Computer Vision on Rolling Shutter Cameras

PART III: Rectification and Stabilisation

Per-Erik Forssén, Erik Ringaby, Johan Hedborg

Computer Vision Laboratory
Dept. of Electrical Engineering
Linköping University

CVPR 2012
Providence, Rhode Island
June 16-21, 2012

Linköping University
INSTITUTE OF TECHNOLOGY

Tutorial overview

1:30-2:00pm	Introduction	Per-Erik
2:00-2:15pm	Rolling Shutter Geometry	Per-Erik
2:15-3:00pm	**Rectification and Stabilisation**	**Erik**
3:00-3:30pm	Break	
3:30-3:45pm	Rolling Shutter and the Kinect	Erik
3:45-4:30pm	Structure from Motion	Johan

Distortion examples

Camera pan

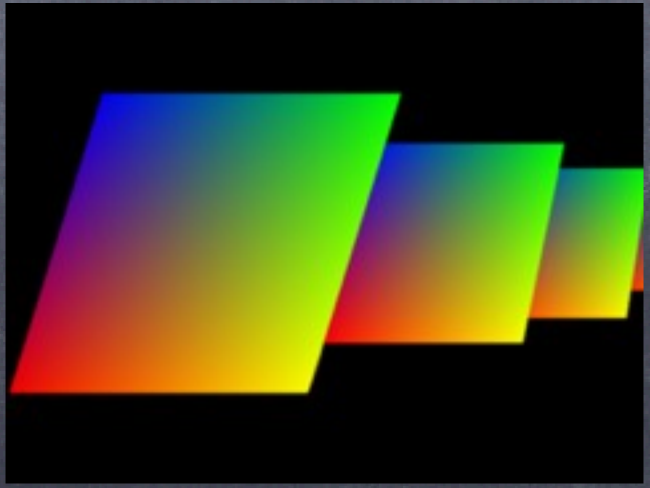
Camera rotation

Fast moving object

Camera vibration (wobble)

The full rectification problem

- A full rectification model requires motion segmentation as fast moving objects must be treated differently from camera motion
- Multiple frames are needed to be able to handle occlusions
- Camera translation gives distortions which depend on scene depth (parallax effects)



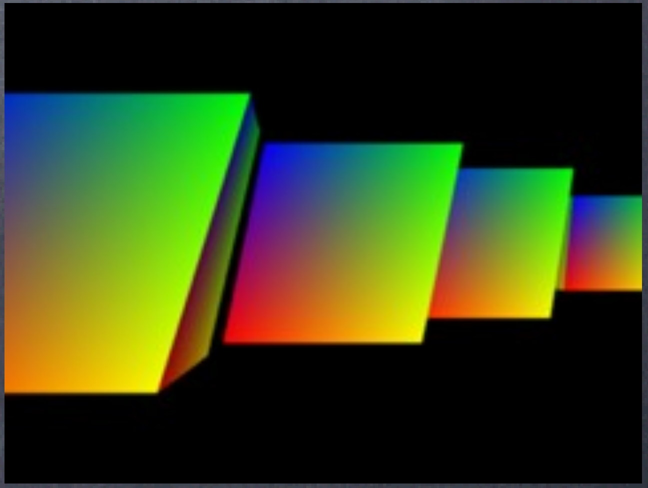


Image plane models

- Model image deformation as caused by a globally constant translational motion across the image [chang05, nicklin07, chun08]

- Improvement by giving each row a different motion, that is found by interpolating between constant global inter-frame motions using a Bézier curve [liang08]

Image plane methods

- Model distortions as a global affine deformation parametrised by the scan-line index [Cho et al. TCE'07]
- Blend linearly between many translational models across a frame to deal with wobble [Baker et al. CVPR'10]
- Mixture model of homographies, where some parts are constant across the frame [Kim et al. CSVT'11] and [Grundmann et al. ICCP'12]

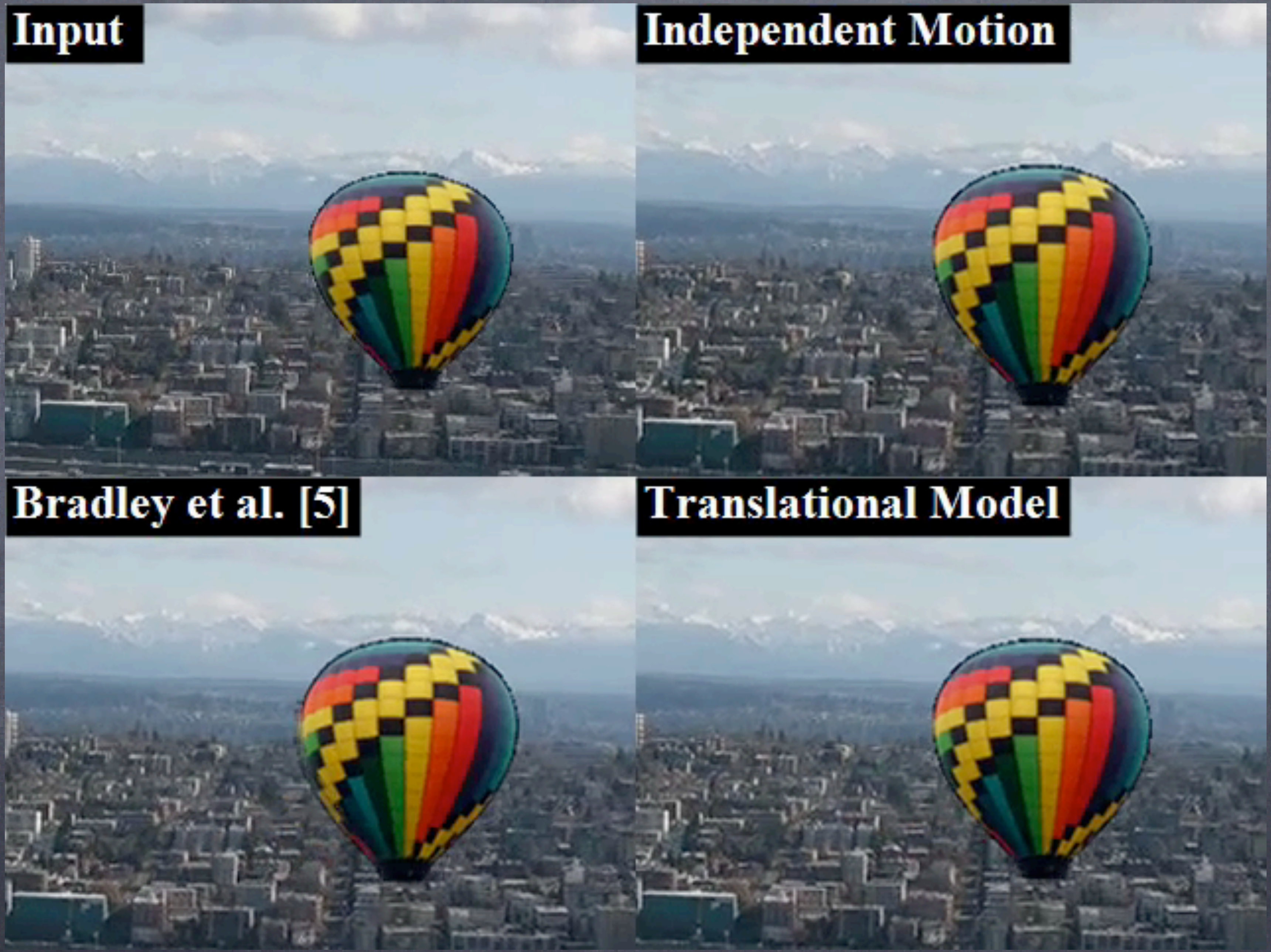
Wobble correction

- [Baker et al. CVPR'10] have experimented with dense flow and many motion models across a frame e.g. 30.
- Preferred translational image plane models.

