined shape and appearance data for scene elements.

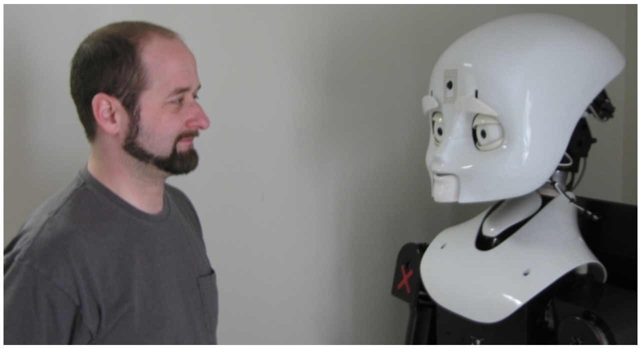


Fig. 1. MDS, ”Isaac”, poses with a tracking subject.

To help identify our system when comparing different techniques throughout this paper we decided to name our system ”FLOAT” which stands for Fast Linear Object Appearance Tracking. The name FLOAT is informative in regard to the proposed algorithm and we will use it throughout this paper to address the techniques proposed here.

II. Background

Visual tracking is commonly used to enable human robot interaction and robot navigation. Human robot interaction utilizes whole body person tracking [15], arm and face tracking [7], and face orientations [22], amongst others. Vision based navigation, commonly employs imaging systems for simultaneous localization and mapping (SLAM) [2], [25], object tracking [19], and scene recognition [3].

While the task space in vision-based robotics is diverse, a few fundamental vision techniques overlap many, if not all, robotic tasks. Of those techniques, two major thrusts are notable—optical flow and template matching. Optical flow tracking began with planar scene elements, eg. Kanade-Lucas-Tomasi (KLT) tracker [27], and has since been extended to 3D objects by introducing shape from stereo and the recent work in tracking via parametric shape models [1]. While the applications of templates is often times different than optical flow, template based methods such as SIFT [8],

This work was supported by the Office of Naval Research under work order number **N0001509WX20173**. The views and conclusions contained in this document should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U. S. Navy.

[18], [24] have shown great versatility in applications ranging from 3D tracking for navigation to object recognition.

Work presented here extends a body of work in optical flow tracking [4], [11], [14] by developing an efficient 3D optical flow algorithm that facilitates tracking large regions. We demonstrate that object tracking and appearance based navigation does not need to be restricted to edges [19], planar geometric restrictions [21], [25], corners, or other fiducials. Additionally, we demonstrate how a time-of-flight range camera can be integrated with a color camera at the pixel level. In doing so, accurate position estimates can be recovered for each pixel without the dense texture requirements presented by stereo cameras. This facilitates tracking a much wider range of objects than stereo cameras are capable.

No user interaction is necessary for generating models. Many tracking systems require user input for initialization. Active appearance models [10], for example, commonly require user specified correspondence between a models and a user's face. In contrast, all initialization is performed automatically by this system. In the field of 3D model generation, it is important to note that neither TOF imaging or stereo cameras are necessary. Range imaging can be performed by using shadows from an office lamp [5]. The minimal hardware requirement for producing a tracking system of this type is, therefore, a low-cost camera.

We will start by introducing the system that we use to generate training data. For this task, we register a time-of-flight (TOF) camera with a color camera to produce a dense image of both luminance and 3D positions for every pixel. We refer to this combined 3D and luminance data as the object model. We will then formalize an efficient optical flow algorithm that tracks an object model in a video stream, without the aid of depth information, in real time.

III. Model Generation

To generate 3D models of an object we first derive the methods necessary to register TOF range data with luminance data. Models, as referred to in this paper, are then created using regions of a registered set of data.

