Rectifying Rolling Shutter Video from Hand-Held Devices

Per-Erik Forssén
and Erik Ringaby

Computer Vision Laboratory
Dept. of Electrical Engineering
Linköping University
Sweden

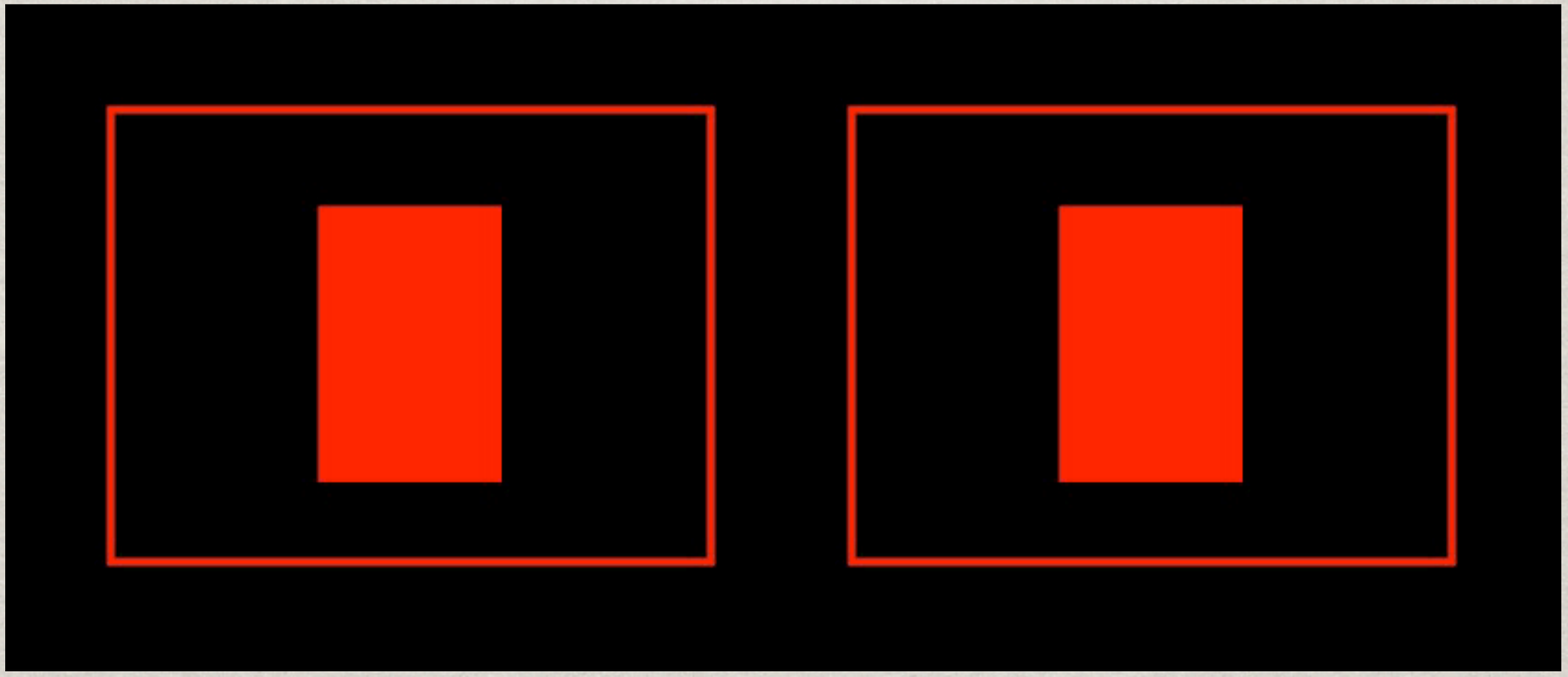
Cell-phone footage

- Non-smooth camera path
- Geometric distortions (wobble)