Independent Motions

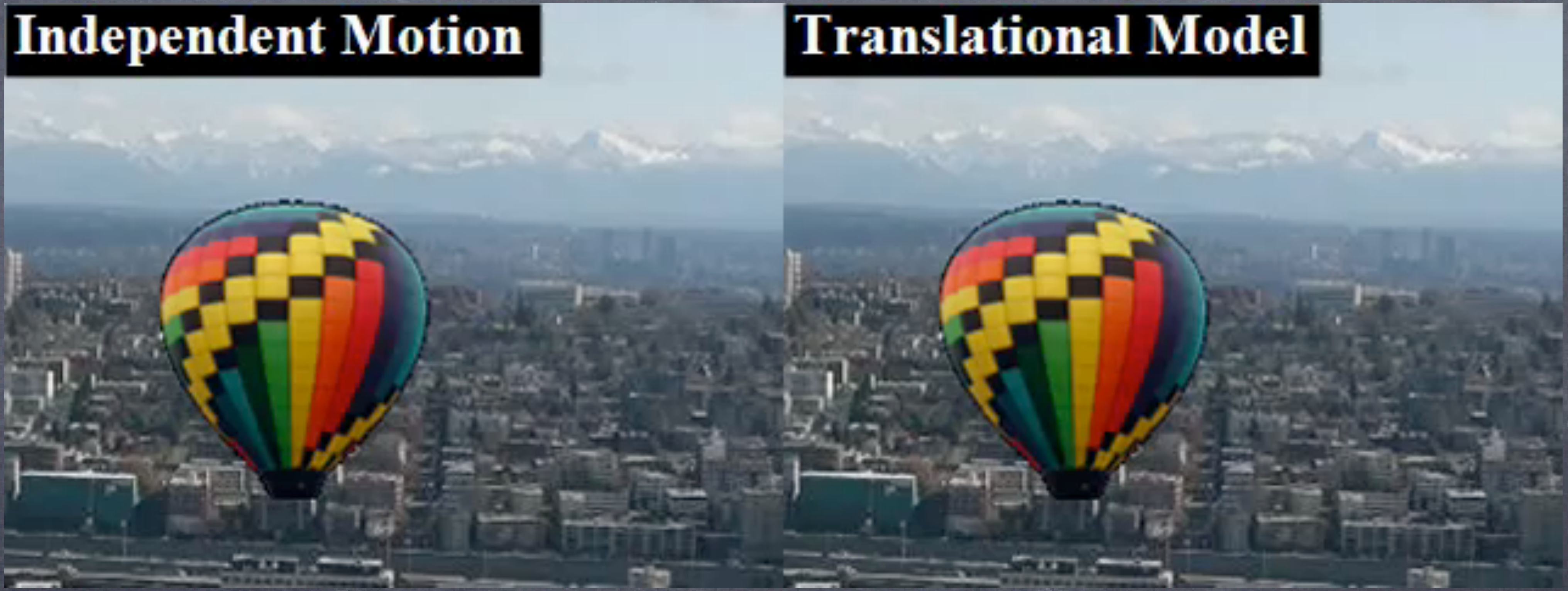
- [Baker et al. CVPR'10] have experimented with separate reconstructions for multiple indepentent motions.
- Clear improvement compared to input videos, but small difference compared to frame-global rectification.
- Distortions at object boundaries visible in still frames. These are however difficult to see in the video.

Independent Motions



Independent Motions

- Work on different rectifications for multiple motions by [Baker et al. CVPR'10]

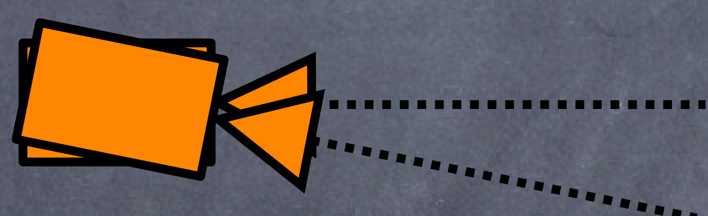


Independent Motions

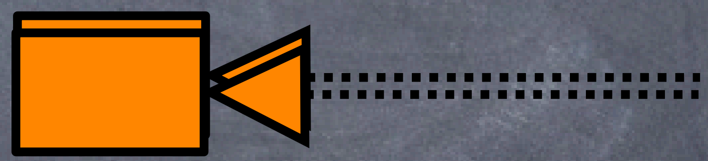
- Work on different rectifications for multiple motions by [Baker et al. CVPR'10]
- Relies accurate dense flow [Black and Anandan CVIU'96]
- Independent motion model is 50x more expensive than their translational model. (102.4 sec instead of 2.1 sec/frame at 320x240)

Rotational model

- Most rectification models assume that the distortion takes place in the image plane
- We know that for hand held motion, the dominant source is 3D rotations.



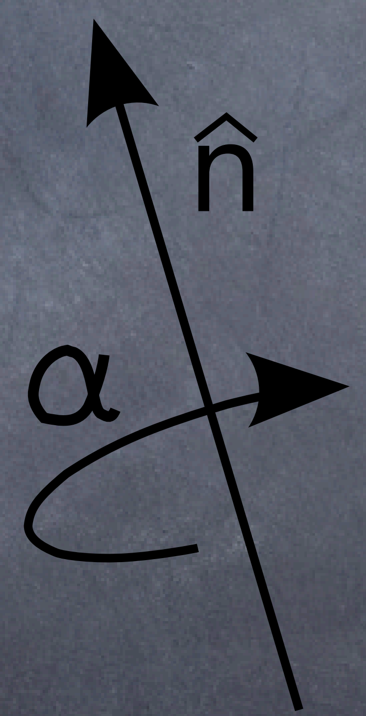
camera rotation



camera translation

Rotation representation

- Euler's rotation theorem states that any 3D rotation may be expressed as a three element rotation axis, and a rotation angle about that axis.



Rotation representation

- The rotation axis $\mathbf{n}$ and angle ϕ are related to a rotation matrix according to the matrix exponent and matrix logarithm:

$$\mathbf{R} = \text{expm}(\mathbf{n}) = \mathbf{I} + [\hat{\mathbf{n}}]_x \sin\phi + [\hat{\mathbf{n}}]_x^2 (1 - \cos\phi)$$

$$\mathbf{n} = \text{logm}(\mathbf{R}) = \phi\hat{\mathbf{n}}\,, \quad \text{where} \quad \begin{cases} \tilde{\mathbf{n}} = \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \\ \phi = \tan^{-1}(||\tilde{\mathbf{n}}||, \text{tr}\mathbf{R} - 1) \\ \hat{\mathbf{n}} = \tilde{\mathbf{n}}/||\tilde{\mathbf{n}}||\,. \end{cases}$$

SO(3) and SE(3)

- SO(3) is the group of 3D rotations (3dof)

$$\mathrm{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \right\}$$

SO(3) and SE(3)

- SO(3) is the group of 3D rotations (3dof)

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\}$$

- SE(3) is the group of Euclidean rigid body transformations (3D rotation+3Dtranslation) (6dof)

$$\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$$

- For SE(3) we can similarly define an exponential map and a log map.

SO(3) and SE(3)

- An element $\mathbf{G} \in \mathrm{SE}(3)$ has the matrix form

$$\mathbf{G} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{R} \in \mathrm{SO}(3)\,, \quad \mathbf{t} \in \mathbb{R}^3$$

- It is the exponential of a twist

$$\mathbf{G} = \exp(\hat{\xi}\theta) \qquad \hat{\xi} = \begin{bmatrix} \mathrm{logm}(\mathbf{R}) & \mathbf{v} \\ 0 & 0 \end{bmatrix} \quad \theta \in \mathbb{R}$$

SO(3) and SE(3)

- An element $\mathbf{G} \in \mathrm{SE}(3)$ has the matrix form

$$\mathbf{G} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{R} \in \mathrm{SO}(3), \quad \mathbf{t} \in \mathbb{R}^3$$

- It is the exponential of a twist

$$\mathbf{G} = \exp(\hat{\xi}\theta) \qquad \hat{\xi} = \begin{bmatrix} \mathrm{logm}(\mathbf{R}) & \mathbf{v} \\ 0 & 0 \end{bmatrix} \qquad \theta \in \mathbb{R}$$

- One could do smoothing and interpolation of rigid body motions using the geodesic distance on SE(3) (via the log map). However...

SO(3) and SE(3)

- It turns out that physically meaningful motions do not follow geodesics in SE(3). Rather (if no external force):
- The centre of mass moves linearly
- Rotation happens about the centre of mass
- Thus we should represent R(t) in object centered coordinates, and interpolate R(t) and t(t) separately.

SO(3) and SE(3)

- A very good treatment of SO(3) and SE(3) can be found in the book:
  Murray et al. A Mathematical Introduction to Robotic Manipulation, CRC Press. 1994

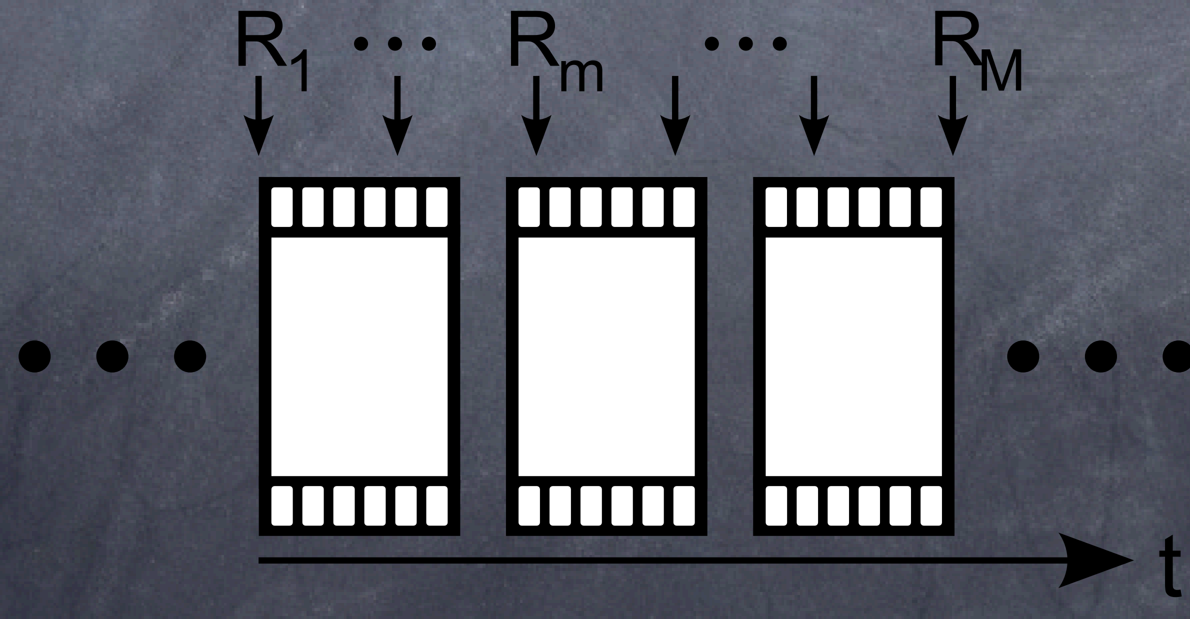
- http://www.cds.caltech.edu/~murray/mlswiki/

