# Illumination Insensitive Model-Based 3D Object Tracking and Texture Refinement

Hua Yang Greg Welch Marc Pollefeys
Computer Science Department
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599

## Abstract

*A common approach to model-based tracking is to use a model of the object to predict what will be observed, and then to compare that with real observations. For methods that make use of the object's photometric properties (appearance) in their measurements, illumination inconsistencies between the modeled and actual scene can cause tracking problems. In this paper we address one case: model-based tracking of Lambertian objects under directional light sources. We present an iterative optimization method that uses a Kalman filter to simultaneously refine estimates of the object motion, the illumination, and the model texture. We model the illumination variance between the real and predicted observation using the intensity ratios of corresponding surface points, which we then use to make model-based image predictions consistent with the real lighting. To demonstrate the effectiveness of our method we present experimental results using both synthetic (controlled) and real image sequences.*

## 1. Introduction

Conventional model-based tracking approaches often extract and match *geometric* features, e.g., contours or edges. Because such features are generally invariant to illumination, the methods can be considered *illumination insensitive*. On the other hand, methods [5, 12, 11] have been proposed to exploit the *appearance* information. For example, tracking or registration between the real and predicted images can be performed by comparing photometric properties such as intensity, color, or texture. These methods have proven to be accurate and efficient in many object tracking and recognition applications. Unfortunately, because the appearance of an object can change dramatically under different lighting conditions, typical appearance-based methods are prone to difficulties when the model and the real object are illuminated differently, as is usually the case.

In this paper, we present an illumination insensitive model-based 3D object tracking system. The inputs of the system consist of a texture-mapped graphics model and image sequences of the target object. The system outputs are the object pose estimate over frames, a refined texture model and an map that models the illumination variance between the model space and the real scene. We use a Kalman filter to recursively refine the pose (position and orientation), a model for the illumination, and the texture component of the object model. During tracking, we use graphics hardware (with texture mapping) to render synthetic images of the graphics model at its estimated pose, from the view point of each of the cameras. We use the pixel intensities of the real and synthetic images as the measurements. We handle the illumination inconsistency problem by estimating an Illumination Normal Map (INM)—a mapping from the surface normal of a 3D point to the intensity ratio of its projections in the real and synthetic images. The INM defines an illumination transformation between the real and synthetic images. Using this transformation, we can generate illumination-corrected synthetic images and compare them with the real images, as if they were taken under the same illumination.

In the next section we discuss some related work. In Section 2 we describe the Illumination Normal Map and analyze its properties. In Section 3 we provide a brief introduction to the relevant parts of the Kalman filter, and in Section 4 we describe our estimation process in the context of the Kalman filter framework. Finally in Section 5 we shows some results from both synthetic and real image sequences, and in Section 6 we discuss some future plans.

### 1.1. Previous Work

Classical model-based tracking methods typically use geometric object models that describe the *shape* of the objects. For instance, both Gennery and Bray exploited a wire frame model to track polyhedral objects [7, 2]. Rehg and

Kanade tracked articulated hand motion using a generalized cylinder model of the hand [14]. Stenger *et al.* proposed a method to capture hand motion using an hand model that is built from truncated quadrics [16]. Rosenhahn *et al.* proposed a silhouette-based approach for pose estimation of free-form surface models [15]. These methods avoid the illumination inconsistency problem by defining geometric features that are illumination invariant. However, geometric features like contours do not contain all the information useful for object tracking. For example, the rotation of a sphere cannot be detected using its contour information alone. In addition, extracting and matching features can be expensive when the object is geometrically complex.

Alternatively, some approaches use *appearance* models of objects—predicting the appearance of the object in each specific pose. Photometric properties such as intensity, color or texture can then be compared between the predicted and real observations. Dellaert *et al.* proposed a Kalman filter based system that simultaneously estimates the pose and refines the texture of a planar patch [5]. Nickels and Hutchinson tracked articulated objects through matching feature templates [12]. In [20] a novel 3D object tracking system was developed where the model of the object is a pre-acquired light-field. Such appearance-based methods make more direct and complete use of available information from the image observations, and have achieved outstanding performance in applications like face tracking. However, because the real and predicted observations are registered using illumination dependent properties, these methods are prone to difficulties or failure in the presence of illumination inconsistencies between the modeled and real scenes.