A. Sensor Integration

Integrated range and color images are generated through pixel level registration between a TOF range camera and a calibrated color camera. The range camera used for this system is a Swiss Ranger time-of-flight sensor. The Swiss Ranger produces a 176x144 image with a 47.5°x39.6°field of view with a maximum range of 7.5m. The transformation from 3D coordinates in the TOF camera's coordinate system to the color camera's coordinate system is recovered through a least square point pair registration method described by Horn et al [13]. To produce point pair correspondence, a standard checkerboard calibration pattern is presented to both cameras. The range camera returns both a luminance image, measuring the reflectivity at each point in the range image, and a 3D depth estimation. Using the range camera's luminance image, the calibration grid can be detected and point correspondences can be generated between pixels in the range image and pixels in the color image. The camera systems mounted on the MDS robot [6] are shown in Figure 2. For simplicity, we chose to rotate the eye cameras to make the optical axis parallel between the time-of-flight camera and the color camera. While this is not absolutely necessary, it can greatly simplify the camera registration process. A larger image showing the robot with a tracking subject is shown in Figure 1.

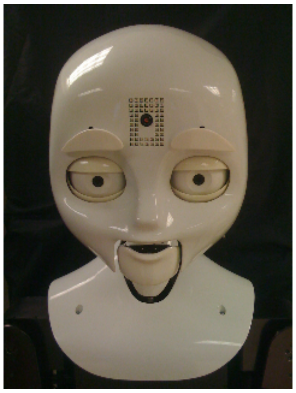


Fig. 2. Close up of sensor systems. Each eye contains a Point Grey FireFly camera and the Swiss Ranger time-of-flight sensor is located in the forehead.

B. Merging Range and Luminance Images

Once the coordinate between cameras is recovered, a projection matrix can be used to project TOF range data into the luminance image as follows.

A luminance image I is a one-dimensional function defined at pixel locations $\vec{u}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$. Similarly, a range image is a one-dimensional function defined at each pixel $\vec{x}_j = \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix}$. Using a pinhole camera model, correspondence between the range image and luminence image can be solved.

A pinhole camera projective model P with focal lengths (F_u, F_v) and optical center (C_u, C_v) projects points from 3D to image coordinates as

$$u_i = \frac{F_u\, x_j}{z_j} + C_u,\ v_i = \frac{F_v\, y_j}{z_j} + C_v. \quad (1)$$

Renumbering corresponding points in the the range and luminence image, we now have a correspondence between points imaged by the TOF sensor and pixels in an image

$$(u_i, v_i) = P(x_i, y_i, z_i).$$

P can be recovered through stereo camera calibration [16] or generated for any web camera through freely available calibration toolboxes [28].

IV. Model Tracking

Each image I_i in a sequence shows the same scene from some viewpoint. Image formation can be broken down into two parts: a motion model for the scene and a projection from 3D to 2D. We use the SE(3) transformation (a rotation, scaling and translation) as our motion model, and a pinhole camera model, reviewed previously. SE(3) is a group, having inverses and being closed under composition.

A. Motion Model

Let $\vec{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}$ be the coordinates of a point on the object to be tracked at time t with respect to the camera's coordinate system. The object's 3D motion is given by motion parameters $\vec{\mu}$ and the function $\vec{f}(\vec{x}, \vec{\mu})$: the position at time t of point $\vec{x}$ is given from its position at time 0 by $\vec{x}(t) = \vec{f}(\vec{x}(0), \vec{\mu}(t))$.

$$\begin{aligned} \vec{x}(t) &= \vec{f}(\vec{x}(0), \vec{\mu}(t)) & (2) \\ &= R_t^{3\times 3} \begin{bmatrix} x(0) \\ y(0) \\ z(0) \end{bmatrix} + \vec{T}_t^{3\times 1}. & (3) \end{aligned}$$

For $(R_t, \vec{T}_t)$ to be an element of SE(3) it is necessary to enforce that R_t gives a rotation and a homogeneous scaling. This allows us to use a nonobvious $\vec{\mu}$, which we discuss later.