(c) 2010 Per-Erik Forssén

ROLLING SHUTTER SENSORS

- Image rows are read sequentially

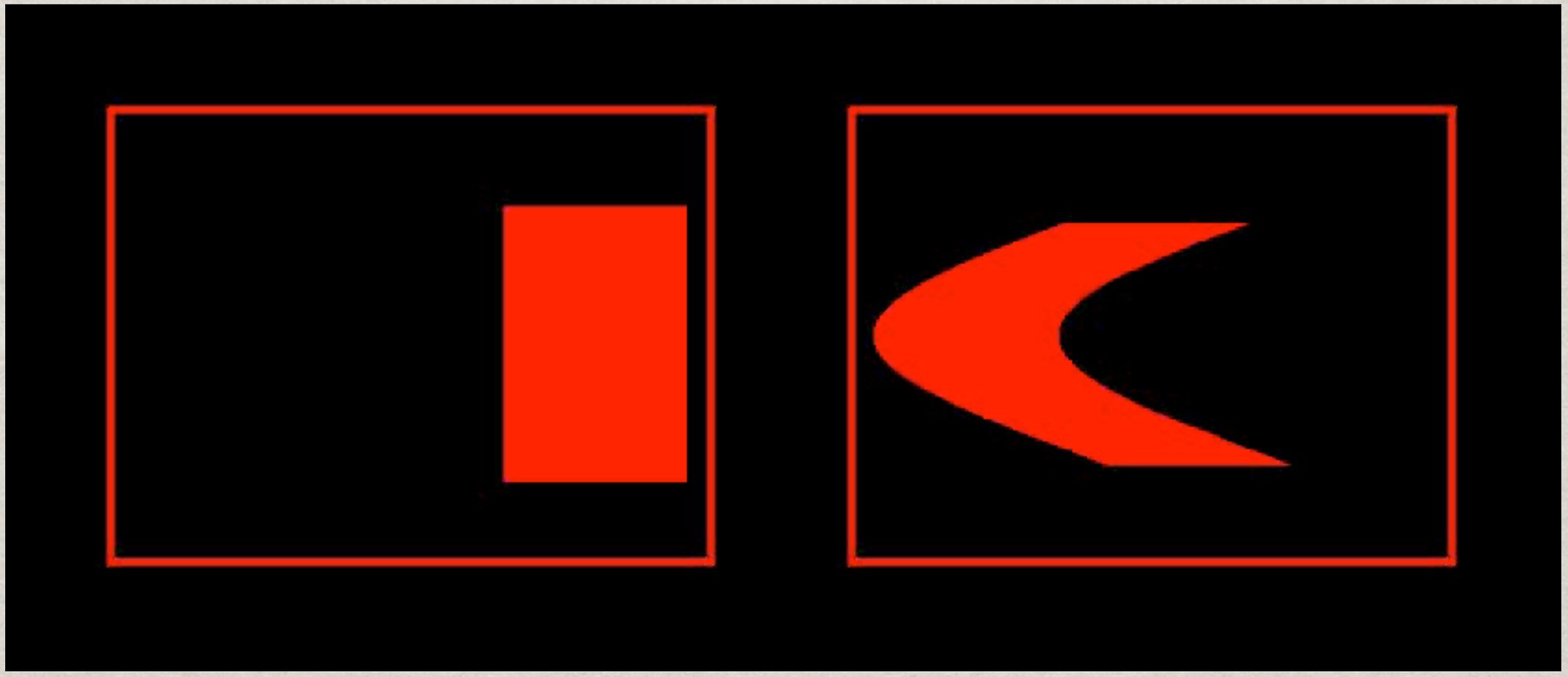


Static Scene Captured Image

(c) 2010 Per-Erik Forssén

ROLLING SHUTTER SENSORS

- Image rows are read sequentially

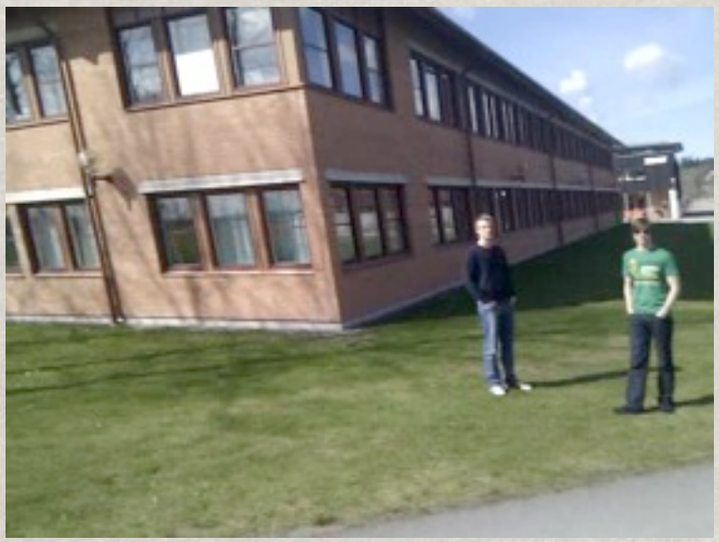


Dynamic Scene Captured Image

(c) 2010 Per-Erik Forssén

EXISTING SOLUTION:
MECHANICAL IMAGE STABILISATION (MIS)

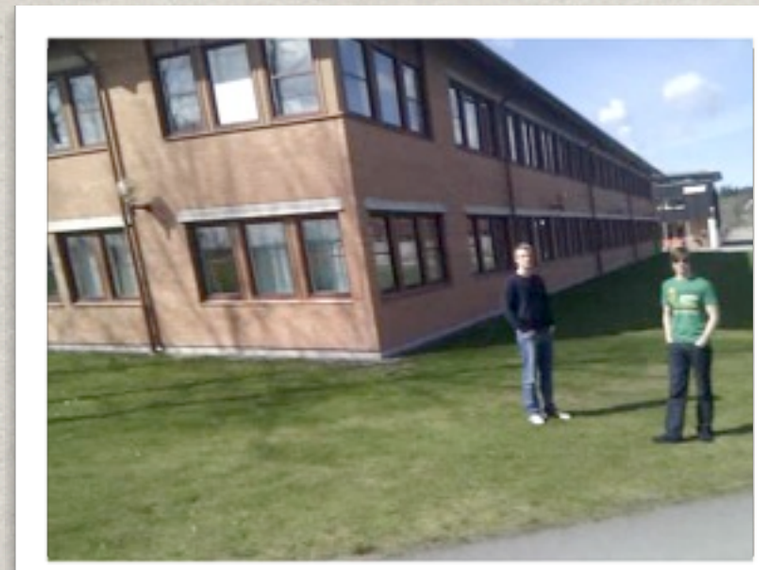
- **Assumption**: Small camera rotations.
- **Solution**: Tilt lens or move sensor to counteract motion.
- **Failure cases**: Camera **pan** and **rotation** and **moving objects**



(C) 2010 PER-ERIK FORSSÉN

EXISTING SOLUTION:
Mechanical image stabilisation (MIS)

- **Assumption**: Small camera rotations.
- **Solution**: Tilt lens or move sensor to counteract motion.
- **Failure cases**: Camera **pan** and **rotation** and **moving objects**



EXISTING SOLUTION:
Mechanical image stabilisation (MIS)

- **Assumption**: Small camera rotations.
- **Solution**: Tilt lens or move sensor to counteract motion.
- **Failure cases**: Camera **pan** and **rotation** and **moving objects**

EXISTING SOLUTION:
Mechanical image stabilisation (MIS)

- **Assumption**: Small camera rotations.
- **Solution**: Tilt lens or move sensor to counteract motion.
- **Failure cases**: Camera **pan** and **rotation** and **moving objects**

EXISTING SOLUTION:
OPTICAL FLOW BASED COMPENSATION

- **Assumptions**: Planar fronto-parallel scene [**Chun et al. TCE 2008**] or global affine deformation[**Liang et al. TIP 2008**].