Rotation representation

- SLERP (Spherical Linear intERPolation) is used to interpolate rotations

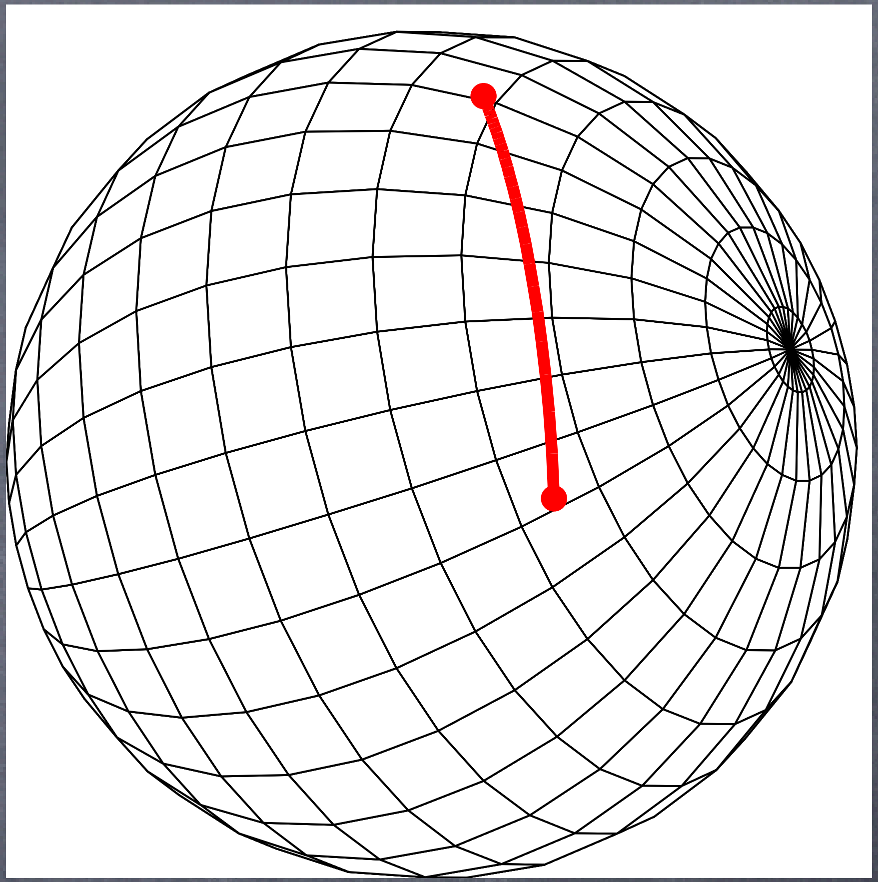
$$\mathbf{n}_{\text{diff}} = \text{logm}\left(\text{expm}(-\mathbf{n}_1)\text{expm}(\mathbf{n}_2)\right)$$

$$\mathbf{R}_{\text{interp}} = \text{expm}(\mathbf{n}_1)\text{expm}(\tau\mathbf{n}_{\text{diff}})$$



SLERP

- The SLERP construction is a geodesic on SO(3), i.e. a walk along the shortest path, on the manifold, between the two rotations.
- If we use unit quaternions, the geodesic lies on a 4D sphere.

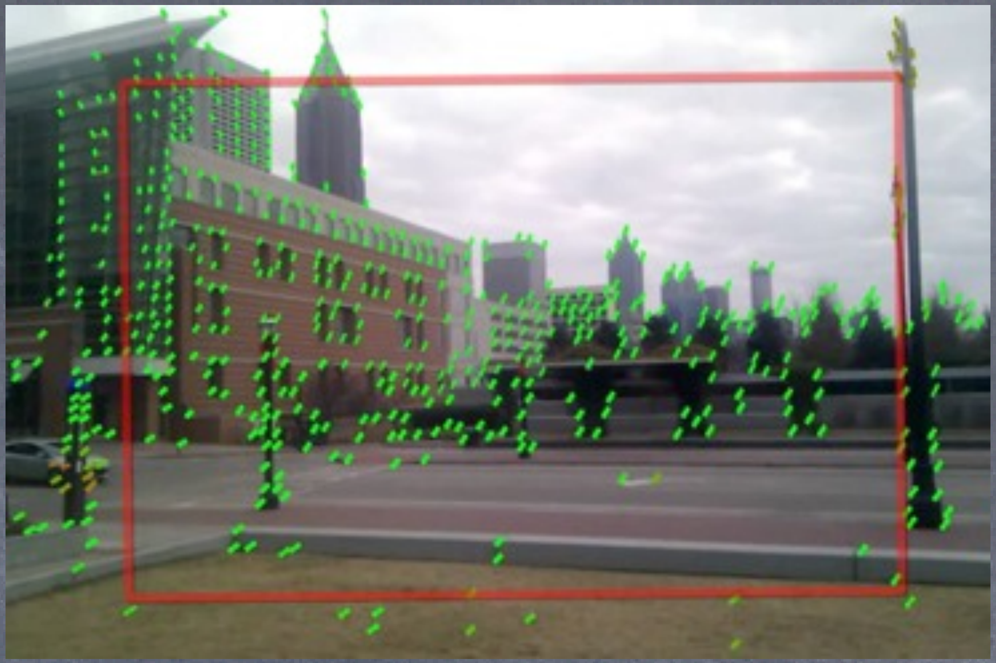


Geodesic on the sphere

Interest point selection

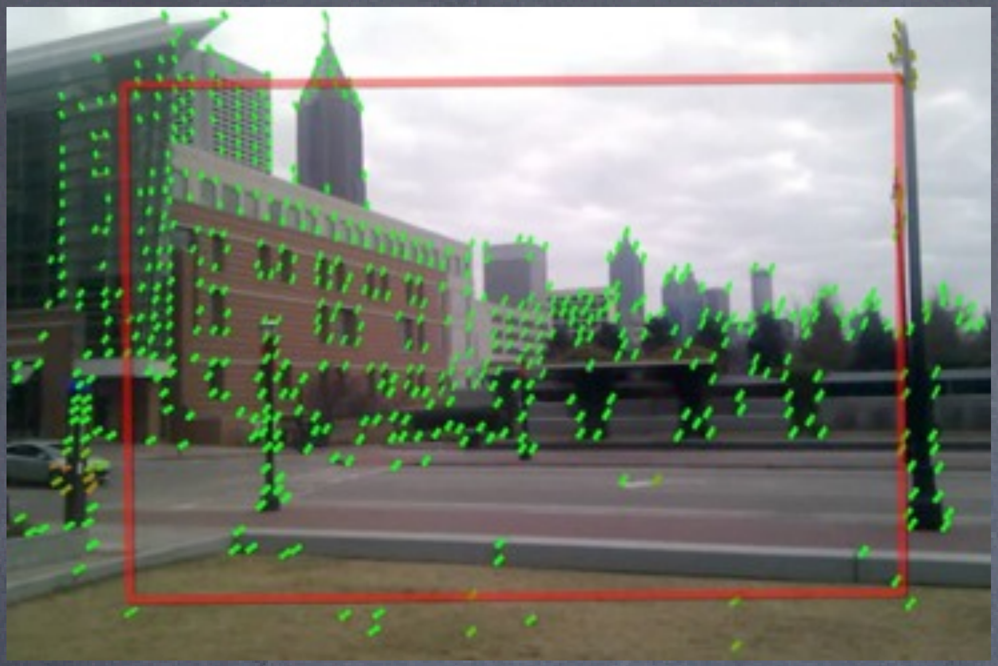
- For sparse optical flow, the state of the art in global shutter cameras is to use either of:
  - Harris points [Harris&Stephens 86]
  - Good features to track [Shi&Tomasi CVPR'94]
  - FAST points [Rosten&Drummond ECCV'06]
- All of these compute a constrast sensitive measure of "cornerness", and select the N strongest such points in the image.

Interest point selection



- Motion estimation for an RS camera benefits from a uniform distribution of interest points.
- Thresholding on corner strength is slightly problematic on RS cameras, as low contrast regions (e.g. sky and road above) will get very few points.

Interest point selection



uniform threshold

locally adapted threshold
by Grundmann et al.

- Adaptive thesholding of good features to track, by [Grundmann et al. ICCP'12]
- Divide image into blocks, and require similar number of interest points in each block.