The coordinates of $\vec{x}$ at times t and k are then related by

$$\vec{x}(t) = R_t \left(R_k^{-1} \vec{x}(k) - \vec{T}_k \right) + \vec{T}_t. \quad (4)$$

B. Tracking

The problem of image registration consists of finding, for each image I_i in a sequence, a geometric transformation between the scene as seen in the first image I_0 and as seen in I_i, parameterized by a vector $\vec{\mu}(i)$. At time 0 an object template consisting of a set $\mathcal{R} = \{\vec{x}_i(0)\}$ of 3D points is acquired with the help of a depth camera as mentioned previously. We center the template at the origin to help minimize numerical error when applying transformations later by the motion model,

$$\begin{aligned} \vec{f}(\vec{x}(0), \vec{\mu}(t)) &= R \begin{bmatrix} x(0) \\ y(0) \\ z(0) \end{bmatrix} + \vec{T} & (5) \\ &= \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x(0) \\ y(0) \\ z(0) \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} & (6) \end{aligned}$$

One method employed for image registration algorithms is gradient descent. One group of gradient-descent-based algorithms, called compositional algorithms, assume that warps form a group with the composition operation. Baker [4] introduced the term inverse compositional image alignment (ICIA), an especially efficient compositional algorithm, and showed its mathematical equivalence to previous popular gradient-descent-based approaches. Recent work by Buenaposada and Muoz [14] gives a very readable introduction to ICIA. While most uses of gradient descent algorithms have been planar, (*e.g.* [11]), we track in 3D. The most recent 3D face tracking system from the developers of Watson [20] also does 3D tracking with a variant of optical flow based tracking, but with an ellipsoidal rather than a data-defined model and at a reduced frame rate of only 12 Hz. In addition to run time performance, we avoid the inaccuracies of parametric models and initialization difficulties encountered when parametric models are registered with objects in a video stream.

One distinction for work presented here from previous work is the motion model employed. Previous work utilizes an incremental motion model based on the small angle approximation, eg. [12]. Because we intend to track large rotation changes, we provide a derivation for recovering a full rotation matrix without restricting each incremental change by the small angle approximation. This is done for tracking of rapidly moving objects and to facilitate future use of this algorithm in object recognition experiments where detecting objects at large angle changes is beneficial.

One straightforward choice for the parameter vector $\vec{\mu}$ is then $\begin{bmatrix} T_x & T_y & T_z & a & b & c & d & e & f & g & h & i \end{bmatrix}$.

We apply our motion model in an ICIA framework as follows. Estimate $\vec{\mu}$ at each timestep by least-squares:

$$O(\vec{\mu}) = \sum_{\vec{x} \in \mathcal{R}} \left(\vec{I}(\vec{f}(\vec{x}, \vec{\mu}), t_0) - \vec{I}(\vec{x}, t) \right)^2 .$$

At each timestep t, find the change in $\vec{\mu}$ that takes I_t to $I_{t+\tau}$:

$$O(\delta\vec{\mu}) = \left| \vec{I}_t(\vec{f}(\vec{x}(0), \vec{\mu}^{-1}(t))) - \vec{I}_0(\vec{f}(x(0), \delta\vec{\mu})) \right|^2 ,$$

Here we use the notation $\vec{I}_t(g(\vec{x}))$ to stand for a column vector concatenating

$$I_t(g(\vec{x}_i)) \forall i \in \mathcal{R}.$$

Also define $\vec{\mu}^{-1}$ to be the parameter vector such that

$$\vec{x}_2 = \vec{f}(\vec{x}_1, \vec{\mu}) \iff \vec{x}_1 = \vec{f}(\vec{x}_2, \vec{\mu}^{-1}).$$

Linearize the objective:

$$\begin{aligned} O(\delta\vec{\mu}) &\approx \left| \vec{I}_t(\vec{f}(\vec{x}(0), \vec{\mu}^{-1}(t))) - \vec{I}_0(\vec{x}(0)) - \frac{\partial \vec{I}_0}{\partial \vec{\mu}}(\vec{x}(0))\delta\vec{\mu} \right|^2 \\ &= \left| \frac{\partial \vec{I}_0}{\partial \vec{\mu}}(\vec{x}(0))\delta\vec{\mu} + \vec{I}_0(\vec{x}(0)) - \vec{I}_t(\vec{f}(\vec{x}(0), \vec{\mu}^{-1}(t))) \right|^2 \end{aligned}$$

Define the *error image* at time t:

$$\mathcal{E}(\vec{x}, \vec{\mu}) \equiv I_0(\vec{x}) - I_t(\vec{f}(\vec{x}, \vec{\mu}^{-1}(t)))$$

Then the least-squares solution for $\delta\vec{\mu}$ is

$$\delta\vec{\mu}^* = -(M^T M)^{-1} M^T \vec{\mathcal{E}}(\vec{\mu}),$$

where $M \equiv \frac{\partial \vec{I}_0}{\partial \vec{\mu}}$.