Illumination insensitive methods have been developed to recognize and track objects under varying lighting conditions. To maintain color consistency through image sequences, Cubber *et al.* used a transformed color space that is invariant to highlights, viewing direction, surface orientation and illumination directions [4]. Hager *et al.* developed an efficient region tracking method that uses a three-dimensional linear subspace to model all possible surface appearance of the same object under varying lighting conditions [9]. Training images are taken under sample lighting conditions to serve as the illumination templates. In [3], the linear subspace illumination model is improved to generate subject-independent templates. Ramamoorthi and Hanrahan [13] and Brasi and Jacobs [1] formulated the relationship between radiance and irradiance as a spherical convolution whose kernel is determined by the surface BRDF, and proved that under the assumption of Lambertian surfaces and directional light sources the illumination function can be recovered using first nine coefficients of a spherical harmonic basis. Freedman and Turek developed a tracking method that computes illumination-invariant optical flow using graph cuts [6]. They make the assumption that intensity consistency across pixels can be propagated over frames. Recently, Zhang *et al.* proposed the use of the intensity ratio between corresponding pixels to model the illumination variance between an image pair [18]. Belief Propagation is used to simultaneously estimate per-pixel illumination ratio and disparity.

## 2. Illumination Ratio Estimation

Illumination variation between the real and synthetic scenes can be modeled as the ratio of the intensities of corresponding pixels in two images. Consider a 3D point $p$ on the surface of the target object. The projection of $p$ on the image plane is $(x,\, y)$. Under Lambertian surface and directional lighting assumption, in the absence of shadow, the pixel intensity of $I(x, y)$ is given by

$$I(x,y) = \sum_k c_k a(p) L_k \cdot N(p) \tag{1}$$

where $a(p)$ is the nonnegative absorption coefficient (albedo) of the surface at the point $p$, $L_k$ is the unit direction vector of light source $k$, $c_k$ is the intensity of the light source $k$, $N$ is the unit inward normal vector of the surface at $p$.

The intensity ratio $r$ of corresponding pixels at $(x,\, y)$ in the real and synthetic images can be computed as

$$r(x,y) = \frac{I(x,y)}{\hat{I}(x,y)} = \frac{\sum_k c_k a(p) L_k \cdot N(p)}{\sum_k \hat{c}_k a(p) \hat{L}_k \cdot N_(p)} \tag{2}$$

where $\hat{I}(x,y)$ is the pixel intensity in the synthetic image, $\hat{c}_k$ and $\hat{L}_k$ represent the different illumination conditions in the synthetic scene. Since the albedo $a(p)$ is constant, it can be moved outside the summation and canceled in the division. Thus Eqn. (2) can be rewritten as

$$r(x,y) = \frac{\sum_k c_k L_k \cdot N(p)}{\sum_k \hat{c}_k \hat{L}_k \cdot N(p)} \tag{3}$$

### 2.1. Illumination Normal Map

One can see from Eqn. (3) that the illumination ratio $r$ of a point $p$ is independent of the albedo $a$, and is a smooth function of the surface normal $N$. This key observation leads to the formulation of the Illumination-Normal-Map (INM), a lookup table that maps a normal to an illumination ratio. Each point in the INM corresponds to a unit normal vector $N$ in the 3D space. Its horizontal $(\theta \in [0, 2\pi))$ and vertical coordinate $(\phi \in [0, \pi))$ is defined by the projection of $N$ in certain fixed spherical coordinate system. Its value is the illumination ratio value $\rho$ for the normal $N$ (Fig. 1). An INM defines an illumination transformation between the real and the synthetic scene. A good property of INM is that

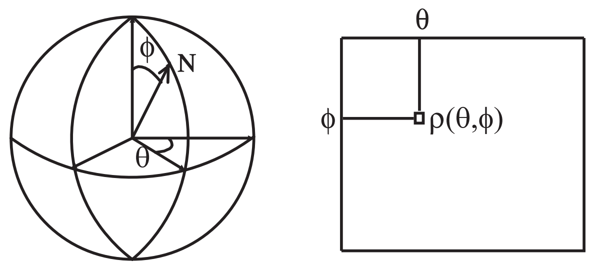

**Figure 1. Spherical coordinate and INM.**

$\rho$ is a smooth function of $(\theta, \phi)$. Thus we can compute $\rho$ for relatively sparse samples of the INM and estimate the rest using interpolation.

## 2.2. Compute Illumination Ratio

Consider a pixel $(x, y)$ in the synthetic image. Given the geometry model and pose estimation of the object, the corresponding surface point $p$ with normal $N(p)$ in the world space can be found through back-projection. Its coordinate $(\theta, \phi)$ in the INM can then be derived by projecting $N(p)$ to a specified spherical space. Thus $\rho(\theta, \phi)$, which equals to $r(x, y)$, can be computed by comparing the corresponding pixel intensities in two images Eqn. (2).

All surface points are illuminated by the same directional light source. For a convex object (simple self-occlusion), this means that points with the same normals share the same illumination ratio within one frame. Moreover, if the scene lighting is static, illumination ratio consistency should hold for points with same surface normals across frames. This means multiple observations of the same illumination ratio variable can be acquired with the presence of measurement noise. We formulate illumination estimation problem in the Kalman filter framework. The detailed approach is explained in the Section 5.

