Jacobian Images of Super-Resolved Texture Maps for Model-Based Motion Estimation and Tracking

Frank Dellaert Sebastian Thrun Chuck Thorpe
Computer Science Department and The Robotics Institute
Carnegie Mellon University, Pittsburgh PA 15213

Abstract

We present a Kalman filter based approach to perform model-based motion estimation and tracking. Unlike previous approaches, the tracking process is not formulated as an SSD minimization problem, but is developed by using texture mapping as the measurement model in an extended Kalman filter. During tracking, a super-resolved estimate of the texture present on the object or in the scene is obtained. A key result is the notion of Jacobian images, which can be viewed as a generalization of traditional gradient images, and represent the crucial computation in the tracking process. The approach is illustrated with three sample applications: full 3D tracking of planar surface patches, a projective surface tracker for uncalibrated camera scenarios, and a fast, Kalman filtered version of mosaicking with detection of independently moving objects.

1. Introduction

This paper deals with model based motion estimation and tracking in video-streams, an area of intense research with many applications [2]. The motion we consider can be due to a moving object in the scene, the camera motion, or both. We expect that prior knowledge about the application led to the formulation of a model, whose *state variables* x we will track over time, using information contained in *measurements* z_i. As an example, consider a planar surface patch moving in 3D, and observed in a monocular video stream. We use a six-variable state vector $\mathsf{x} = [X\ Y\ Z\ \psi\ \theta\ \phi]^T$ to characterize the pose of the patch, and each measurement z_i is a vector of pixels.

We present a Kalman filter approach to the motion estimation problem, in which *texture mapping* is used as the measurement model. A *texture map* of the object or scene is incorporated in the model state and is estimated along with the pose state variables. Texture mapping is then used to predict the measurements and to revise the pose estimate. Then, the texture map part of the state is updated using the newly recovered pose. This two-step makes the approach computationally feasible. If the texture map is kept at a higher resolution than the input image, super-resolved texture maps can be built during the tracking process. This has applications in model building, image restoration and robot vision. Our approach can also be used to register incoming images in a pre-existing texture map, possibly at lower resolution. This has applications in (robot) localization, and can also be viewed as an alternative approach to mosaicking.

In what follows, we first pose the problem in a Bayesian framework in Section 2. This will naturally lead to a Kalman filter based approach. In Section 3 we discuss the use of texture mapping as the measurement model, and in Section 4 we explain how it can be used as part of an iterated Kalman filter, to track motion parameters over time. Of crucial importance here is the calculation of the *Jacobian images*, which we elaborate on in Section 5. Finally, in the applications section (Section 6) we present three different applications in which this approach has been used, after which we conclude in Section 7.

2. A Bayesian View

In this section we first view the motion estimation problem in the general framework of Bayesian estimation theory. We will be interested in tracking the state x of a model for an object or a scene, given as input one or more video-streams. In the Bayesian tradition, we will base our estimate $\hat{\mathsf{x}}(t)$ on the posterior probability density $P(\mathsf{x}|\mathsf{Z}_0^t = \mathsf{z}_0 \ldots \mathsf{z}_t)$ of the state x given all measurements z_i up to the current time t. We can express this density in terms of a prior $P(\mathsf{x}|\mathsf{Z}_0^{t-1})$ and a likelihood $P(\mathsf{z}_t|\mathsf{x}, \mathsf{Z}_0^{t-1})$, using Bayes law:

$$P(\mathsf{x}|\mathsf{Z}_0^t) = \frac{P(\mathsf{z}_t|\mathsf{x}, \mathsf{Z}_0^{t-1})P(\mathsf{x}|\mathsf{Z}_0^{t-1})}{P(\mathsf{z}_t|\mathsf{Z}_0^{t-1})} \qquad (1)$$

Previous approaches have traditionally modeled the likelihood $P(\mathsf{z}_t|\mathsf{x}, \mathsf{Z}_0^{t-1})$ by assuming that the current image z_t is equal to the previous image z_{t-1}, warped according to x, and corrupted by noise. Under certain assumptions the

maximum likelihood solution is then equivalent to minimizing the sum of square differences (SSD), the approach followed in much of the literature [10, 2]. Note that in this case (a) no prior knowledge about x is used, and (b) the previous measurement z_{t-1} needs to be kept around.