3D solution

- Assume the camera is moving in a **static scene**
- Estimate **3D camera motion** from a sparse optical flow
- Use 3D motion to **rectify each row separately**
- Better models the cause of the distortions

[Forssén, Ringaby CVPR'10]
[Ringaby, Forssén IJCV'12]

Algorithm overview

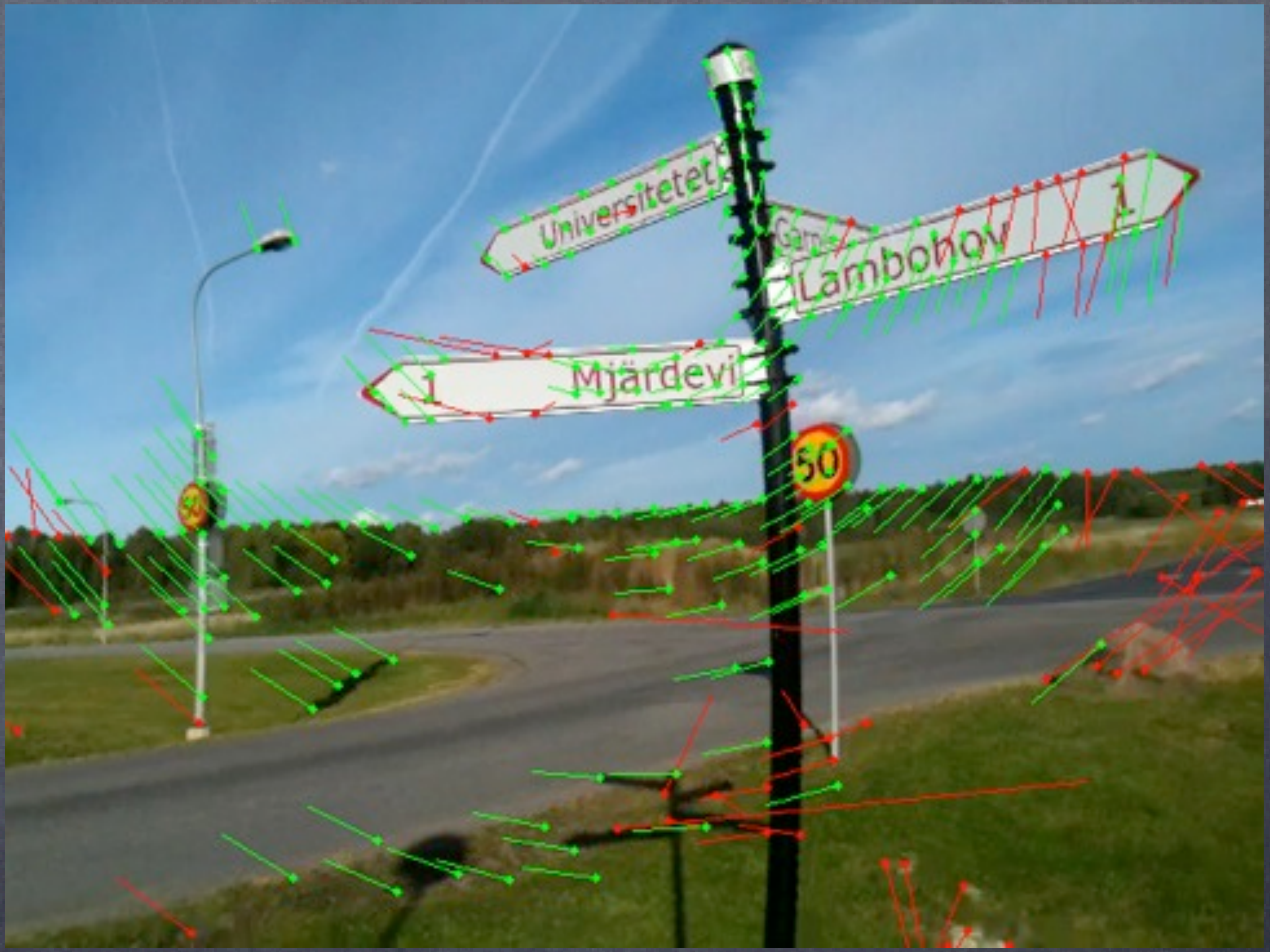
1. Find inter-frame **correspondences** using point detection and tracking
2. Define reprojection error cost function using **scene rigidity constraints**
3. Solve for **3D camera motion** over short frame intervals
4. **Rectify each row separately** e.g. using camera motion relative to middle row

Point correspondences

- Good features to track
- KLT-tracker
- Track short frame interval, 2-4 frames, then detect points again

Cross-checking

- Tracking sometimes fail
- Cross-checking step in order to minimize incorrect point matches



- Green: accepted
- Red: rejected

Camera models

- Previously we introduced the pinhole camera model
- Also, for a moving rolling-shutter camera, the external parameters will be time dependent

$$\mathbf{x} \sim \mathbf{K}\mathbf{R}(x_2)^T \left[\mathbf{I} | -\mathbf{d}(x_2)\right] \mathbf{X}$$

Camera models

- We have seen that camera rotation is the major cause of distortions
- Simplify the model to only take rotation into account

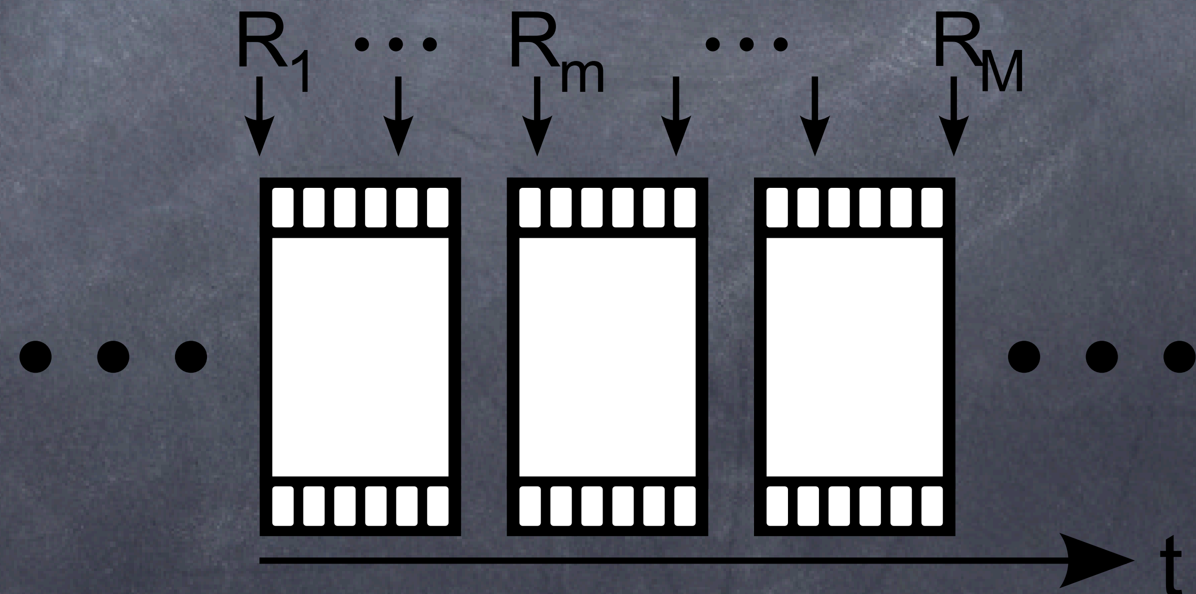
$$\mathbf{x} \sim \mathbf{K}\mathbf{R}(x_2)\mathbf{X}$$

- Model valid if the distance to scene objects is large compared to the baseline
- We use short frame interval → small translation