So far, we have made the assumption that the real image and the synthetic image are exactly registered. In other words, there is no error in object pose estimation and camera calibration. Unfortunately, in practice such errors always exist. Thus the per-pixel-based illumination computation is subject to fail, for the ratio maybe computed by comparing the imaging of two different 3D surface points. However, since most object are piece-wise smooth and illumination ratio is a smooth function of the surface normal, $r(x, y)$ should be spatially smooth, discontinuities only happen at places where $N(p)$ changes abruptly. This observation leads to the area-based method for illumination ratio estimation. The real and synthetic images can be blurred using a box filter. The illumination ratio is then computed as the relative scale of intensities of the corresponding pixels in the filtered images. Notice that blurring across regions with normal discontinuity should be prohibited.

## 2.3. Illumination Correction

Once the INM is computed, it can be applied to generate illumination corrected synthetic image. For each pixel $(x, y)$ in the synthetic image, its INM coordinate $(\theta, \phi)$ can be computed using back-projection as described earlier. The value of $\rho(\theta, \phi)$ in INM is then set to $r(x, y)$. When $(\theta, \phi)$ is not a grid point of the INM, $\rho(\theta, \phi)$ can be computed by interpolating the ratio values of neighboring grid points. After acquiring $r(x, y)$, the corrected pixel intensity $\hat{I}^{-}(x, y)$ is computed as $I(x, y) * r(x, y)$. Applying this process to all the pixels occupied by the synthetic object will generate an image that is considered to be illuminated under the lighting condition of the real scene.

## 3. Kalman Filtering

The Kalman filter (KF) is a popular and time-tested Bayesian estimation tool [10]. While the theoretical assumptions about noise characteristics and models are usually violated, in practice the filter has proven very popular as it offers a robust and efficient means for on-line estimation. In the classical KF framework, an uncontrolled linear system is modeled as

$$\hat{X}_t^- = A\hat{X}_{t-1} + W_{t-1} \tag{4}$$

$$\hat{Z}_t = H_t\hat{X}_t^- + V_t \tag{5}$$

Eqn. (4) is known as the state-transition or process equation, where $t$ is the time step, $\hat{X}$ is the estimate of the real system state $X$, $\hat{X}^-$ is the *a priori* (predicted) state, $A$ is the deterministic state transition between steps, and $W$ is the process noise. Eqn. (5) is known as the measurement equation, where $\hat{Z}$ is the estimate of the real measurement $Z$, $H$ represents the linear observation function that relates the state of the system to its measurements, and $V$ is the measurement noise. $W$ and $V$ are assumed to be normally distributed with covariances $Q$ and $R$ respectively (used below), spectrally white, and independent of each other. In addition to estimating the system state $\hat{X}$, the filter also estimates the error covariance $P$. Similar to Eqn. (4) it is assumed the *a priori* error covariance $P^-$ can be modeled (predicted) using $P$ from the previous step as follows.

$$P_t^- = AP_{t-1}A^T + Q \tag{6}$$

A Kalman filter performs system state estimation using a prediction-correction mechanism. At each frame, the filter predicts $\hat{X}^-$ and $\hat{Z}$ using the deterministic portions of Eqn. (4) and Eqn. (5), and $P^-$ using Eqn. (9). It then corrects the state by subtracting the estimated from the real observations, and factoring the residual back into the estimated state. The *a posteriori* state and error covariance then serves as the basis for the next step. The *measurement update* equations are as follows.

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R)^{-1} \tag{7}$$

$$\hat{X}_t = \hat{X}_t^- + K_t (Z_t - H_t \hat{X}_t^-) \tag{8}$$

$$P_t = (I - K_t H_t) P_t^- \tag{9}$$

In theory, the *Kalman gain* $K$ as computed in Eqn. (7) provides the optimal weighting of the observation residual in Eqn. (8), minimizing the mean of the diagonal of $P$.

While Eqn. (4)–Eqn. (9) model linear systems, the filter can be reformulated to accommodate a nonlinear process function $a(X)$ and/or observation function $h(X)$ using linear approximations about $\hat{X}$. In this *extended Kalman filter* (EKF) formulation the matrices $A$ and $H$ in Eqn. (4) and Eqn. (5) are replaced by the Jacobian matrices $A = \partial a/\partial \hat{X}$ and $H = \partial h/\partial \hat{X}$ respectively. Note that for our object pose estimation we actually used an *iterated* EKF. We describe this in Section 4.2. For more complete KF/EKF information the readers can refer to [17] for example.

