Modelling and Generating Complex Motion Blur for Real-time Tracking

Christopher Mei and Ian Reid
Department of Engineering Science - University of Oxford
\{cmei, ian\}@robots.ox.ac.uk

Objective
Track planar regions in real-time in presence of complex motion blur.

Contributions:
- hierarchical approach for modelling homography transformations,
- efficient parameterisation of motion blur through the use of Lie algebras,
- an algorithm for generating complex blur with implementation on a GPU.

Tracking through gradient descent

\[ \nabla^\lambda \sum_{p \in \mathbb{R}^2} \left( ||I_p - I_{\mathcal{H}}(\mathcal{H}p) - \mathcal{I}^*||^2 \right) \]

Objective

Blur kernels and Lie-group parameterisation

The scale factor of \( \mathbf{H} \) can be chosen such that \( \mathbf{H} \in \mathfrak{sl}(3) \) (Special Linear Group, i.e. \( \det(\mathbf{H}) = 1 \)).

\( \mathfrak{sl}(3) \) is a Lie group so \( \mathbf{H} \) can be represented locally in exponential form using a basis \( \mathbf{G}_1, \ldots, \mathbf{G}_8 \) of \( \mathfrak{so}(3) \):

\[
\mathbf{H}(\mathbf{x}) = \exp(\mathcal{H}(\mathbf{x})), \quad \mathcal{H}(\mathbf{x}) = \sum_{i=1}^8 \mathbf{g}_i^n
\]

A Hierarchy of transformations

Choosing specific \( \mathbf{G}_8 \) provides a hierarchy of transformations:

- \( \mathbf{G}_1, \mathbf{G}_2 \rightarrow \) translation
- \( \mathbf{G}_1, \ldots, \mathbf{G}_8 \rightarrow \) affine transformation

A hierarchy of transformations greatly reduces the computational time for tracking (the complexity is quadratic in the number of parameters) and provides better robustness to noise.

Parameterising blur

In [Jin et al. ICCV 05], a Gaussian kernel is proposed for tracking in presence of translational blur:

\[
\mathcal{I}(\mathbf{p}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\tau^2} \mathcal{I}(\mathbf{p} - \mathbf{v}t)dt
\]

We propose the following homography-based kernel:

\[
\mathcal{I}_{\mathcal{H}}(\mathbf{p}) = \int_{0}^{\infty} \mathcal{I}(e^{-\tau\mathcal{H}\mathbf{p}})dt
\]

Assumption: the flow field generating the blur is constant w.r.t. time. Complex motion such as combinations of rotation and changes in scale can be modelled.

Different tracking cost functions

- 8 blur parameters (+8 homography parameters)
- 1 blur parameter, the exposure time, denoted \( \lambda \):

\[
\mathcal{I}_{\mathcal{H}}(\lambda) = \int_{0}^{\infty} \mathcal{I}(e^{-\tau\lambda\mathcal{H}\mathbf{p}})dt
\]

Generating motion blur

A recursive formulation can be obtained using the properties of the exponential map (the exponentials can be pre-calculated):

\[
\begin{align*}
\mathcal{P}_0 &= \mathcal{P} \\
\mathcal{P}_i &= e^{-\lambda \mathcal{H}} \mathcal{P}_{i-1}
\end{align*}
\]

Calculating blur is highly parallelisable and can be efficiently implemented on a GPU. Applying a blur of 10 deg. to a patch of size 100x100 takes:

- 12 ms on a 2.4 GHz CPU (using a single core)
- 0.44 ms on a GPU (GeForce 8800 GTX using CUDA)

Simulation

- E0 standard homography-based tracking algorithm without taking into account motion blur,
- E1 only the blur magnitude is estimated
- E2 translational motion blur, \( \mathcal{I} \) indicates a directional kernel and \( \mathcal{I} \) refers to the Gaussian model,
- E8 8 independent blur parameters

Results

Handheld sequence with blurred reference image (middle row) and warped current template (bottom row)

DVD sequence with blurred reference image (middle row) and warped current template (bottom row)

Conclusion

- tracking in presence of motion blur can be done efficiently by only estimating the exposure time,
- the proposed formulation can implemented efficiently on a GPU,
- Lie groups provide a natural hierarchy of transformations leading to better computational complexity and robustness.