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. In the process, 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 are the crucial components 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 [3]. The motion we consider can be due to a moving object in the scene, the camera motion, or a combination of both. In both cases, we expect that prior knowledge about the application led to the formulation of a model. We will track the *state variables* x of this model over time, using information contained in the *measurements* z_i. As an example, consider a planar surface patch moving in 3D, which is 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 the measurements z_i will consist of image regions.

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. The texture mapping process is then used to predict the measurements and to revise the pose estimate. 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 or previously estimated texture map, possibly at *lower* resolution. This has applications in (robot) localization, and can also be viewed as an alternative approach to video 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. Connections to related work are made along the way. 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, whereas the following sections will be more specific. 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 $\hat{\mathsf{x}}$ 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:

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

$P(\mathsf{z}_t|\mathsf{x}, \mathsf{Z}_0^{t-1})$ is traditionally modeled by assuming that the current image z_t is equal to the previous im-

age 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, 3]. 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.

An alternative approach 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. The texture map could be kept at a higher resolution than the images to yield superior predictions. In addition, the estimated texture is a composite of all previous images in the video-stream, not just the last image. And finally, it 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 [4, 1], visual servoing [13], and model-based tracking [9, 5]. 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. In this case each *measurement* z 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 *preimage* in texture space, and assigned the value of the nearest integer texture coordinate [12]:

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

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 ${}_t^i\mathsf{T}(\mathsf{x}_p)$, 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]

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

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, 6]. 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, as embodied by (1). 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[1], 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]:

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

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

$$\mathsf{P} \Leftarrow \mathsf{P} - \mathsf{K}\,\mathsf{H}\mathsf{P} \tag{6}$$

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

Unfortunately, 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 $\hat{\mathsf{x}}_p$. This suggests a two-tier approach, in which first the model is registered by estimating the pose variables $\hat{\mathsf{x}}_p$ using an extended Kalman filter (see below), and only then updating the texture map $\hat{\mathcal{T}}$ using a linear KF. Finally, we also predict the state x at the next time step using a model for the dynamics of the system.

In summary, the recursive estimation approach we propose follows the usual KF structure, albeit with the measurement update step split into two phases:

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

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

$$\mathsf{H}(\hat{\mathsf{x}}_p) \stackrel{\text{def}}{=} \left.\frac{\partial \mathsf{h}(\mathsf{x}_p, \hat{\mathcal{T}})}{\partial \mathsf{x}_p}\right|_{\mathsf{x}_p = \hat{\mathsf{x}}_p} \tag{7}$$

[1] provided x is propagated over time using linear dynamics.

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 iterated EKF measurement update algorithm [11, 4][2]:

$$\mathsf{K}_n = \mathsf{P}\,\mathsf{H}(\hat{\mathsf{x}}_n)^T\,[\mathsf{H}(\hat{\mathsf{x}}_n)\,\mathsf{P}\,\mathsf{H}(\hat{\mathsf{x}}_n)^T + \mathsf{R}]^{-1} \tag{8}$$

$$\hat{\mathsf{x}}_{n+1} \Leftarrow \hat{\mathsf{x}}_0 + \mathsf{K}_n\,[\mathsf{z} - \mathsf{h}(\hat{\mathsf{x}}_n, \hat{\mathcal{T}}) - \mathsf{H}(\hat{\mathsf{x}}_n)[\hat{\mathsf{x}}_0 - \hat{\mathsf{x}}_n]] \tag{9}$$

Here $\hat{\mathsf{x}}_0$ is initialized to the state estimate $\hat{\mathsf{x}}$ before the update, and the iteration in n is stopped after a fixed number of steps or when consecutive values $\hat{\mathsf{x}}_n$ and $\hat{\mathsf{x}}_{n+1}$ differ by less than a preselected amount [11].

4.3 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 cross-correlation terms between texture mixel values in $\hat{\mathcal{T}}$ and the pose $\hat{\mathsf{x}}_p$.

The update equations are of exactly the same form as (4-6). But now, the Jacobian H is given to us 'for free' by the texture mapping calculations. Indeed, H describes the derivative of image pixel intensity with respect to a change in texture mixel value. From (3) we see that this derivative is simply $\rho(\mathsf{m}^{-1}(\mathsf{p}), \mathsf{k})$.