## 4. Iterative Estimation

We implement our model-based tracking system using the Kalman filter framework. The state of the system $X$ is modeled as $[S, M, T]$, where $S$ is the object pose parameters, $M$ is the Illumination Normal Map and $T$ represents the texture of the object. Pixel intensities $I$ are used as the system measurements. The estimated measurement of a pixel at $(x, y)$ in the synthetic image is computed as:

$$I(x, y, S, M, T) = M(H_M(S, x, y)) \cdot T(H_T(S, x, y)) \tag{10}$$

where $H_M$ and $H_T$ are the back-projection functions. Given a pose estimate $\hat{S}$, they define the mapping from an image pixel at $(x, y)$ to an INM pixel at $(\theta, \phi)$ and a texture pixel at $(u, v)$. In fact, the camera calibration and object model are also implicitly used by $H_M$ and $H_T$. However, since they are constant throughout the image sequences, they are not included in the equation. One can see that Eqn. (10) can be divided into two parts. The fist part $M(H_M(S, x, y))$ defines the illumination correction. The second part $T(H_T(S, x, y))$ represents the texture mapping.

$H_M$ defines the mapping between discrete coordinates in the image and the INM. A pixel in the image may be mapped to a non-grid point in the INM. In this case, the illumination ratio of the pixel should be computed by interpolating neighboring grid points, where the values are defined. A similar issue holds for the image-texture mapping $H_T$. Moreover, the projection of a pixel in the texture space is an ellipsoid rather than a point [8]. When a relatively high-resolution texture is provided, such an ellipsoid covers multiple texture pixels. Thus, the value of the image pixel should be computed as a weighted average of the covered texture pixels. Eqn. (10) can then be rewritten as

$$\begin{aligned} I(x, y, S, M, T) = \sum_{\theta,\phi} M(\theta, \phi) \xi_M (H_M(S, x, y), \theta, \phi) \\ \cdot \sum_{u,v} T(u, v) \xi_T (H_T(S, x, y), u, v) \end{aligned} \tag{11}$$

where $\xi_M$ and $\xi_T$ are the filtering kernels centered at $H_M(S, x, y)$ and $H_T(S, x, y)$.

With the above system model definition, the Kalman filter can be used to estimate the object pose, Illumination Normal Map, and texture in a recursive fashion. The processing at each frame includes four steps: prediction, pose update, illumination estimation and texture refinement.

### 4.1. Prediction

We predict the state $\hat{X}^-$ based on the previous state estimate $\hat{X}$ as in Eqn. (4). Recall that the system state consists of the object pose $S$, texture $T$ and Illumination Normal Map $M$. $T$ is fixed for a given object. $S$ is predicted using certain motion model. For instance, constant-speed motion model is used in our system. $M$ can be predicted using different dynamic models. For instance, if we assume the scene lighting is static, $M$ would be constant, and an identity matrix should be used for the state-transition matrix $A$. However if for example the light source intensity and/or direction vary, a non-identity dynamic model could be used to predict $M$. The prediction step is important as it provides a good initialization for the remaining steps.

### 4.2. Pose Update

Here we assume that the INM and texture estimates $\hat{M}$ and $\hat{T}$ are fixed. The estimated intensity measurement $\hat{I}$ of a specific image pixel is then a nonlinear function of the object pose estimate $\hat{S}$ as in Eqn. (11). As described earlier, an Extended Kalman filter can be used to linearize $\hat{I}$ around $\hat{s}$. However, since $\hat{I}$ is highly nonlinear this linearization process can cause significant error if $\hat{S}$ is not close enough to the true pose. One approach to address this problem is to use an Iterated Extended Kalman filter (IEKF). The key point about the IEKF is that you iteratively refine the state *within* each step. The IEKF generates an estimate of the state, which can be used to predict the measurement, which can be used to update the state, which can again be used to predict the measurement, etc. A crucial point is that after each iteration, the measurement function is re-linearized around the newly updated state estimate. A detailed description of the IEKF can be found in [19].

## 4.3. Illumination Estimation

In this step, we estimate the values of INM pixels by comparing the image intensities of corresponding pixels in the synthetic and the real image. At this time, the pose $\hat{S}$ and the texture $\hat{T}$ are fixed. For a particular pixel at $(x_0, y_0)$ whose normal is mapped to $(\theta_0, \phi_0) = H_M(\hat{S}, x_0, y_0)$ with neighboring INM grid points $(\theta_k, \phi_k)$, the measurement function and Jacobian matrix can be written as

$$\hat{I}(\hat{M}) = c(x_0, y_0) \sum_{k=1}^{n} \hat{M}(\theta_k, \phi_k) \xi_M(\theta_0, \phi_0, \theta_k, \phi_k) \quad (12)$$

$$H(\hat{M}) = c(x_0, y_0)[\xi_M(\theta_0, \phi_0, \theta_1, \phi_1), ..., \xi_M(\theta_0, \phi_0, \theta_n, \phi_n)] \quad (13)$$

where $c(x_0, y_0) = \sum_{u,v} \hat{T}(u, v)\xi_T(H_T(\hat{S}, x_0, y_0), u, v)$ is the intensity of pixel $(x_0, y_0)$ computed directly from texture mapping, $\xi_M(\theta_0, \phi_0, \theta_k, \phi_k)$ is the interpolation kernel used to compute $\hat{M}(\theta_0, \phi_0)$, and $n$ is the size of the interpolation window. Currently, we compute $\hat{M}(\theta_0, \phi_0)$ using bi-linear interpolation of the surrounding $n = 4$ pixels.