For calculation purposes, we can break down each row M_i of M as

$$\begin{aligned} M_i &\equiv \frac{\partial I_0}{\partial \vec{\mu}}(\vec{f}(\vec{x}_i(0), \vec{\mu}(0))) \\ &= \frac{\partial I_0}{\partial \vec{f}}(\vec{f}(\vec{x}_i(0), \vec{\mu}(0)))\frac{\partial \vec{f}}{\partial \vec{\mu}}(\vec{x}_i(0), \vec{\mu}(0)) \\ &= \frac{\partial I_0}{\partial \vec{f}}(\vec{x}_i(0))\frac{\partial \vec{f}}{\partial \vec{\mu}}(\vec{x}_i(0), \vec{\mu}(0)) \\ &= \frac{\partial I_0}{\partial \vec{u}}(\vec{u}(\vec{x}_i(0)))\frac{\partial \vec{u}}{\partial \vec{x}}(\vec{x}_i(0))\frac{\partial \vec{f}}{\partial \vec{\mu}}(\vec{x}_i(0), \vec{\mu}(0)). \quad (7) \end{aligned}$$

The major advantage of the inverse compositional algorithm over the basic compositional algorithm, which is why we use the ICIA framework, is that M has no dependence on $\vec{\mu}(t)$, meaning M can be calculated entirely during initialization.

$\frac{\partial \vec{u}}{\partial \vec{x}}$ is the Jacobian of the pinhole transform:

$$\frac{\partial \vec{u}}{\partial \vec{x}}(\vec{x}) = \begin{bmatrix} \frac{F_u}{z} & 0 & \frac{-xF_u}{z^2} \\ 0 & \frac{F_v}{z} & \frac{-yF_v}{z^2} \end{bmatrix},$$

We model all the elements of $\vec{\mu}$ as independent, resulting in a straight forward Jacobian:

$$\frac{\partial \vec{f}}{\partial \vec{\mu}}(\vec{x}_i(0), \vec{\mu}(0)) = \begin{bmatrix} 1 & & & \vec{x}_i(0)^T & & \\ & 1 & & & \vec{x}_i(0)^T & \\ & & 1 & & & \vec{x}_i(0)^T \end{bmatrix}.$$

This formulation doesn't actually restrict $\vec{\mu}$ to represent a rotation, translation and scaling; it can be any 3D affine transformation. Thus we could express camera movement and measurement errors as a skew in the rotation matrix R. However, in this work we instead use properties of rotation matrices and of the tracked object to constrain $\vec{\mu}$. Firstly, the rows of a rotation matrix are orthogonal:

$$\begin{bmatrix} g & h & i \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix} \times \begin{bmatrix} d & e & f \end{bmatrix}.$$

Secondly, each row of a rotation matrix has unit 2-norm; in particular

$$a^2 + b^2 + c^2 = 1 = d^2 + e^2 + f^2.$$

Thirdly, since a rigid object can't change size, for our purposes "scaling" must be interpreted as a change in z coordinate: after optimizing over $a \ldots i$ but before normalizing them, we can set

$$T_z(t + \tau) = |R| \; T_z(t).$$

Taking all these facts into account, we can remove g, h, i and T_z from the vector we put through least-squares, reducing the dimensionality of the optimization.

Now

$$\vec{\mu} = \begin{bmatrix} T_x & T_y & a & b & c & d & e & f \end{bmatrix},$$

$$\frac{\partial \vec{u}}{\partial \vec{x}}(\vec{x}) = \begin{bmatrix} \frac{F_u}{z} & 0 \\ 0 & \frac{F_v}{z} \end{bmatrix}$$

and

$$\frac{\partial \vec{f}}{\partial \vec{\mu}}(\vec{x}_i(0), \vec{\mu}(0)) = \begin{bmatrix} 1 & & \vec{x}_i(0)^T & \\ & 1 & & \vec{x}_i(0)^T \end{bmatrix}.$$