The mixel update equations are then simple. Summarizing, we have a diagonal covariance matrix P_t, having as elements the mixel variances σ_{kk}^2, and we have the Jacobian H containing the resampling weights $w_k = \rho(\mathsf{m}^{-1}(\mathsf{p}), \mathsf{k})$. When integrating one pixel measurement at a time, the innovation $v(\mathsf{p}) = z(\mathsf{p}) - \mathcal{I}(\mathsf{p}, \hat{\mathsf{x}}_p, \hat{\mathcal{T}})$ reduces to a scalar, as does the measurement noise variance R. It can then be easily derived that the update equations (4-6) become:

$$K_k = \sigma_{kk}^2 w_k / (R + \sum\nolimits_j \sigma_{jj}^2 w_j^2) \tag{10}$$

$$\hat{\mathcal{T}}(\mathsf{k}) \Leftarrow \hat{\mathcal{T}}(\mathsf{k}) + K_k v(\mathsf{p}) \tag{11}$$

$$\sigma_{kk}^2 \Leftarrow \sigma_{kk}^2 (1 - w_k K_k) \tag{12}$$

Remarkably, these simple equations produce quite satisfactory super-resolved estimates of the texture $\mathcal{T}$.

[2] here we drop the subscript p for clarity's sake

4.4 Dynamics

The usual Kalman filter *dynamics update* step is used to propagate the pose estimate $\hat{\mathsf{x}}_p$ forward in time, given our current estimate and a motion model. One important approximation we make is neglecting the cross-correlation terms between texture and position variables. In most cases, we use a linear constant velocity model, with appropriate noise terms to account for acceleration. 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:

$$\dot{\hat{\mathsf{x}}}_p(t) = \mathsf{f}[\hat{\mathsf{x}}_p(t), \mathsf{u}(t)] \tag{13}$$

$$\dot{\mathsf{P}}_p(t) = \mathsf{F}(t)\,\mathsf{P}_p(t) + \mathsf{P}_p(t)\,\mathsf{F}^T(t) + \mathsf{G}\,\mathsf{Q}\,\mathsf{G}^T \tag{14}$$

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 linearization of f around the estimate $\hat{\mathsf{x}}_p(t)$ and will in general vary with time, *even if* f *is time-invariant.*

5 Jacobian Images

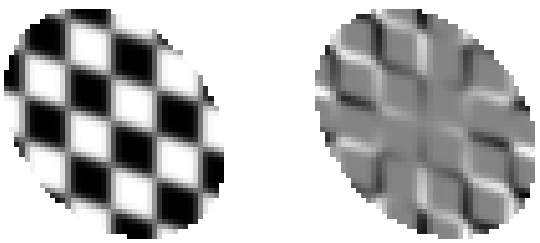

Figure 1: A texture mapped planar patch and I_ψ, its partial derivative with respect to yaw (rotating around Z). Gray is zero.

In this section we elaborate on the calculation of the measurement Jacobian as in (7). 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:

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

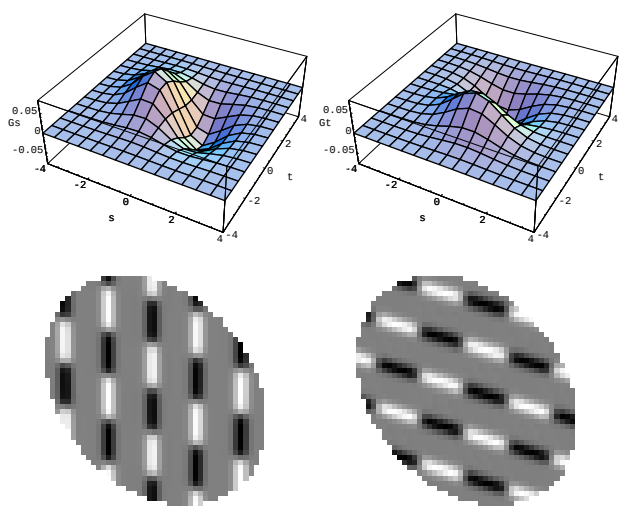

Figure 2: Top: Gradient kernels. Bottom: Gradient images.