The intensity measurement in Eqn. (12) is a linear function of $\hat{M}$. This means the standard Kalman measurement update equations can be applied to estimate $\hat{M}$. The intensity measurement for each pixel is sent to the Kalman filter one at a time. Then $H$ is an $n$-vector with elements $w_k = c(x_0, y_0)\ \xi_M(\theta_0, \phi_0, \theta_k, \phi_k)$, Kalman gain $K$ is an $n$-vector and the noise covariance $R$ becomes a scaler $\sigma^2$. In addition, we make the assumption that pixel values in the INM are independent of each other. In this case, the process covariance $P$ is an $n \times n$ diagonal matrix. Given the definition of these matrices, the measurement update equations Eqn. (7)-Eqn. (9) can be transformed into

$$K_k = P_{kk}^{-} w_k (\sum_{j=1}^{n} (P_{jj}^{-} w_j^2) + \sigma^2)^{-1} \quad (14)$$

$$\hat{M}_k = \hat{M}_k^{-} + K_k(I(x_0, y_0) - \hat{I}(x_0, y_0)) \quad (15)$$

$$P_{kk} = (1 - K_k w_k) P_{kk}^{-} \quad (16)$$

From the above equations one can see that the residual of intensity measurement of an image pixel is used to update the values of the INM pixels that contribute in computing its illumination ratio.

## 4.4. Texture Refinement

The texture refinement is performed in a way similar to the illumination estimation. The pose $\hat{S}$ and illumination $\hat{M}$ are held constant. Thus for a synthetic pixel at $(x_0, y_0)$ with texture coordinate $(u_0, v_0) = H_T(\hat{S}, x_0, y_0)$ we can derive the following equations

$$\hat{I}(\hat{T}) = l(x_0, y_0) \sum_{k=1}^{n} \hat{T}(u_k, v_k)\ \xi_T(u_0, v_0, u_k, v_k) \quad (17)$$

$$H(\hat{T}) = l(x_0, y_0)[\xi_T(u_0, v_0, u_1, v_1), ..., \xi_T(u_0, v_0, u_n, v_n)] \quad (18)$$

where $l(x_0, y_0) = \sum_{\theta,\phi} \hat{M}(\theta, \phi)\xi_M(H_M(\hat{S}, x, y), \theta, \phi)$ is the illumination ratio at $(x_0, y_0)$, and $(u_k, v_k)$ are the $n$ texture pixels inside the projection ellipsoid of pixel $(x_0, y_0)$. We can see that $\hat{I}$ is a linear combination of texture intensities. Currently we use equal weights for all the texture pixels, i.e. $\xi_T(u_0, v_0, u_k, v_k)$ equals to $1/n$ if $(u_k, v_k)$ is inside the projection ellipsoid around $(u_0, v_0)$, or 0 otherwise.

The measurement estimate $\hat{I}(\hat{T})$ in Eqn. (17) is a linear function of $\hat{T}$. Therefore, we can use a linear Kalman filter to estimate the texture $\hat{T}$ using Eqn. (14)-Eqn. (16). The elements in the $H$ vector become $w_k = l(x_0, y_0)\ \xi_T(u_0, v_0, u_k, v_k)$.

## 5. Experiment Results

We have tested our method with both synthetic and real data. First we present results using synthetic data. The synthetic scene consists of one textured sphere (an earth model), two cameras, and two directional light sources. One of the light sources remains static, and the other changes its direction. Synthetic image sequences rendered from the two camera views are used for tracking and illumination estimation. Fig. 2 demonstrates the illumination correction process. The INMs in (c) are computed by comparing pixel intensities in image pairs (a,d) and (b,e). The INM is then used to transform (d) to generate illumination corrected image (f). Although (a) and (b) are textured differently, they are generated using the same illumination parameters. As shown in (c), the INMs computed from (a) and (b) are very similar. This demonstrates that our INM estimation method is insensitive to the surface texture. Fig. 3 shows the tracking and INM estimation of the synthetic sphere under varying lighting conditions. One can tell from the estimated INMs that the direction of the light source changes. The resolution of the INMs shown in Fig. 2 and Fig. 3 is $72 \times 36$. Each pixel in the INM represents an interval of $5 \times 5$ degrees.

We also tested the proposed method on real data. We used two calibrated and synchronized Point Grey Dragonfly cameras to capture image sequences of an ambient cube undergoing 6D free motion. The cameras were set to capture at 15 frames per second. Currently camera photo-consistency is not enforced, thus an INM is estimated for each of the two cameras. The scene lighting environment consists of the ambient light and two strong distant light sources. The pose of the cube is manually initialized. Fig. 4 illustrates the illumination correction. As shown in Fig. 5, without illumination correction the system loses tracking within a few frames. The texture refinement result is presented in Fig. 6. One can see that the texture extracted from real image (d) is apparently darker, which shows that there is a significant illumination difference between the model and the real scene. However the refined texture (c) estimated using illumination-corrected measurements still preserves the luminance inherent in the initial texture input (b).