Camera motion estimation

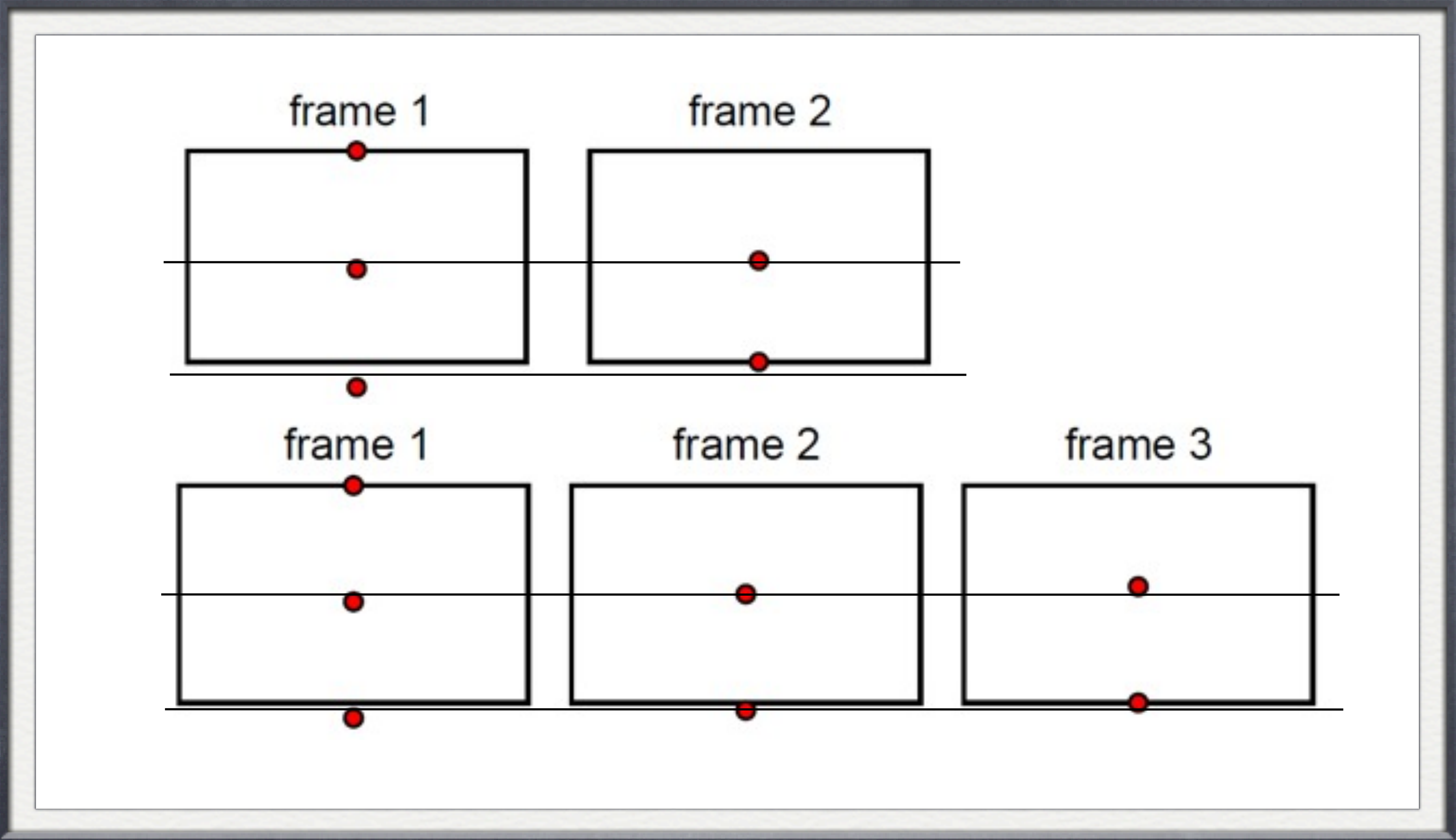
- Since the image rows are exposed at different times, one would like to have the camera pose for each of them
  → high number of parameters to be estimated

- Instead, model the motion as a sequence of "key-rotations"



Knot positions

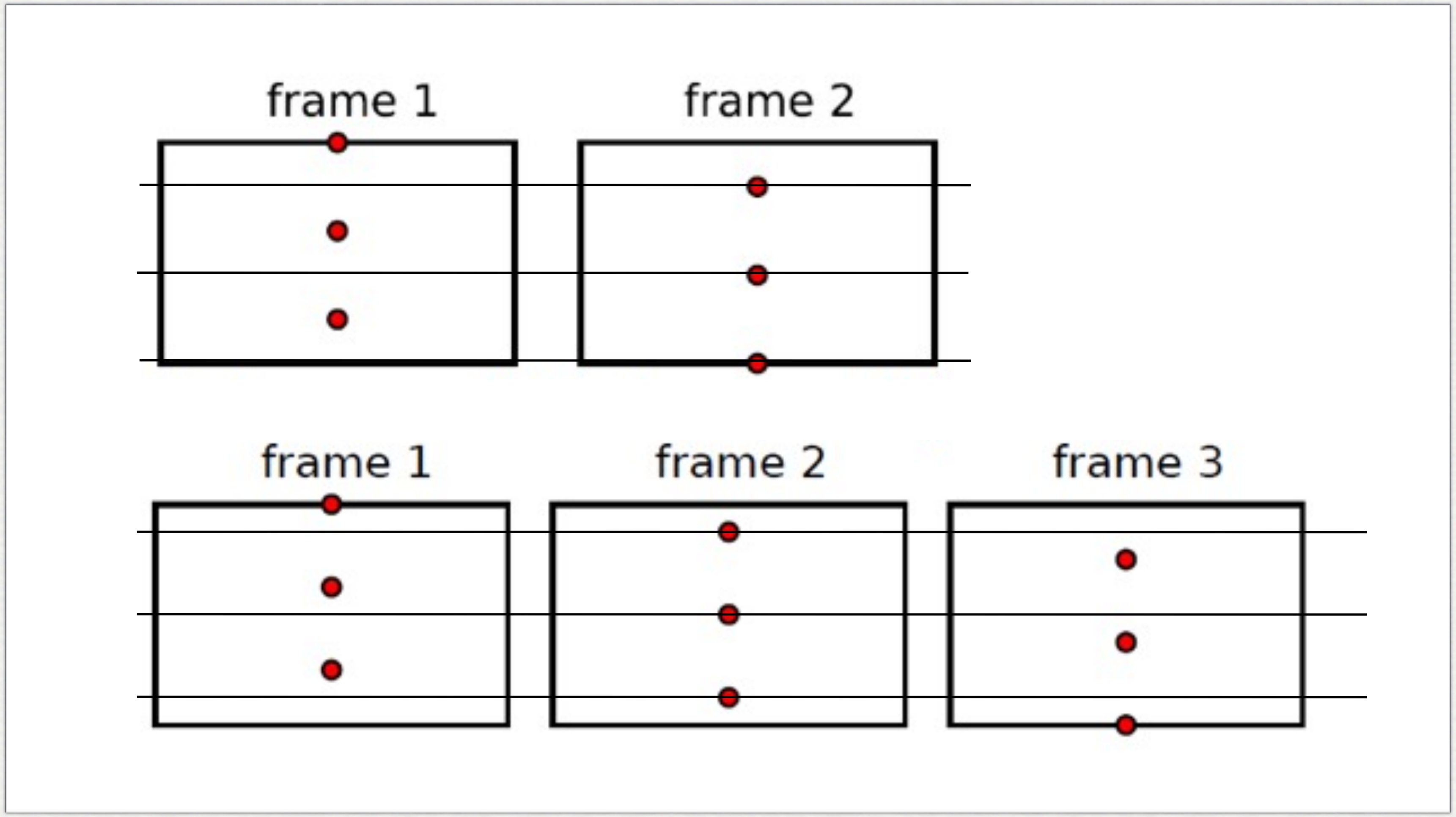
- Examples with configurations with less good results:



[Ringaby IJCV'12]

Knot positions

- Examples with configurations with good results:



[Ringaby IJCV'12]

Camera motion estimation

- A 3D point **X** will project into two consecutive frames as

  $\mathbf{x} \sim \mathbf{KR}(x_2)\mathbf{X}$ and $\mathbf{y} \sim \mathbf{KR}(y_2)\mathbf{X}$

  where the time parameter t has been exchanged to the point's corresponding row number N

↔

Camera motion estimation

- A 3D point $\mathbf{X}$ will project into two consecutive frames as

$$\mathbf{x} \sim \mathbf{KR}(x_2)\mathbf{X} \quad \text{and} \quad \mathbf{y} \sim \mathbf{KR}(y_2)\mathbf{X}$$

- This gives us a relationship between the two corresponding points $\mathbf{x}$ and $\mathbf{y}$

$$\mathbf{x} = \mathbf{KR}(x_2)\mathbf{R}^T(y_2)\mathbf{K}^{-1}\mathbf{y} = \mathbf{Hy}$$

Camera motion estimation

- Spline parameters are solved for using **iterative optimisation** on the cost function

$$J = \sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{H}\mathbf{y}_k)^2 + d(\mathbf{y}_k, \mathbf{H}^{-1}\mathbf{x}_k)^2,$$

$$\text{where} \quad d(\mathbf{x}, \mathbf{y})^2 = (x_1/x_3 - y_1/y_3)^2 + (x_2/x_3 - y_2/y_3)^2$$

- K is the number of point correspondences

$$\mathbf{x} = \mathbf{K}\mathbf{R}(x_2)\mathbf{R}^T(y_2)\mathbf{K}^{-1}\mathbf{y} = \mathbf{H}\mathbf{y}$$