The approach taken in this paper is to extend the state x so that we can predict z_t from x alone, satisfying the *Markov property* $P(\mathsf{z}_t|\mathsf{x}, \mathsf{Z}_0^{t-1}) = P(\mathsf{z}_t|\mathsf{x})$. For example, for a surface patch, we could explicitly estimate the texture on the surface and use it to predict the image. There are many advantages to taking this view. First, the texture map can be kept at a higher resolution than the images to yield superior predictions. Second, the estimated texture is a composite of all previous images in the video-stream, not just the last image. Finally, the texture would remain valid even when the surface patch is temporarily occluded.

In general, one needs to estimate *all* variables relevant to the appearance of the scene, including surface structure, surface reflectance characteristics and lighting conditions. Then, by modeling how the image measurement is obtained by the camera(s), we can derive an expression for $P(\mathsf{z}_t|\mathsf{x})$. If this density is Gaussian, and if the dynamics of the model can be described using a linear system, then the *Kalman filter* [11] can be used to efficiently evaluate Bayes law (1) over time. This approach has been used before in feature-based approaches to structure from motion [3, 1], visual servoing [13], and model-based tracking [4]. By making the surface characteristics part of the state, and using an appropriate measurement model, the set of tools provided by optimal estimation theory, foremost the Kalman filter, can be applied equally well to iconic or image based approaches.

3. The Measurement Model

Here we discuss the use of texture mapping as the measurement model. Without loss of generality, we continue the discussion for the case of a planar surface patch observed in a video-stream, as introduced above. Thus, we extend the state estimate to $\hat{\mathsf{x}} = \{\hat{\mathsf{x}}_p, \hat{\mathcal{T}}\}$, where $\hat{\mathsf{x}}_p$ collects the pose variables, and $\hat{\mathcal{T}}$ is a texture map modeling the actual texture $\mathcal{T}$ present on the patch. We will refer to the pixels of this texture map as *mixels*, or *m*odel p*ixels*, to distinguish them from image pixels. In this case each *measurement* z_i consists of a collection of image intensity values $\mathcal{I}(\mathsf{p}, \mathsf{x}_p, \mathcal{T})$, one for each pixel p in the area occupied by image of the patch. As indicated, these pixel values will be a function of both the pose x_p and the texture $\mathcal{T}$. Predicting the measurement can then be reformulated as asking the question: *given $\hat{\mathsf{x}}_p$ and $\hat{\mathcal{T}}$, what is the value $\mathcal{I}(\mathsf{p}, \hat{\mathsf{x}}_p, \hat{\mathcal{T}})$ of each pixel in the image of the patch ?* This is exactly the problem addressed by texture mapping [7, 12].

The most basic form of texture mapping simply inverts the *mapping* m between (homogeneous) texture coordinates (s, t, u) and image coordinates (x, y, w). In this scheme, each pixel is inverse-mapped to its *pre-image* in texture space, and assigned the value of the nearest mixel [12]:

$$\mathcal{I}(\mathsf{p}, \hat{\mathsf{x}}_p, \hat{\mathcal{T}}) = \hat{\mathcal{T}}(round(\mathsf{m}^{-1}(\mathsf{p}))) \tag{2}$$

The dependence on the pose estimate $\hat{\mathsf{x}}_p$ is subsumed in the mapping m. In the case of a planar patch, m is simply a projective mapping, characterized by a 3×3 matrix ${}^i_t\mathsf{T}(\mathsf{x}_p)$, that transform points k in texture space to pixels p in image space. This *homography* is invertible, and we can apply inverse mapping as in (2).

However, as we are resampling from one discrete grid (the texture) to another (the image), *aliasing* can occur if the warping process introduces spatial frequencies in the image that exceed the Nyquist sampling frequency. As this will negatively affect our ability to perform motion estimation, we need to prevent aliasing from occurring. Ideally, this is done by first reconstructing the continuous texture signal using a sync filter, warping it according to the mapping m, and then pre-filtering it with an ideal low-pass filter that cuts off undesired high frequencies. By combining these two filters the predicted value for each pixel p can be obtained by convolving the texture image $\hat{\mathcal{T}}$ with a *resampling filter* ρ centered around $\mathsf{m}^{-1}(\mathsf{p})$: [7]

$$\mathcal{I}(\mathsf{p}, \hat{\mathsf{x}}_p, \hat{\mathcal{T}}) = \sum_{k \in \mathcal{Z}^2} \hat{\mathcal{T}}(\mathsf{k})\, \rho(\mathsf{m}^{-1}(\mathsf{p}), \mathsf{k}) \tag{3}$$

In practice we use a Gaussian low-pass filter for both the reconstruction filter and the prefilter. In the planar case, this has the convenient property that the combined filter is again a (warped and space-variant) Gaussian filter in texture space [7, 5]. The resulting filter yields high quality predictions for the image measurement, and has smooth and continuous derivatives, which will be required below.