## 6. Future Work

The illumination ratio is a smooth function under Lambertian reflectance. Thus our method only estimates values for a sparse set of INM grid points and recovers the entire INM through interpolation. In fact, due to the low-pass-filtering property of Lambertian reflectance, the illumination can be closely approximated using nine coefficients of a spherical harmonic basis, as proved in [13, 1]. This indicates that we can further reduce the state dimensionality of the Kalman filter through estimating the coefficients of the spherical harmonics.

At the moment we use the graphics hardware to render a predicted image *without* illumination correction, then we transfer that image back to main memory and perform all of the subsequent per-pixel computations using the CPU. However we believe that we could implement the illumination correction directly on the graphics card while rendering, and then transfer that image to texture memory, re-rendering lower resolution versions instead of blurring, etc. In addition to the base texture of the object model, there could be a normal (vector) texture and an INM texture, both which are used by Cg programs (for example) that implement the per-pixel computation in parallel.

## Acknowledgements

This effort is primarily supported by National Library of Medicine contract N01-LM-3-3514: "3D Telepresence for Medical Consultation: Extending Medical Expertise Throughout, Between and Beyond Hospitals," and in part by NSF grant EIA-0303590: "Ubiquitous Pixels: Transforming Collaboration & Teaching with Pervasive Wall Displays."

## References

[1] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(2), February 2003.

[2] A. J. Bray. Tracking objects using image disparities. *Image and Vision Computing*, 8(1):4–9, February 1990.

[3] M. L. Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An apporach based on registration of texture-mapped 3d models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(6), June 1999.

[4] G. D. Cubber, H. Sahli, H. Ping, and E. Colon. A colour constancy approach for illumination invariant color target tracking. In *proceeding of the IARP Workshop on Robots for Humanitarian Demining*, 2002.

[5] F. Dellaert, S. Thrun, and C. Thorpe. Jacobian images of superresolved texture maps for model-based motion estimation and tracking. In *proceeding of the Fourth Workshop on Applications of Computer Vision*, 1998.

[6] D. Freedman and M. W. Turek. Illumination-invariant tracking via graph cuts. In *proceeding of the International Conference on Computer Vision and Pattern Recognition*, 2005.

[7] D. B. Gennery. Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, 7(3):243–270, April 1992.

[8] N. Greene and P. S. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6), June 1986.

[9] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(10), November 1998.

[10] R. E. Kalman. A new approach to linear filtering and prediction problems. *In Transactions of the ASME Journal of Basic Engineering*, 82, 1960.

[11] E. Munoz, J. M. Buenaposada, and L. Baumela. Efficient model-based 3d tracking of deformable objects. In *proceeding of the International Conference on Computer Vision and Pattern Recognition*, 2005.

[12] K. Nickels and S. Hutchinson. Model-based tracking of complex articulated objects. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 17:28–36, 2001.

[13] R. Ramamoorthi and P. Hanrahan. On the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. *Journal of the Optical Society of America*, 2001.

[14] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *proceedings of the Fifth International Conference on Computer Vision*, 1995.

[15] B. Rosenhahn and G. Sommer. Pose estimation of free-form objects. In *proceedings of the European Conference on Computer Vision*, 2004.

[16] B. Stenger, P. R. S. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. In *proceeding of the International Conference on Computer Vision and Pattern Recognition*, 2001.

[17] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[18] J. Zhang, J. Yu, and L. McMillan. Robust tracking and stereo matching under variable illumination. In *(to appear) the International Conference on Computer Vision and Pattern Recognition*, 2006.

[19] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *International Journal of Image and Vision Computing*, 25:59–76, 1997.

[20] M. Zobel, M. Fritz, and I. Scholz. Object tracking and pose estimation using light-field object models. In *proceeding of VISION, MODELING, AND VISUALIZATION*, 2002.