- **Solution**: Estimate model for each frame and warp.

- **Failure cases**: **3D scenes** and **moving objects**

EXISTING SOLUTION:
OPTICAL FLOW BASED COMPENSATION

- **Assumptions**: Planar fronto-parallel scene [**Chun et al. TCE 2008**] or global affine deformation[**Liang et al. TIP 2008**].

- **Solution**: Estimate model for each frame and warp.

- **Failure cases**: **3D scenes** and **moving objects**

(c) 2010 Per-Erik Forssén

EXISTING SOLUTION:
OPTICAL FLOW BASED COMPENSATION

- **Assumptions**: Planar fronto-parallel scene [**Chun et al. TCE 2008**] or global affine deformation[**Liang et al. TIP 2008**].

- **Solution**: Estimate model for each frame and warp.

- **Failure cases**: **3D scenes** and **moving objects**

NEW SOLUTION: EGO-MOTION ESTIMATION

- Assume the camera is moving in a **static scene**.
- **3D camera motion** from a sparse optical flow.
- Use 3D motion to **rectify each row separately**

(c) 2010 Per-Erik Forssén

NEW SOLUTION: EGO-MOTION ESTIMATION

- Assume the camera is moving in a **static scene**.
- **3D camera motion** from a sparse optical flow.
- Use 3D motion to **rectify each row separately**
- Fewer parameters than the global affine motion field [**Liang08**] (or the same).
- Better models the cause of the distortions.

NEW SOLUTION: EGO-MOTION ESTIMATION

- Algorithm overview:

1. Find inter-frame **correspondences** using a KLT tracker [**Lucas&Kanade81**].
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

NEW SOLUTION: EGO-MOTION ESTIMATION

- Algorithm overview:

1. Find inter-frame **correspondences** using a KLT tracker [**Lucas&Kanade81**].
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

NEW SOLUTION: EGO-MOTION ESTIMATION

- Algorithm overview:

1. Find inter-frame **correspondences** using a KLT tracker [**Lucas&Kanade81**].
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

NEW SOLUTION: EGO-MOTION ESTIMATION

- Algorithm overview:

1. Find inter-frame **correspondences** using a KLT tracker [**Lucas&Kanade81**].
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

NEW SOLUTION: EGO-MOTION ESTIMATION

- Algorithm overview:

1. Find inter-frame **correspondences** using a KLT tracker [**Lucas&Kanade81**].
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

NEW SOLUTION: EGO-MOTION ESTIMATION

- Algorithm overview:

1. Find inter-frame **correspondences** using a KLT tracker [**Lucas&Kanade81**].
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

EGO-MOTION ESTIMATION

- Recall the regular pin-hole camera

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{d}]\mathbf{X}$$

- Instead we now have

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}(t)|\mathbf{d}(t)]\mathbf{X}$$

- where $t - t_0 \propto x_2/x_3$
  (i.e. time is proportional to image row)

EGO-MOTION ESTIMATION

- Recall the regular pin-hole camera

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{d}]\mathbf{X}$$

- Instead we now have

Notation:

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}(t)|\mathbf{d}(t)]\mathbf{X} = \mathbf{K}[\mathbf{R}(\mathbf{x})|\mathbf{d}(\mathbf{x})]\mathbf{X}$$

- where $t - t_0 \propto x_2/x_3$
  (i.e. time is proportional to image row)

EGO-MOTION ESTIMATION

- Three different simplifying assumptions:
  - **Rotation** only
  - **Dominant plane** in the scene
  - **Translation** only