4. Recursive Motion Estimation

At this point we are in a position to recursively obtain $P(\mathsf{x}|\mathsf{Z}_0^t)$ by evaluating Bayes law. Texture mapping yields a measurement prediction $\hat{\mathsf{z}} = \mathsf{h}(\hat{\mathsf{x}})$ that, assuming Gaussian additive noise, yields a Gaussian likelihood density $P(\mathsf{z}_t|\mathsf{x})$ centered around $\hat{\mathsf{z}}$ and with covariance matrix R. The measurement noise covariance R is typically taken to be diagonal, i.e., individual pixel measurements are assumed to be conditionally independent given x.

If the measurement model is linear, i.e., described by a matrix equation $\mathsf{z} = \mathsf{H}\mathsf{x}$, and corrupted by Gaussian white noise, then the Kalman filter (KF) can be used to efficiently evaluate Bayes law at each time step. Indeed, under those assumptions the density $P(\mathsf{x}|\mathsf{Z}_0^t)$ will remain Gaussian at all times (provided x is propagated over time using linear dynamics), and can be characterized using only two quantities: a mean $\hat{\mathsf{x}}$ and a covariance matrix P. At each time step

then, the incoming measurement z can be integrated using the standard KF measurement update equations [11]:

$$\mathsf{K} = \mathsf{P}\,\mathsf{H}^T\,[\mathsf{H}\,\mathsf{P}\,\mathsf{H}^T + \mathsf{R}]^{-1} \quad (4)$$

$$\hat{\mathsf{x}} \Leftarrow \hat{\mathsf{x}} + \mathsf{K}\,[\mathsf{z} - \mathsf{h}(\hat{\mathsf{x}})] \quad (5)$$

$$\mathsf{P} \Leftarrow \mathsf{P} - \mathsf{K}\,\mathsf{H}\mathsf{P} \quad (6)$$

Here the gain matrix K is used to weigh how much and in what direction the state estimate $\hat{\mathsf{x}}$ is updated in function of the difference between the predicted measurement $\mathsf{h}(\hat{\mathsf{x}})$ and actual measurement z.

4.1. Recursive Estimation Overview

Each iteration consists of three phases: prediction, pose registration and texture update. In the first phase, we predict the state x by using a model for the dynamics of the system. The ability to do this is a crucial advantage of using a Kalman filter, as it makes the ensuing image registration process easier, by providing a good initial state estimate. The two remaining phases comprise the measurement update step. Because the texture mapping process h described by (3) is linear in the texture variables $\hat{\mathcal{T}}(\mathsf{k})$, but highly non-linear in the pose variables $\hat{\mathsf{x}}_p$, we have taken a two-tier approach. First, the model is registered by estimating the pose variables $\hat{\mathsf{x}}_p$ using an extended Kalman filter (see below). Only then the texture map $\hat{\mathcal{T}}$ is updated using a linear KF. In summary, the recursive estimation approach we propose follows the usual KF structure, albeit with the measurement update step split into two phases:

1. Dynamics (Section 4.2): use a motion model to predict the pose x_p at the next time step.
2. Estimate Pose (Section 4.3): incorporate an image measurement z to estimate the *pose* $\hat{\mathsf{x}}_p$.
3. Estimate Texture (Section 4.4): update the *texture* estimate $\hat{\mathcal{T}}$ using the newly aligned image.

4.2. Dynamics

The Kalman filter *dynamics update* equations are used to propagate the state estimate $\hat{\mathsf{x}}$ forward in time. In our current work, we have used various motion models to predict the changing pose, depending on the application. In principle, although we have not yet attempted this, one could also model time-variant properties of the texture, e.g. changes in lighting. One could even attempt to track a television screen by modeling the changes in the televised image.