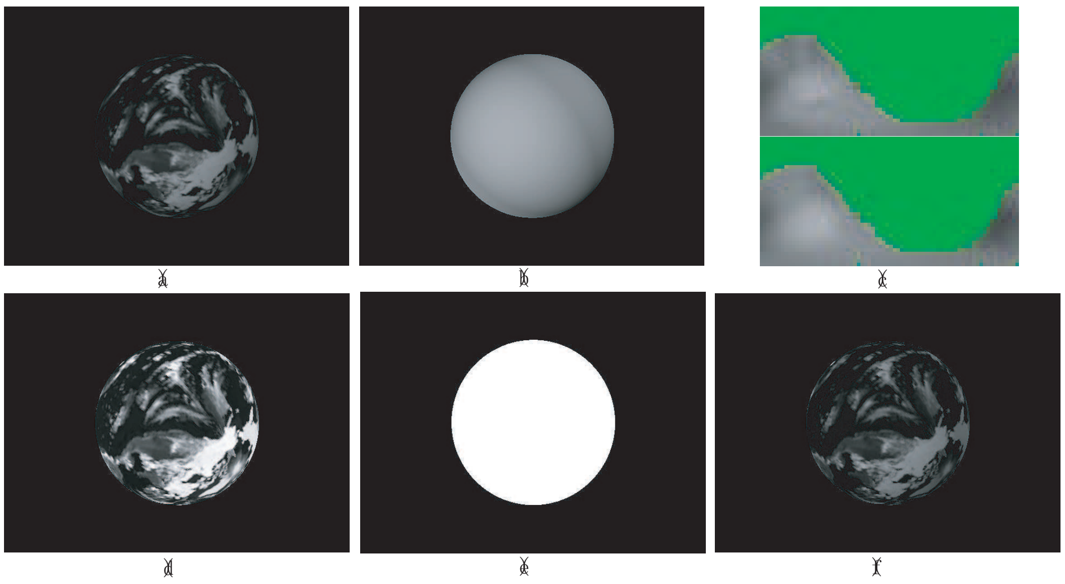

**Figure 2. Illustration of illumination correction. (a) Illuminated textured sphere. (b) Illuminated texture-less white sphere. (c) Upper: INM computed by comparing (a) and (d). Lower: INM computed by comparing (b) and (e). (d) Texture mapped sphere. (e) Solid white sphere. (f) Illumination corrected textured sphere transformed from (d) using the upper INM in (c). Notice that (a) and (b) are rendered under the same lighting condition.**

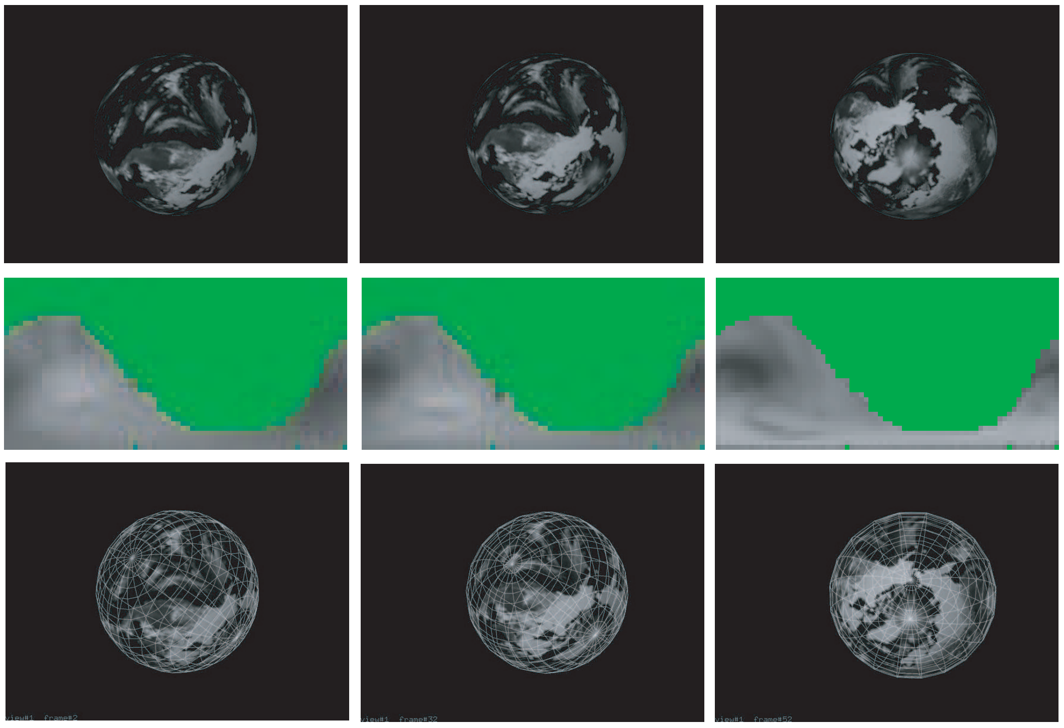

**Figure 3. Tracking and INM estimation for synthetic image sequences with varying illuminations. The first row shows the original synthetic images at frames 3, 33 and 53. The second row are the INM estimated at these frames. One can see that the lighting direction changes over frames. The last row represents the tracking results. Wire-frame sphere rendered using the estimated pose parameter are superimposed onto the original images.**