Camera motion estimation

Estimated camera motion

RGB represents the 3 rotation parameters

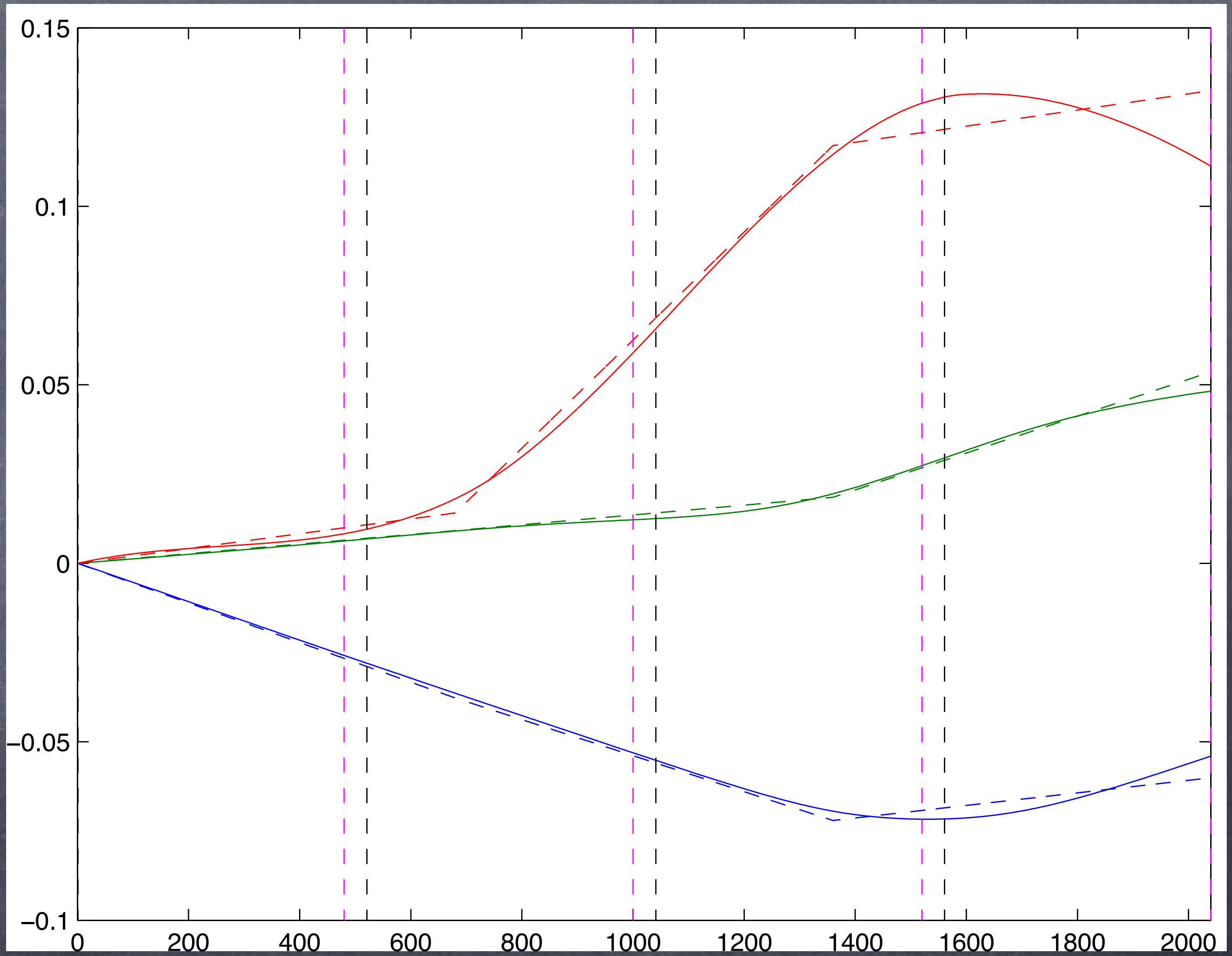


Image rectification

- When the camera motion has been estimated, i.e. the "key-rotation", all the image rows can be transformed to a common coordinate system

$$\mathbf{x}' = \mathbf{K}\mathbf{R}_{\mathrm{ref}}\mathbf{R}^T(x_2)\mathbf{K}^{-1}\mathbf{x}$$

- x′ is the rectified position of x

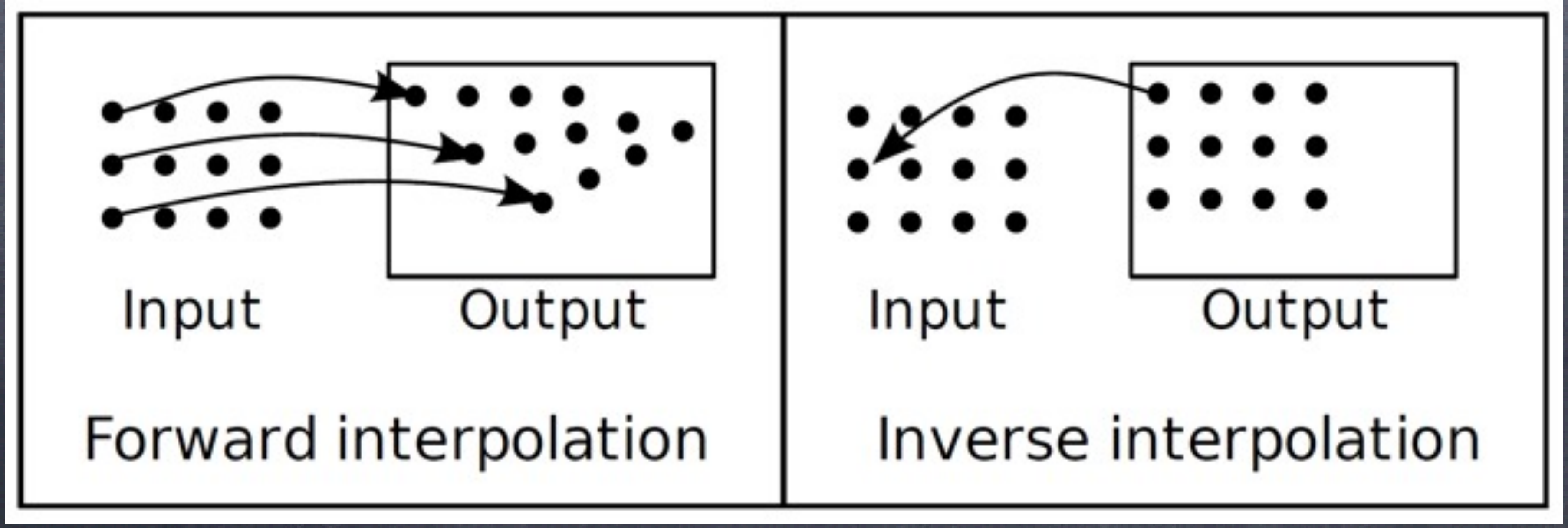
- $\mathbf{R}_{\mathrm{ref}}$ is a reference rotation corresponding to a certain row, e.g. the middle one

Image rectification

- Rectification transformation

$$\mathbf{x}' = \mathbf{K}\mathbf{R}_{\mathrm{ref}}\mathbf{R}^T(x_2)\mathbf{K}^{-1}\mathbf{x}$$

- This is a forward mapping of the points

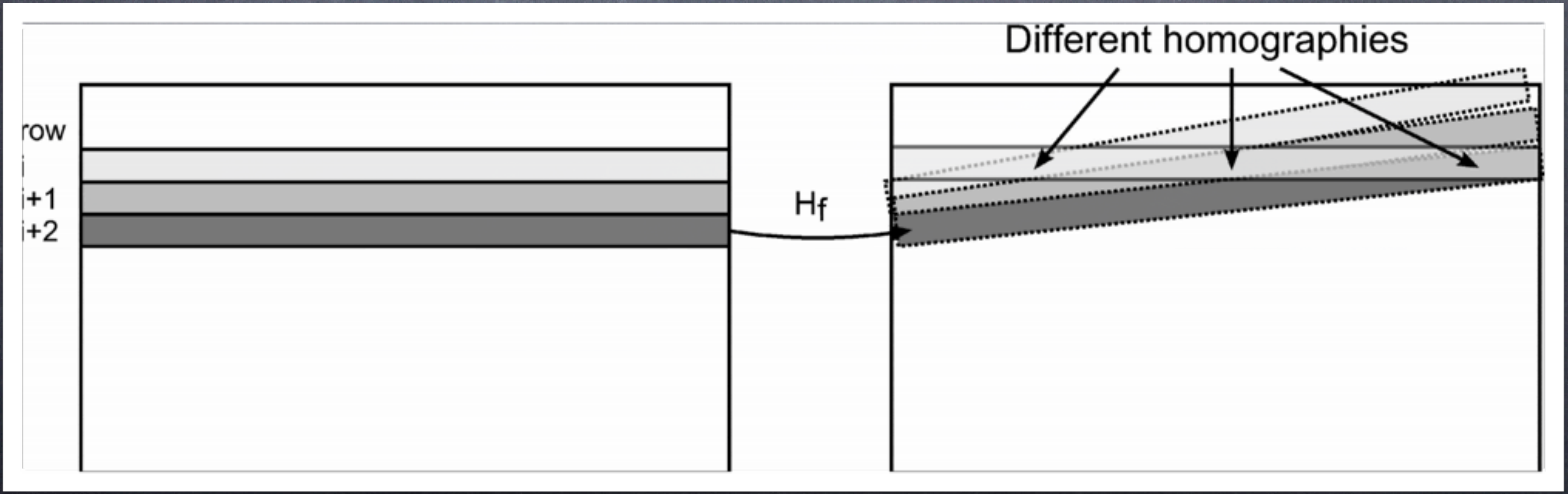


Forward vs. Inverse interp.

- When to use forward interpolation?
- Sometimes inverse mapping not available
- Here only an approximate inverse mapping exists

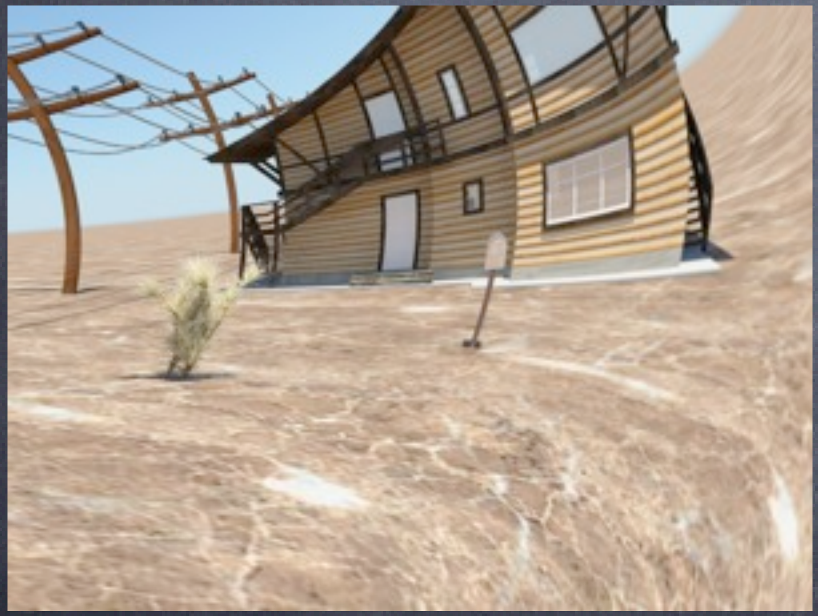
Forward vs. Inverse interp.