Note that despite the fact that we're now explicitly writing some elements of $\vec{\mu}$ as functions of others, for runtime efficiency we retain the assumption that all variables are independent. Since by assumption f_z is independent of (the new) $\vec{\mu}$, the third row $\frac{\partial f_z}{\partial \vec{\mu}}$ of $\frac{\partial \vec{f}}{\partial \vec{\mu}}(\vec{x}_i(0), \vec{\mu}(0))$ becomes zero, so we drop it and $\vec{f}$ becomes a two-dimensional function $\begin{bmatrix} x_i(t) \\ y_i(t) \end{bmatrix} = \vec{f}(\vec{x}_i(0), \vec{\mu}(t))$.

We follow [14] in performing a small fixed number of least-squares iterations per frame, using the most recent frame I_k as an approximation of I_t for

$$t \in \{k, k + \tau, k + 2\tau, \ldots\}.$$

V. Performance

The algorithm presented here has a computational complexity of O(n) where n is the number of pixels. The number of iterations we use for convergence depends on the frame rate of the camera being used. For fast cameras with 60fps image streams we use 15 iterations to maintain full frame rate tracking. For slower cameras, more iterations can be used while still maintaining the maximum frame rate supported by the camera. With a fixed number of iterations, processing requirements are deterministic, which is important for two reasons. Deterministic processing allows us to ensure that data will be available at regular intervals and it allows us to run multiple applications on one processor without worrying that a single application may sporadically overburden the processor reducing resources intended for other applications.

To enable high throughput rates for the system we utilized a multi-threaded architecture. Due to modeling variables independently and the structure of linear systems, the algorithm can be broken down into equal sets of data than can be processed symmetrically. By separating the problem into parts and running each component in a different thread we have been able to capture the full performance potential of today's multi-core architectures. Table I shows the frame rate vs. thread count for a multi-threaded linear system solution on a dual core machine. It is interesting to note the initial speeds gains when moving from two to three threads, followed by a slow decline in performance when the thread count is beyond 4. This shows that the overhead of additional threads outweighs load balancing gains between the cores as the thread count becomes higher. The MDS carries a quad core machine resulting in further speedup with additional threads than reported here and we commonly run that system with 8 threads.

VI. Experiments

To verify the proposed tracking system, experiments were conducted using faces. For validation data, an inertia cube was used to record rotations for ground truth data comparisons. As output, the proposed system produces rotations in degrees and translations in centimeters.

TABLE I

Run-Time Performance Vs. Number of Threads

Threads	Frames/Second
1	32
2	32
3	64
4	65
5	64
6	64
7	63

To visually demonstrate the tracking algorithm, the scene was augmented in two ways. First a coordinate system was drawn in the video. For faces, the coordinate system was placed on the tip of the nose. The coordinate system is transformed using the recovered SE(3) transformation and projected into each image. Each base of the coordinate system is 10 centimeters long for face tracking and 30 centimeters for scene tracking. The second visualization is a projection of the original template back into a darkened black and white version of each image.

A. Face Tracking with Ground Truth Data

To produce ground truth data for comparison purposes, an inertia cube was affixed atop the head mount from a Plexiglas face shield with the actual plastic shield removed. As safety equipment, face shields have sturdy customizable head gear that is well suited for placing sensors. The coordinate systems of the visual tracker and inertia cube were aligned physically by rotating the head gear until alignment. Once aligned, the head gear proved to be reliable and did not slip during any of our tests.

Face tracking is automatically initialized through a boosted cascade algorithm [17] implemented in [9]. Face depth is estimated from the face size. Face location is estimated from the center of the detected face region. Orientation can be recovered if the detected face is within approximately fifteen degrees from a frontal face pose.