In most cases, we have simply used a linear constant velocity model, with appropriate noise terms to account for acceleration. One important approximation we make is neglecting the cross-correlation terms between texture and position variables. The pose estimate $\{\hat{\mathsf{x}}_p(t), \mathsf{P}_p\}$ is then propagated by integrating the dynamics f forward in time until the next measurement is available:

$$\dot{\hat{\mathsf{x}}}_p(t) = \mathsf{f}[\hat{\mathsf{x}}_p(t), \mathsf{u}(t)] \quad (7)$$

$$\dot{\mathsf{P}}_p(t) = \mathsf{F}(t)\,\mathsf{P}_p(t) + \mathsf{P}_p(t)\,\mathsf{F}^T(t) + \mathsf{G}\,\mathsf{Q}\,\mathsf{G}^T \quad (8)$$

Here $\mathsf{F}(t)$ is the Jacobian of the system dynamics f, and Q is the covariance kernel of the dynamic driving noise w. If f is non-linear, $\mathsf{F}(t)$ represents the (time-varying) linearization of f around the estimate $\hat{\mathsf{x}}_p(t)$.

4.3. Pose Update

For the pose update the texture part $\hat{\mathcal{T}}$ of the state is held constant, and the pose $\hat{\mathsf{x}}_p$ is updated using the *iterated extended Kalman filter* (IEKF), explained below. This process is the counterpart of the conventional image registration techniques in the literature. In fact, it can be shown that the IEKF is equivalent to Gauss-Newton non-linear optimization. However, unlike previous work, here we are maximizing a *posterior* probability and not a likelihood, as the dynamics update step yields a prior on the state $\hat{\mathsf{x}}_p$.

The extended KF simply uses equations (4-6), but linearizes the measurement function h at each step, by means of the *measurement Jacobian* H:

$$\mathsf{H}(\hat{\mathsf{x}}_p) \stackrel{\text{def}}{=} \left.\frac{\partial \mathsf{h}(\mathsf{x}_p, \hat{\mathcal{T}})}{\partial \mathsf{x}_p}\right|_{\mathsf{x}_p = \hat{\mathsf{x}}_p} \quad (9)$$

The IEKF is based on the idea that the Jacobian H and the prediction $\mathsf{h}(\hat{\mathsf{x}}_p)$ are likely to be of better quality *after* the update has been done, since the new state estimate is presumably closer to the true state. Repeating this process multiple times leads to the IEKF update algorithm [11, 3]:

$$\mathsf{K}_n = \mathsf{P}\,\mathsf{H}(\hat{\mathsf{x}}_n)^T\,[\mathsf{H}(\hat{\mathsf{x}}_n)\,\mathsf{P}\,\mathsf{H}(\hat{\mathsf{x}}_n)^T + \mathsf{R}]^{-1} \quad (10)$$

$$\hat{\mathsf{x}}_{n+1} \Leftarrow \hat{\mathsf{x}}_0 + \mathsf{K}_n\,[\mathsf{z} - \mathsf{h}(\hat{\mathsf{x}}_n, \hat{\mathcal{T}}) - \mathsf{H}(\hat{\mathsf{x}}_n)[\hat{\mathsf{x}}_0 - \hat{\mathsf{x}}_n]] \quad (11)$$

Here $\hat{\mathsf{x}}_0$ is initialized to the estimate $\hat{\mathsf{x}}$ before the update, and the above equations are iterated in n until convergence.

4.4. Texture Update

After the pose update, the newly aligned image measurement is incorporated to refine the texture estimate $\hat{\mathcal{T}}$. The usual KF measurement update equations are used to update the texture, but to keep the computation tractable we make a number of approximations. Since the estimated texture is typically large, it is infeasible to maintain a full covariance matrix. Instead, we neglect *all* cross-correlation terms between neighboring texture mixels, i.e., we assume a diagonal texture covariance matrix P_t. Furthermore, we currently neglect any mixel-pose cross-correlation terms.

The mixel update equations then become particularly elegant. From (3) one can see that each row of the measurement Jacobian H will simply contain the resampling

weights $w_{pk} = \rho(\mathsf{m}^{-1}(\mathsf{p}), \mathsf{k})$. In addition, we have a diagonal covariance matrix P_t, having as elements the mixel variances P_{kk}. If the measurement noise covariance R is also diagonal, with elements σ_p^2, we can integrate one pixel measurement at a time, and the innovation $v(\mathsf{p}) = z(\mathsf{p}) - \mathcal{I}(\mathsf{p}, \hat{\mathsf{x}}_p, \hat{\mathcal{T}})$ reduces to a scalar. It can then be easily derived that the update equations (4-6) for mixel k become:

$$K_{kp} = P_{kk} w_{pk} / (\sigma_p^2 + \sum\nolimits_j P_{jj} w_{pj}^2) \tag{12}$$

$$\hat{\mathcal{T}}(\mathsf{k}) \Leftarrow \hat{\mathcal{T}}(\mathsf{k}) + K_{kp} v(\mathsf{p}) \tag{13}$$

$$P_{kk} \Leftarrow P_{kk}(1 - w_{pk} K_{kp}) \tag{14}$$

Remarkably, these simple equations produce quite satisfactory super-resolved estimates of the texture $\mathcal{T}$. They are also intuitively appealing: the gain K_{kp} determines how much a pixel measurement $z(\mathsf{p})$ contributes to a mixel k. This gain is higher for mixels with high resampling weight w_{pk}, and decreases as the noise variance σ_p^2 increases.

5. Jacobian Images

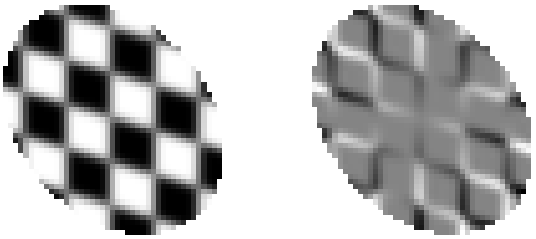

Figure 1. A texture mapped planar patch and I_ψ, its partial derivative with respect to yaw (rotating around Z).

In this section we elaborate on the calculation of the measurement Jacobian in equations (10-11). In our case, $\mathsf{H}(\hat{\mathsf{x}}_p)$ contains the partial derivatives of the texture mapping process with respect to the pose state variables x_p. These partials can also be visualized as images. For example, in the case of a planar surface patch, the pose is characterized by the 6 variables $\mathsf{x}_p = [X\ Y\ Z\ \psi\ \theta\ \phi]^T$. Thus, H will be composed of six partials, each expressing how the image of the patch will change in response to a small change in position or orientation. As an illustration, Figure 1 shows a checkerboard pattern mapped onto a patch, and the difference image that is generated when the patch is rotated an infinitesimal amount around its surface normal. As expected, most change occurs furthest away from the rotation center, while the change incurred at the center itself is zero.

We call this partial derivative image the *Jacobian image* with respect to yaw ψ, and the way in which it is obtained provides considerable insight into the texture tracking process. If we rewrite the resampling filter ρ using separate, scalar functions $s(\mathsf{p})$ and $t(\mathsf{p})$ for the texture coordinates of $\mathsf{m}^{-1}(\mathsf{p})$, (3) becomes:

$$\mathcal{I}(\mathsf{p}, \hat{\mathcal{T}}) = \sum_{k \in \mathcal{Z}^2} \hat{\mathcal{T}}(\mathsf{k})\, \rho(s(\mathsf{p}), t(\mathsf{p}), \mathsf{k}) \tag{15}$$

Taking the partial derivative of (15) with respect to (for example) yaw ψ, we get:

$$\begin{aligned} \frac{\partial \mathcal{I}(\mathsf{p}, \hat{\mathcal{T}})}{\partial \psi} =& \frac{\partial s(\mathsf{p})}{\partial \psi} \sum_{k \in \mathcal{Z}^2} \hat{\mathcal{T}}(\mathsf{k})\, \frac{\partial \rho(s(\mathsf{p}), t(\mathsf{p}), \mathsf{k})}{\partial s} \\ &+ \frac{\partial t(\mathsf{p})}{\partial \psi} \sum_{k \in \mathcal{Z}^2} \hat{\mathcal{T}}(\mathsf{k})\, \frac{\partial \rho(s(\mathsf{p}), t(\mathsf{p}), \mathsf{k})}{\partial t} \end{aligned} \tag{16}$$

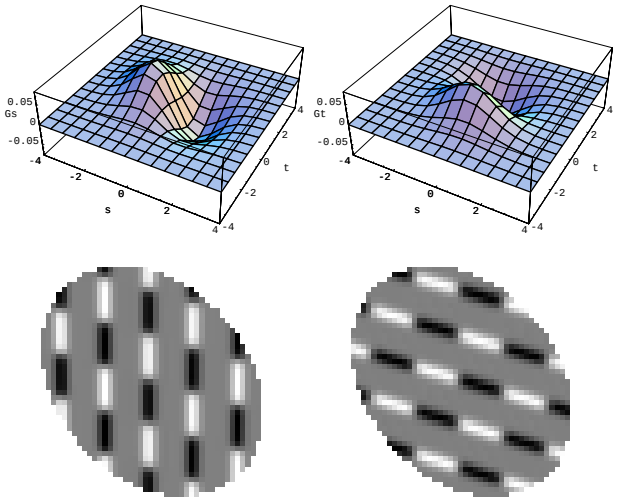

Figure 2. Top: gradient kernels in texture space. Bottom: predicted horizontal (left) and vertical gradient images.