- Express each correspondence constraint with a homography

$$\mathbf{x}_k \leftrightarrow \mathbf{x}_l$$

$$\mathbf{x}_k \sim \mathbf{H}\mathbf{x}_l$$

EGO-MOTION ESTIMATION

- Rotation homography model:

$$\mathbf{x}_1 \sim \mathbf{KR}(\mathbf{x}_1)\mathbf{X} \qquad \Rightarrow \qquad \mathbf{x}_1 \sim \mathbf{H}\mathbf{x}_2$$
$$\mathbf{x}_2 \sim \mathbf{KR}(\mathbf{x}_2)\mathbf{X} \qquad \mathbf{H} = \mathbf{KR}(\mathbf{x}_1)\mathbf{R}^T(\mathbf{x}_2)\mathbf{K}^{-1}$$

- Valid if the distance to scene objects is large compared to the baseline

- Gave better results than the other two.

EGO-MOTION ESTIMATION

- Dominant plane model:

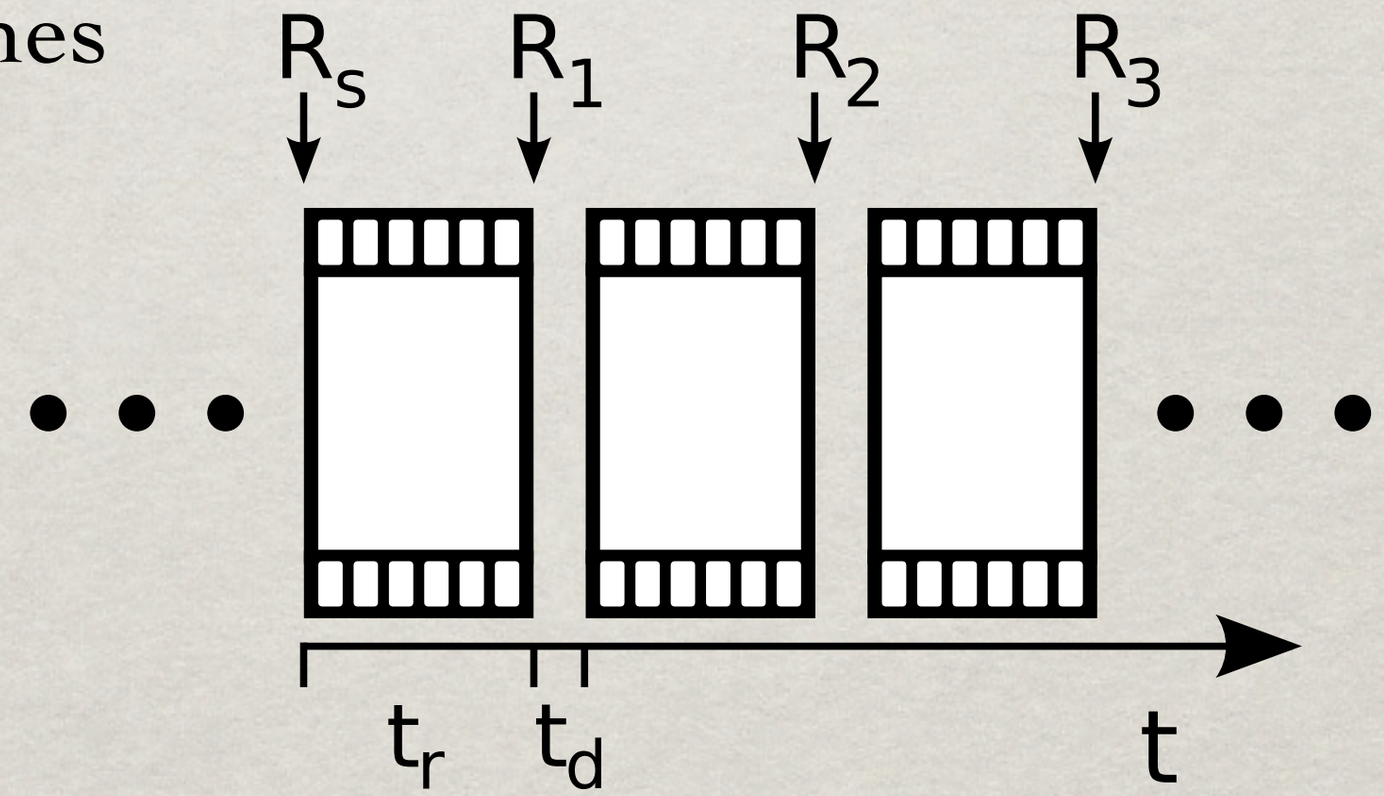
$$\mathbf{H} = \mathbf{K}\mathbf{R}(\mathbf{x}_1)\mathbf{D}(\mathbf{x}_1)\mathbf{D}(\mathbf{x}_2)^{-1}\mathbf{R}^T(\mathbf{x}_2)\mathbf{K}^{-1}$$