Results comparing ground truth data to user tracks are shown in Figure 3. For further comparison, we also tracked the person using the Watson system [20]. The original training picture, template and tracking results are shown in Figure 4. These results demonstrate the system's capacity to accurately track. To mitigate the effects of occlusion, three templates were generated from the original training image. The templates regions cover different sections of the face. Doing so, large yaw angles were possible without occlusion. Root mean squared (RMS) error, calculated in degrees, for pitch, yaw and roll for both Watson and FLOAT are shown in Table: II. Other than yaw, resulting error is within the 3 degrees RMS error of the inertia cube with FLOAT producing 30% less error than the Watson system. One possible source of error for the Watson system is a misalignment between the inertia cube's coordinate system and Watson's coordinate system determined from face detection. The Watson system reinitializes the tracker continuously to reduce drift. If during one of the detection phases the face is not frontal, the ground truth data and recovered track will not match. FLOAT does not suffer from drift and only needs to detect objects once at the start of a track sequence. The low resulting error shows the system's capacity for accurately tracking faces.

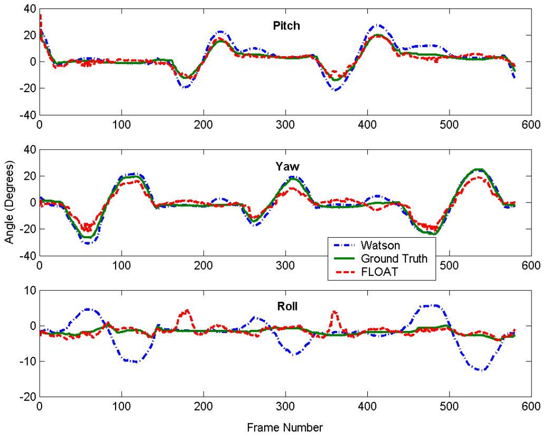


Fig. 3. Pitch, yaw and roll results for visual tracking vs. inertia cube generated ground truth data.

TABLE II

Error rates for FLOAT and WATSON

	Pitch	Yaw	Roll
FLOAT	1.72°	3.36°	0.85°
WATSON	2.86°	1.96 °	2.84°

VII. Conclusions and Future Work

A 3D tracking system based on scene data data has been derived and demonstrated as an accurate and efficient method for tracking faces. It is important to note that this system is not limited to face tracking and can be used to track a wide range of scene elements. Run time performance has been demonstrated to be in excess of 60 frames/second facilitating accurate tracking of rapidly moving objects. Additional research remains in further developing robust statistics for 3D flow based tracking and in expanding this system for use in object recognition.

References

[1] K. H. An and M. J. Chung. 3d head tracking and pose-robust 2d texture map-based face recognition using a simple ellipsoid model. In *IROS*, pages 307–312. IEEE, 2008.

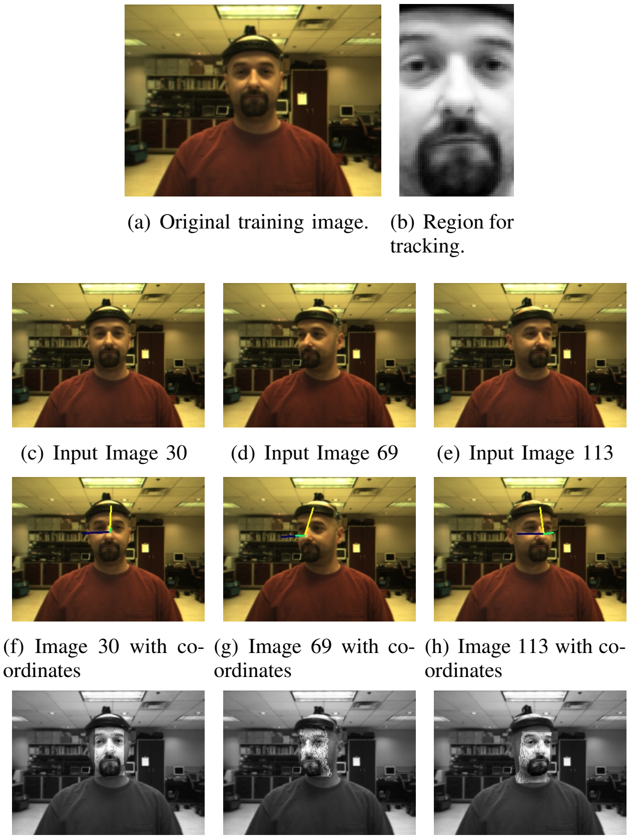
[2] H. Andreasson, T. Duckett, and A. J. Lilienthal. A minimalistic approach to appearance-based visual slam. *IEEE Transactions on Robotics*, 24(5):991–1001, 2008.