- Neighbouring pixels within a row in the rectified image do not necessarily have the same homography

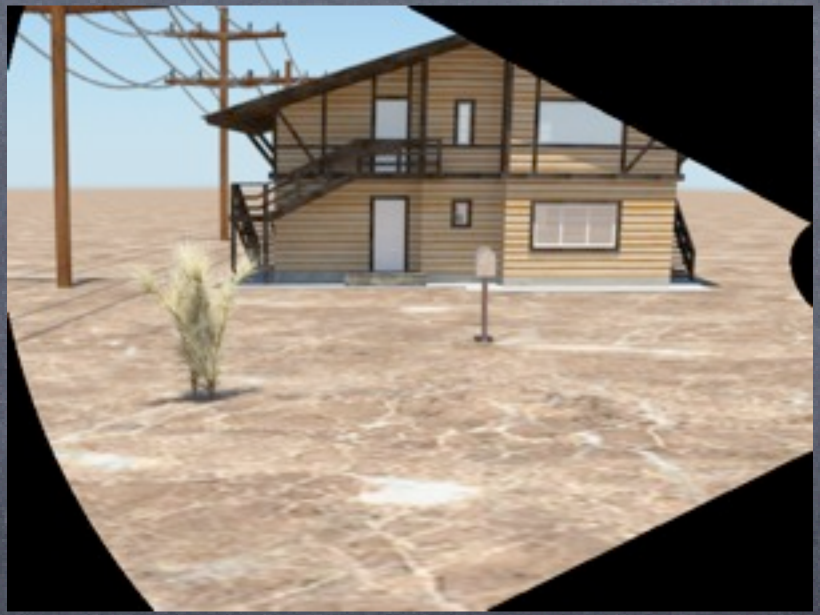


Interpolation comparison

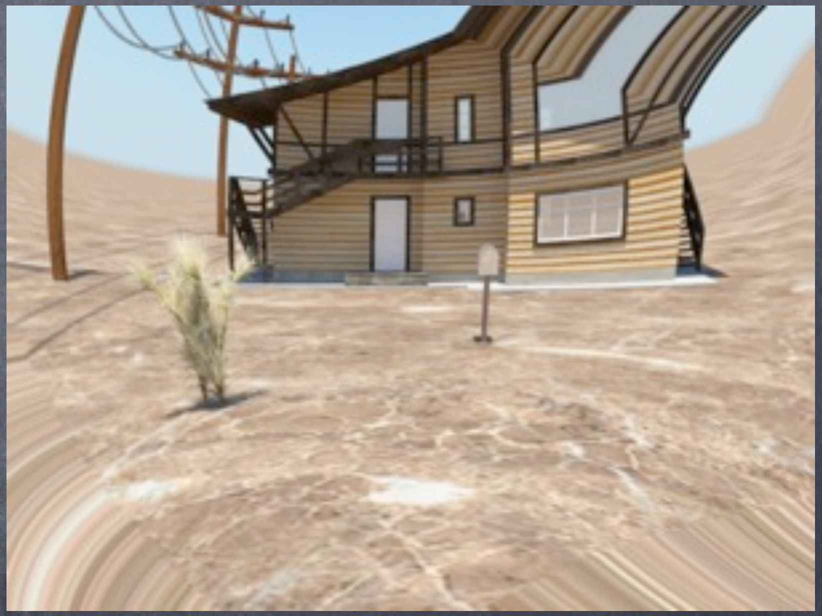
- Example with fast motion:



RS frame



Forward interpolation



Inverse interpolation

Image rectification

Image rectification

Synthetic dataset, Maya

- High-end 3D computer graphics and 3D modeling software
- Possible to create ground-truth images
- MEL (Maya Embedded Language)
- Easy to automate, extract camera parameters etc.

Synthetic dataset

RS frame Ground-truth frame Mask

Available at: http://www.cvl.isy.liu.se/research/rs-dataset

Motion from sensors

- Instead of estimating the camera motion from video data (optical flow), additional sensors can be used
- Many smartphones have both accelerometers and gyroscopes
- Using sensor fusion techniques, the camera / device orientation can be estimated [Törnqvist 08]
  + Faster than doing non-linear optimisation
  + Not sensitive to dynamic scene when est. camera motion
  - Can not compensate for moving objects
  - Bias
  - Need to synchronize sensors to video

[Hanning et.al. IWMV'11]

Stabilisation

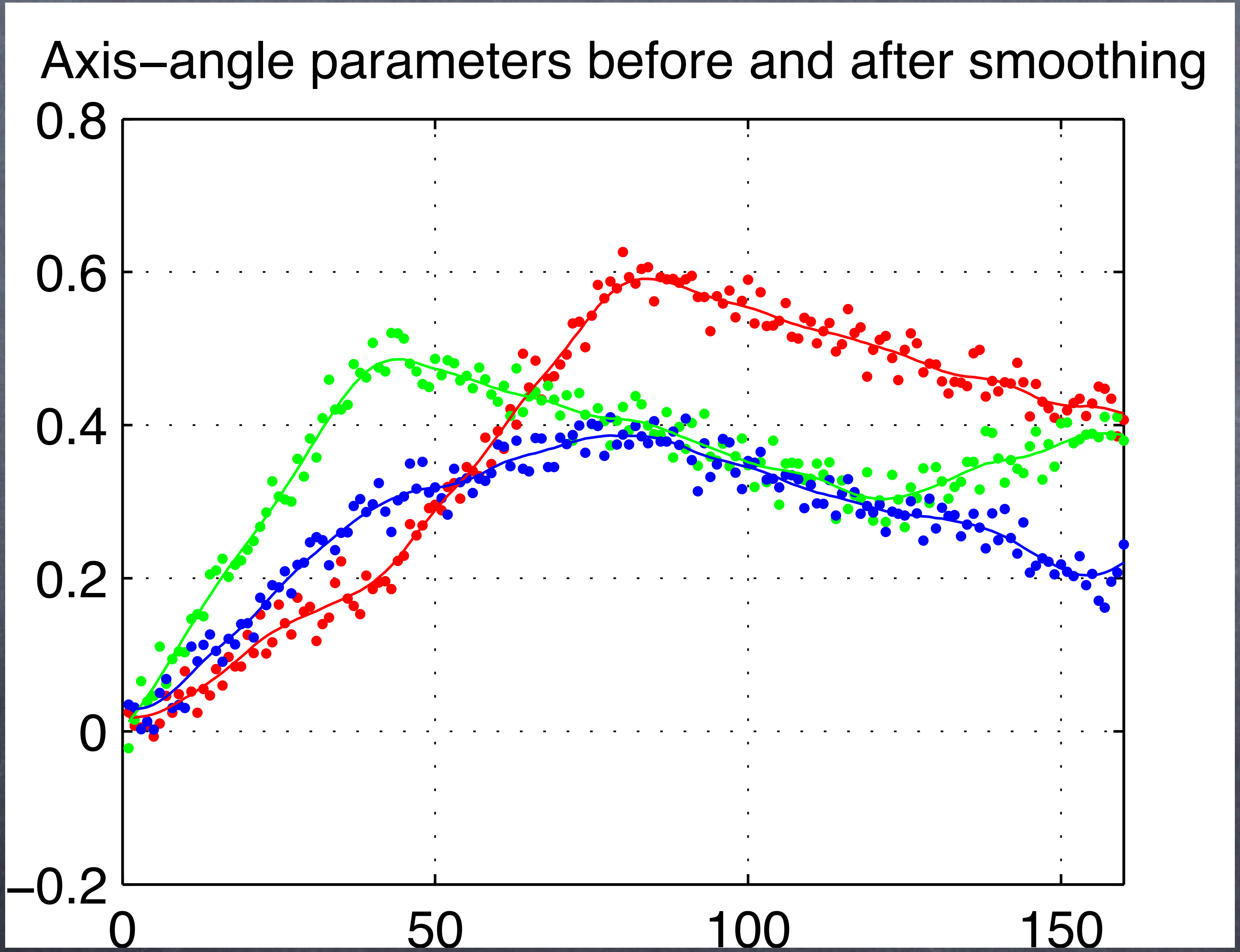
- Instead of using R_{ref} directly in the rectification

$$\mathbf{x}' = \mathbf{K}\mathbf{R}_{\mathrm{ref}}\mathbf{R}^T(x_2)\mathbf{K}^{-1}\mathbf{x}$$

one can rectify to a (temporal) smoothed version for efficient video stabilisation

Rotation Smoothing

- Problem: We have a sequence of noisy rotations, and want a smoother trajectory.