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. Note that the calculation of gradients is an important and often neglected element of all gradient based motion estimation algorithms, and the performance of an implementation often depends on it [8, 2].

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

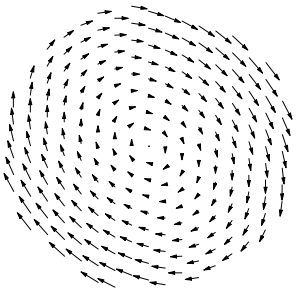

Figure 3: Vector field induced by a change in yaw ψ.

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. Since a patch has six degrees of freedoms, there are 6 vector fields, yielding 6 different Jacobian images. These vector fields correspond to the motion *in texture space* of the preimages 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 computational 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

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

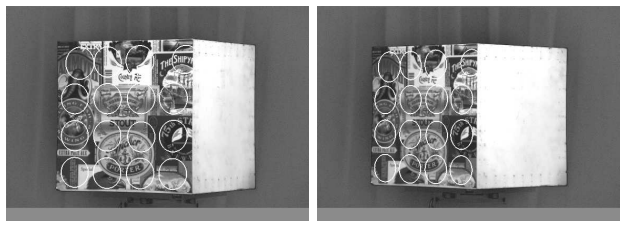

Figure 4: 16 'stickers' tracking the textured face of a cube.

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 [6].

6.2 A Projective Tracker

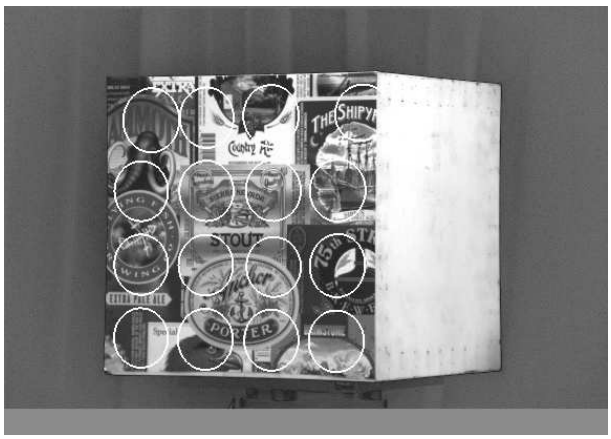

Figure 5: Reading off a moving object: the projective tracker.

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, an example 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. For example, Figure 5 shows...

6.3 Fast Mosaicking

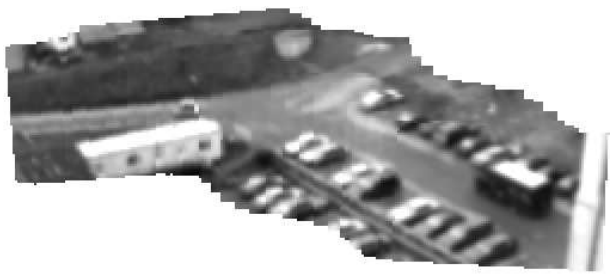

Figure 6: Mosaic.

Figure 7: Independently moving object detection.

A final example illustrates the use of our technique as an alternative approach to mosaicking. The specific application 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 4.3. Figure 7 shows how this mosaic can be 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 in favor of 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 Conclusion and Future Work

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.

Although we have only presented examples using planar surfaces, there are no important difficulties in extending this approach to non-planar surface models. Future work will address the investigation of *arbitrary* surface representations, and how their parameters could be estimated in parallel from the image data. In addition, in the 3D tracking application, we would like to investigate the simultaneous recovery of camera parameters for uncalibrated scenarios.

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. L. Barron, D. J. Fleet, and S. S. Beuachemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

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

[4] 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.

[5] F. Dellaert, D. Pomerleau, and C. Thorpe. Model-based car tracking integrated with a road-follower. In *Proceedings, IEEE Conference on Robotics and Automation (ICRA 98)*, Leuven, Belgium, In Press.

[6] F. Dellaert, C. Thorpe, and S. Thrun. Super-resolved tracking of planar surface patches. In *submitted to IROS 98*, 1998.

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

[8] B. k P Horn and E. J. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 1(2):51–76, 1988.

[9] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 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.