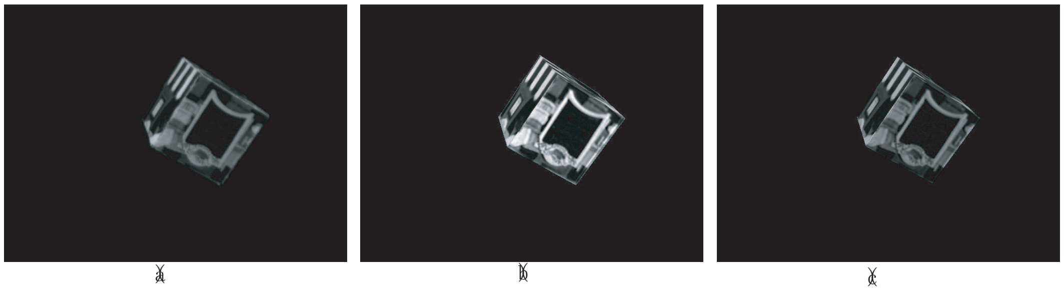

Figure 4. Illumination correction on real data. (a) Real image. (b) Synthetic image without illumination correction. (c) Illumination corrected synthetic image.

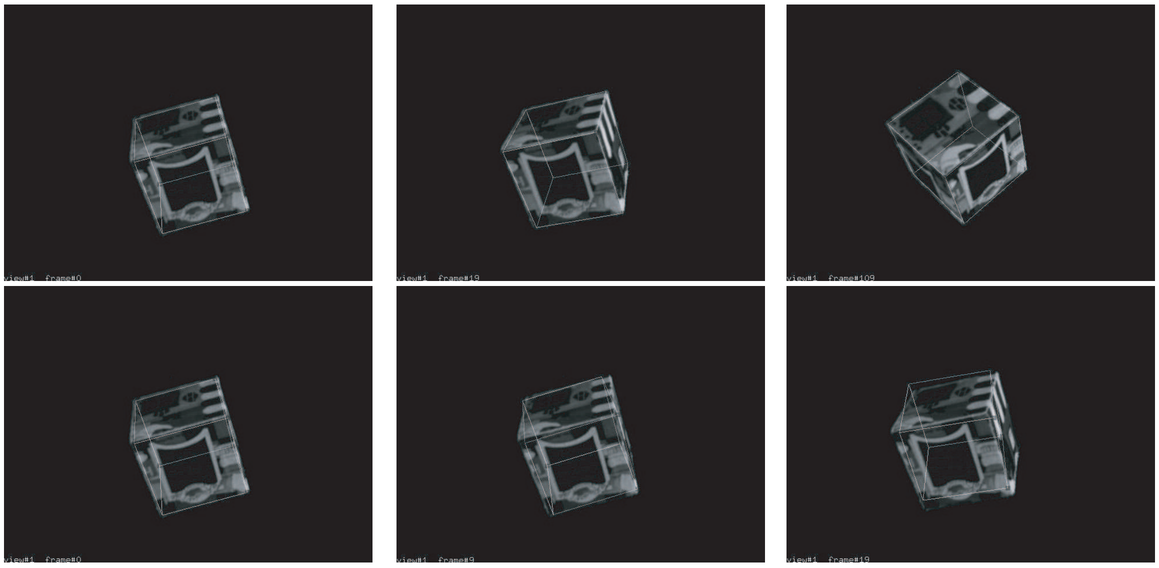

Figure 5. Tracking of a cubic object under static directional lighting. Wire-frame cube rendered using the estimated pose parameter are superimposed onto the original images. The first row shows the tracking results (frames 1, 20 and 110) with illumination correction. The second row shows the tracking results (frames 1, 10 and 20) without illumination correction.

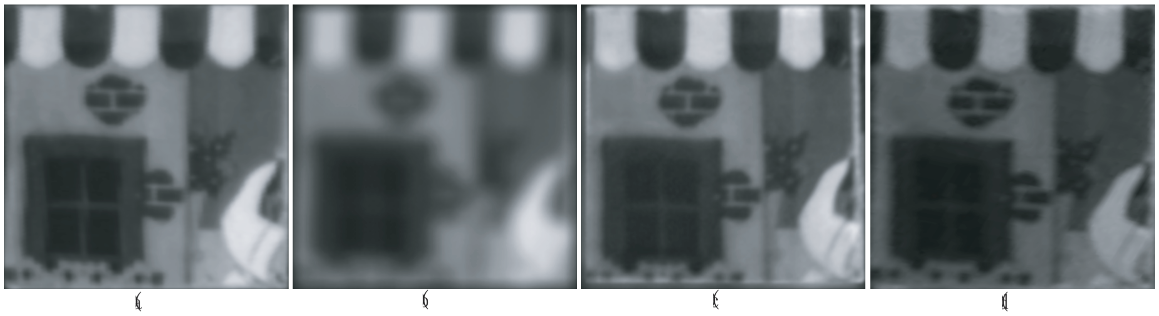

Figure 6. Texture refinement results. (a) Original texture. (b) Initial input texture generated by blurring (a). (c) Refined texture. (d) Texture directly extracted from a real image.