Rotation Smoothing

- For each temporal window, this can be solved by ML as:

$$\mathbf{R}^* = \arg \min_{\mathbf{R} \in SO(3)} \sum_k d_{\text{geo}}(\mathbf{R}, \mathbf{R}_k)^2$$

- Where

$$d_{\text{geo}}(\mathbf{R}_1, \mathbf{R}_2)^2 = \frac{1}{2} ||\text{logm}(\mathbf{R}_1^T \mathbf{R}_2)||^2_{\text{fro}}$$

Rotation Smoothing

- For each temporal window, this can be solved by ML as:

$$\mathbf{R}^* = \arg\min_{\mathbf{R}\in SO(3)} \sum_k d_{\text{geo}}(\mathbf{R}, \mathbf{R}_k)^2$$

- Where

$$d_{\text{geo}}(\mathbf{R}_1, \mathbf{R}_2)^2 = \frac{1}{2}||\text{logm}(\mathbf{R}_1^T\mathbf{R}_2)||_{\text{fro}}^2$$

- Expensive, and maybe too slow :-(
- There are fast and almost as good alternatives :-)

Rotation Smoothing

- For a sequence of unit quaternions

$$\mathbf{q}_k \ , \quad \mathbf{q}_{k+1} \ , \quad \mathbf{q}_{k+2} \ , \quad \cdots$$

$$\mathbf{q}_k = (\cos\frac{\theta_k}{2}, \sin\frac{\theta_k}{2}\hat{\mathbf{n}})$$

- Note that $\mathbf{q}_k$ and $-\mathbf{q}_k$ represent the same rotation (double folding property)
- We need to first ensure that $\mathbf{q}_k \cdot \mathbf{q}_l > 0$
- Now we can simply average them!

Rotation Smoothing

- If we have a sequence of unit quaternions

$$\mathbf{q}_k \ , \quad \mathbf{q}_{k+1} \ , \quad \mathbf{q}_{k+2} \ , \quad \ldots$$

$$\mathbf{q}_k = (\cos\frac{\theta_k}{2}, \sin\frac{\theta_k}{2}\hat{\mathbf{n}})$$

- Apply a temporal convolution, followed by a normalisation to unit length.

$$\tilde{\mathbf{q}}_k = \sum_{l=-2}^{2} w_l \mathbf{q}_{k+l} \ , \qquad \hat{\mathbf{q}}_k = \tilde{\mathbf{q}}_k / \sqrt{\tilde{q}_1^2 + \tilde{q}_2^2 + \tilde{q}_3^2 + \tilde{q}_4^2}$$

Rotation Smoothing

- If we have a sequence of rotation matrices

$$\mathbf{R}_k\,,\quad \mathbf{R}_{k+1}\,,\quad \mathbf{R}_{k+2}\,,\quad \ldots$$

We could apply a temporal convolution, followed by an orthogonalisation.

$$\tilde{\mathbf{R}}_k = \sum_{l=-2}^{2} w_l \mathbf{R}_{k+l}$$

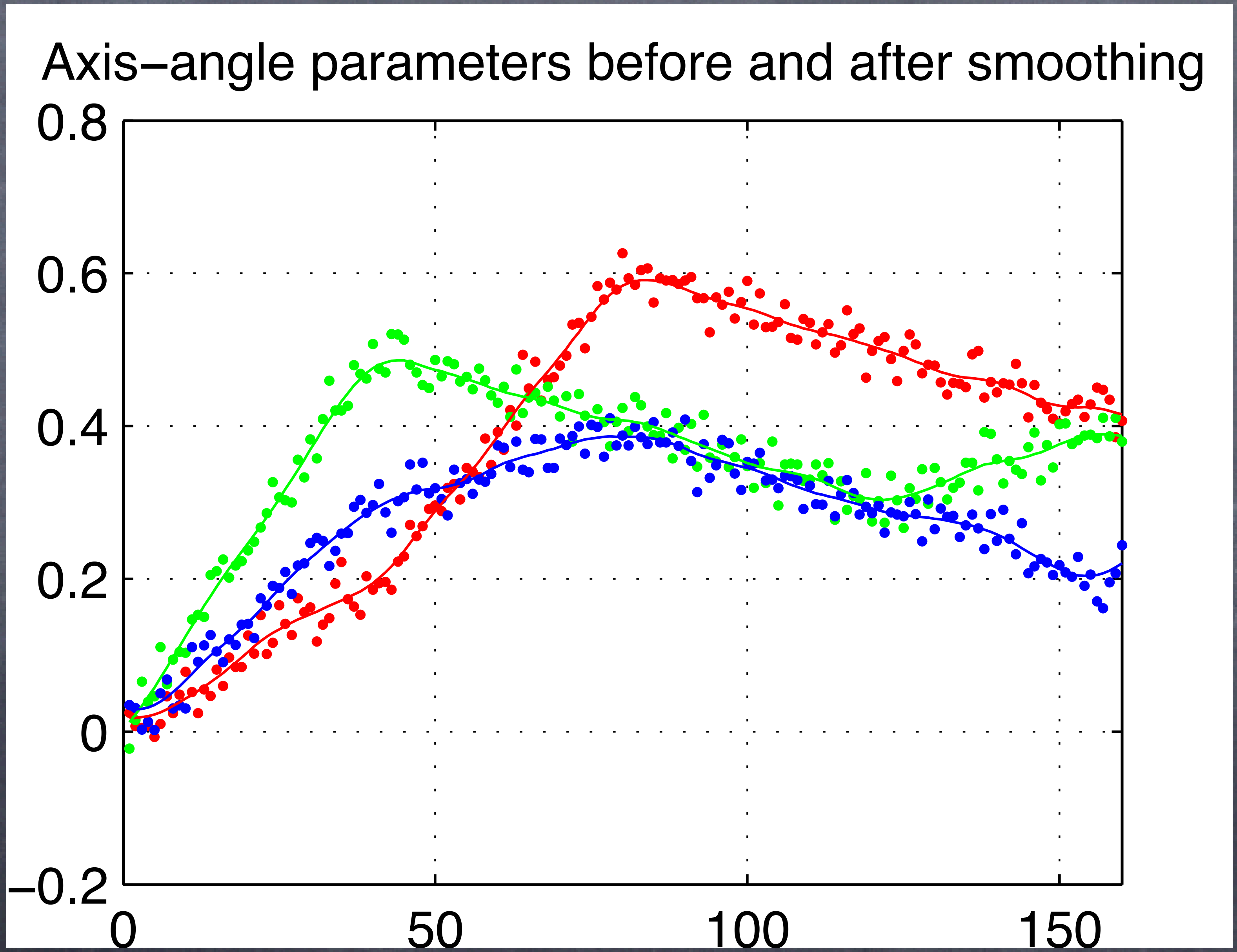
$$\mathbf{UDV}^T = \mathtt{svd}(\mathbf{R}_k)\,,\quad \hat{\mathbf{R}}_k = \mathbf{U}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & |\mathbf{UV}|\end{bmatrix}\mathbf{V}^T$$

Rotation Smoothing

- Both versions can be shown to be 2nd order Taylor approximations of the geodesic distance. [Gramkow IJCV'01]
- Gramkow also compares both against ML.
  Both are very accurate (<5% relative error at 40deg)
- The quaternion variant is slightly closer to the ML solution, and also significantly faster.

Rotation Smoothing

- Result (both methods indistinguishable)



RS correction

Video examples

Video examples

Video examples

Video examples

Video examples

Video examples

Summary

- No current algorithm solves the full rectification problem
- Independent motions and parallax effects are not fully solved
- Camera motion can be super-resolved and corrected for based on optical flow
- Evaluation dataset with ground-truth exists

References

- M. Grundmann, V. Kwatra, D. Castro, I. Essa, "Calibration-Free Rolling Shutter Removal", ICCP'12
- Chang, Liang, Chen, "Analysis and Compensation of Rolling Shutter Distortion for CMOS Image Sensor Arrays", ISCOM'05
- Nicklin et.al. "Rolling Shutter Image Compensation", Robocup 2006
- Chun et.al. "Suppressing rolling-shutter distortion of {CMOS} image sensors by motion vector detection", TCE'08
- Liang et.al. "Analysis and Compensation of Rolling Shutter Effect", Transaction on Image Processing 08
- Cho, Kong, "Affine Motion Based {CMOS} Distortion Analysis and {CMOS} Digital Image Stabilization", TCE'07
- S. Baker, E. Bennet, S.B. Kang, R. Szeliski, "Removing Rolling Shutter Wobble", CVPR'10
- Kim, Jayanthi, Kweon, "System-on-Chip Solution of Video Stabilization for CMOS Image Sensors in Hand-Held Devices", CSVT'11
- Forssén, Ringaby, "Rectifying rolling shutter video from hand-held devices", CVPR'10
- Ringaby, Forssén, "Efficient Video Rectification and Stabilisation for Cell-Phones", IJCV'12
- Hanning et.al. "Stabilizing Cell Phone Video using Inertial Measurement Sensors", IWMV'11
- Törnqvist, "Estimation and Detection with Applications to Navigation", PhD Thesis, Linköping University 2008
- C. Gramkow, "On Averaging Rotations", IJCV'01