- Pure translational model:

$$\mathbf{H} = \mathbf{K}\mathbf{D}(\mathbf{x}_1)\mathbf{D}(\mathbf{x}_2)^{-1}\mathbf{K}^{-1}$$

EGO-MOTION ESTIMATION

- Rotations and translations are represented as linear splines



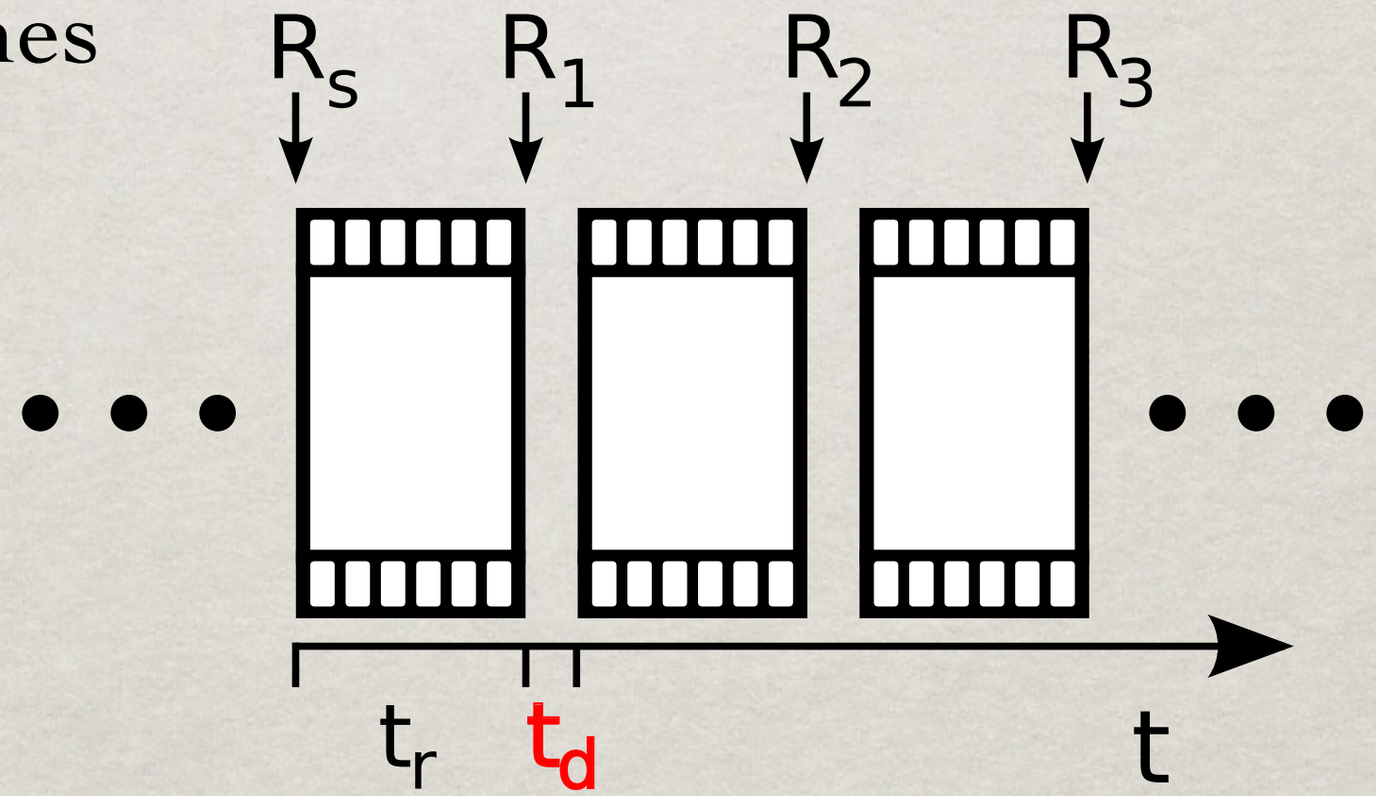
$$\mathbf{d}_{\text{interp}} = (1 - w)\mathbf{d}_1 + w\mathbf{d}_2$$