As we use Gaussian low-pass filters for the resampling, the derivatives of the combined filter ρ are *derivative of Gaussian* filters. For the example in Figure 1 these gradient kernels are shown in Figure 2. Convolving the texture $\hat{\mathcal{T}}$ with these gradient kernels yields two *predicted gradient images*, also shown above (corresponding to the patch in Figure 1). The gradient filters require only an incremental amount of computation in addition to the resampling filter, and the resulting predicted gradients are of high quality.

Equation (16) can now be interpreted as follows: the Jacobian images, representing the partial derivatives of the image with respect to one pose variable, can be obtained as a linear combination of the two gradient images. The coefficients of the linear combination are *pixel dependent* and are the components of a vector field, induced by the motion of the patch. In the planar surface patch example, the vector field induced by yaw is shown in Figure 3. This is used to combine the gradient images of Figure 2 into the Jacobian image I_ψ of figure 1. Since a patch has six degrees of

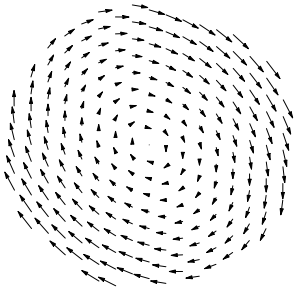

Figure 3. Vector field induced by a change in yaw ψ.

freedom there are 6 vector fields, yielding 6 different Jacobian images. These vector fields correspond to the motion *in texture space* of the pre-images of the pixels, as a result of a change in one of the pose parameters. Intuitively, if one casts rays from the camera center to the surface patch, these rays intersect with the patch at specific locations. When the patch moves, these locations move as well.

In summary, the calculation of the measurement Jacobians H proceeds as follows: (a) calculate the predicted gradient images; (b) calculate the vector fields; (c) combine them as in (16). In practice, we never calculate complete Jacobian images as shown, but proceed on a pixel per pixel basis. This is useful, as pixels can be selectively integrated one at a time, until the pose uncertainty has dropped below a given threshold. This yields considerable savings.

6. Applications

In this section, we discuss three application settings in which we have applied this approach. All three applications involve the planar case, and concern monocular video-streams. However, no important difficulties should arise in extending the approach to more general surface models and multiple video-streams.

6.1. Super-Resolved Tracking in 3-D

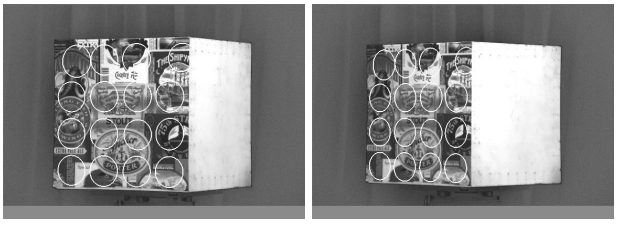

Figure 4. 16 'stickers' tracking the textured face of a cube.

A first application involves the tracking of planar surface patches, moving in 3D, as already introduced above. As an illustration, Figure 4 shows two frames from an image sequence of a textured cube, which is tracked by 16 planar surface patches in parallel. Because of the way these trackers stick to the surface of the cube, we have named them *stickers*. Each sticker is modeled using a six-dimensional pose state vector $\hat{\mathsf{x}}_p = [X\, Y\, Z\, \psi\, \theta\, \phi]^T$, and an appropriately sized texture map. Knowledge of the camera calibration is used to calculate the homography m between the texture plane of the sticker and the image plane, which is then symbolically differentiated as part of the Jacobian calculation. Further details can be found in [5].

Figure 5. Super-resolution. Left: original image resolution. Right: estimated texture (in circle) after 15 frames.