[3] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Real-time visual loop-closure detection. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2008.

[4] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR01)*, 2001.

[5] J.-Y. Bouguet and P. Perona. 3d photography on your desk. *Computer Vision, 1998. Sixth International Conference on*, pages 43–50, Jan 1998.

[6] C. Breazeal, M. Siegel, M. Berlin, J. Gray, R. Grupen, P. Deegan, J. Weber, K. Narendran, and J. McBean. Mobile, dexterous, social robots for mobile manipulation and human-robot interaction. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2008.



(a) Original training image.

(b) Region for tracking.

(c) Input Image 30

(d) Input Image 69

(e) Input Image 113

(f) Image 30 with coordinates

(g) Image 69 with coordinates

(h) Image 113 with coordinates

(i) Image 30 augmented.

(j) Image 69 augmented.

(k) Image 113 augmented.

Fig. 4. Images from a face tracking sequence with ground truth data. Images a and b show the original image and region for tracking. Images c through e show input images. F through h show a coordinate system projected onto the face. The scene is augmented by projecting the template, b, into the scene for images i through k.

[7] B. Fransen, V. Morariu, E. Martinson, S. Blisard, M. Marge, S. Thomas, A. Schultz, and D. Perzanowski. Using vision, acoustics, and natural language for disambiguation. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 73–80, New York, NY, USA, 2007. ACM.

[8] S. Frintrop, P. Jensfelt, and H. I. Christensen. Attentional landmark selection for visual slam. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2582–2587, 2006.

[9] e. a. G. Bradski. Intel Open Source Computer Vision Library: http://www.intel.com//research//mrl//research//opencv//.

[10] R. Gross, I. Matthews, and S. Baker. Abstract constructing and fitting active appearance models with occlusion, June 2004.

[11] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1025–1039, 1998.

[12] M. Harville, A. Rahimi, T. Darrell, G. Gordon, and J. Woodfill. 3d pose tracking with linear depth and brightness constraints. *iccv*, 01:206, 1999.

[13] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629, 1987.

[14] L. B. Jose M Buenaposada, Enrique Munoz. Efficient illumination independent appearance-based face tracking. *Image and Vision Computing*, In Press, 2008.

[15] M. Kobilarov, G. S. Sukhatme, J. Hyams, and P. H. Batavia. People tracking and following with mobile robot using omnidirectional camera and a laser. In *ICRA*, pages 557–562. IEEE, 2006.

[16] K. Konolige. The sri small vision system. http://www.ai.sri.com/konolige/svs/.

[17] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. volume 1, pages I–900–I–903 vol.1, 2002.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[19] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. Gpu-accelerated real-time 3d tracking for humanoid locomotion and stair climbing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pages 463–469, October 2007. This work was supported by an NVIDIA Fellowship.

[20] L. Morency, J. Whitehill, and J. Movellan. Generalized adaptive view-based appearance model: Integrated framework for monocular head pose estimation. *Face and Gesture Recognition*, 2008.

[21] M. Okutomi, K. Nakano, J. Maruyama, and T. Hara. Robust estimation of planar regions for visual navigation using sequential stereo images. In *ICRA*, pages 3321–3327. IEEE, 2002.

[22] A. Rahimi and T. Darrell. Adaptive view-based appearance model. In *In CVPR*, pages 803–810, 2003.

[23] J. Schiff, M. Meingast, D. Mulligan, S. Sastry, and K. Goldberg. Respectful cameras: detecting visual markers in real-time to address privacy concerns. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 971–978, 29 2007-Nov. 2 2007.

[24] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pages 2051–2058, 2001.

[25] G. F. Silveira, E. Malis, and P. Rives. An efficient direct method for improving visual slam. In *ICRA*, pages 4090–4095. IEEE, 2007.

[26] G. Taylor and L. Kleeman. Fusion of multimodal visual cues for modelbased object tracking. In *In Australasian Conference on Robotics and Automation (ACRA2003), Brisbane,Australia*, 2003.

[27] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[28] A. Whitehead and G. Roth. The projective vision toolkit, 2000.