Figure 5 highlights the super-resolution aspect of our approach. In this case, the cube was tracked as it was moving away from the camera. The original image resolution as seen by the camera is at left: it is hard if not impossible to read the words in the oval. The right panel shows the estimated texture after 15 frames of tracking the sequence, where now the words in the oval can be read. In this case, the texture map was kept at about four times the resolution of the original images.

6.2. A Projective Tracker

For situations in which the camera calibration is unknown or changing, we have also devised a *projective tracker* which estimates the changing homography between a plane in 3D and the image plane. In this case, the state $\hat{\mathsf{x}}_p$ consists of the *image* coordinates of 4 points known to lie in the same 3D plane. Again the Jacobian is derived by differentiating a symbolic expression of the homography m with respect to each of the 8 state variables. If properly initialized (e.g. by user interaction), this projective tracker can track the deforming quadrilateral without camera calibration. When combined with super-resolution, one application is *reading off moving objects*, e.g., trucks, passing traffic signs, or in general any planar surface in any video-stream that contains some interesting texture.

6.3. Fast Mosaicking

A final example illustrates the use of our technique as an alternative approach to mosaicking. The specific appli-

Figure 6. Top: Mosaicked scene. Bottom: Comparing the actual (left) with the predicted (right) measurement can serve to detect independently moving objects.

cation in which we have investigated this is the tracking of moving objects filmed by a hand-held camera. For example, Figure 6 shows the mosaic obtained from a sequence in which a moving truck was followed. As input we used down-sampled low-resolution images, and the mosaic was obtained by resampling image strips rather than using the texture update from section 4.4. Figure 6 shows how this mosaic is used to detect independently moving objects.

In contrast to the tracking applications above, here the estimated texture map is *larger* than the field of view, and the state consists of translation and rotation with respect to a reference frame. Advantages of using our method rather than the more conventional SSD minimization include (a) Kalman filtering can lead to improved tracking, (b) other sources of information can be easily integrated, (c) it can be much faster by only integrating selected pixels (in the example, only 100 pixels were used at each time step).

7. Conclusions

We propose using texture mapping as the measurement model in a Kalman filter approach to motion estimation. This technique is easy to apply in many different applications. In addition, we introduced the notion of Jacobian images, which are a crucial component of the approach, and have shown an intuitive way to view their computation. Finally, the key computational burden involves texture mapping, which is now typically hardware-accelerated even on low-end PC's. Thus, our approach could be well suited to enabling vision-based applications in the consumer domain.

We view our main contribution as introducing one unifying framework to derive and formalize many different aspects of image-based motion estimation and tracking. Several of the results or individual computational techniques we derive in this way were already present in the literature in one form of another. Most notably, Irani and Peleg have written several papers on super-resolution from motion analysis [8, 9], and Hager and Belhumeur's motion templates [6] are analogous to the Jacobian images presented here. The Bayesian estimation framework used here introduces a powerful new way to view these techniques in context, and to make explicit the assumptions underlying them.

References

[1] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(6):562, June 1995.

[2] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In G. Sandini, editor, *European Conference on Computer Vision*. Springer-Verlag, 1992.

[3] T. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *IEEE Trans. Aerospace and Electronic Systems*, 26(4):639–656, July 1990.

[4] F. Dellaert, D. Pomerleau, and C. Thorpe. Model-based car tracking integrated with a road-follower. In *Proceedings of IEEE Conference on Robotics and Automation (ICRA)*, Leuven, Belgium, 1998.

[5] F. Dellaert, C. Thorpe, and S. Thrun. Super-resolved tracking of planar surface patches. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1998.

[6] G. Hager and P. Belhumeur. Real time tracking of image regions with changes in geometry and illumination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 403–410, 1996.

[7] P. S. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, University of California, Berkeley, 1989.

[8] M. Irani and S. Peleg. Improving resolution by image registration. *Graphical Models and Image Processing*, 53:231–239, 1991.

[9] M. Irani and S. Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communication and Image Representation*, 4:324–335, 1993.

[10] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 674–679, 1981.

[11] P. Maybeck. *Stochastic Models, Estimation and Control*, volume 2. Academic Press, New York, 1982.

[12] D. F. Rogers. *Procedural Elements for Computer Graphics*. McGraw Hill, Boston, MA, second edition, 1998.

[13] W. W. Wilson. Visual servo control of robots using kalman filter estimates of robot pose realtive to work-pieces. In K. Hashimoto, editor, *Visual Servoing*. World Scientific, 1993.