$$\mathbf{R}_{\text{interp}} = \texttt{SLERP}(\mathbf{R}_1, \mathbf{R}_2, w)$$

(c) 2010 Per-Erik Forssén

EGO-MOTION ESTIMATION

- Rotations and translations are represented as linear splines



$$\mathbf{d}_{\text{interp}} = (1-w)\mathbf{d}_1 + w\mathbf{d}_2$$

$$\mathbf{R}_{\text{interp}} = \texttt{SLERP}(\mathbf{R}_1, \mathbf{R}_2, w)$$

(c) 2010 Per-Erik Forssén

EGO-MOTION ESTIMATION

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

$$J = \epsilon(\mathbf{n}_1, \ldots, \mathbf{n}_N, \mathbf{d}_1, \ldots, \mathbf{d}_N)$$

- $\mathbf{n_n}$ 3-element axis-angle vectors
  $\mathbf{d_n}$ translation vectors

EGO-MOTION ESTIMATION

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

$$J = \epsilon(\mathbf{n}_1, \ldots, \mathbf{n}_N, \mathbf{d}_1, \ldots, \mathbf{d}_N)$$

- $\mathbf{n_n}$ 3-element axis-angle vectors
  $\mathbf{d_n}$ translation vectors

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

Rectification

- Each line is rectified with a separate homography $\mathbf{x}' = \mathbf{K}\mathbf{R}_o^T\mathbf{R}(\mathbf{x})\mathbf{K}^{-1}\mathbf{x}$

Original Corrected

SYNTHETIC DATASET

- Rendered using Autodesk Maya™
- Camera trajectories
- Ideal reconstructions
- Masks of seen pixels

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

SYNTHETIC DATASET

- Example

Rolling shutter image

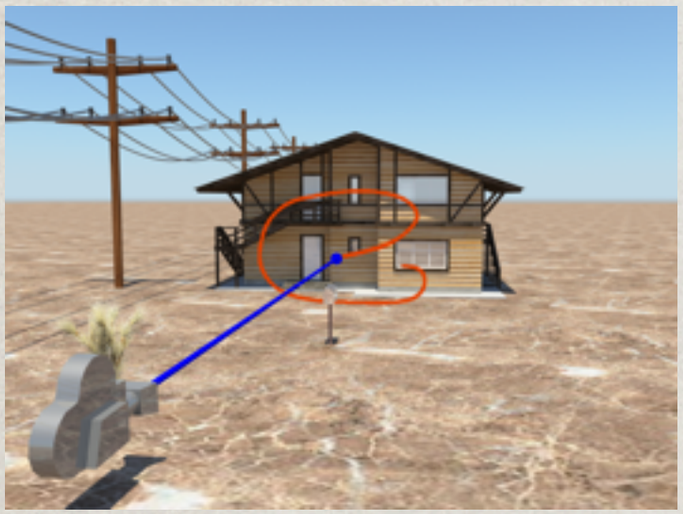
Ground truth rectification

Visible pixels mask

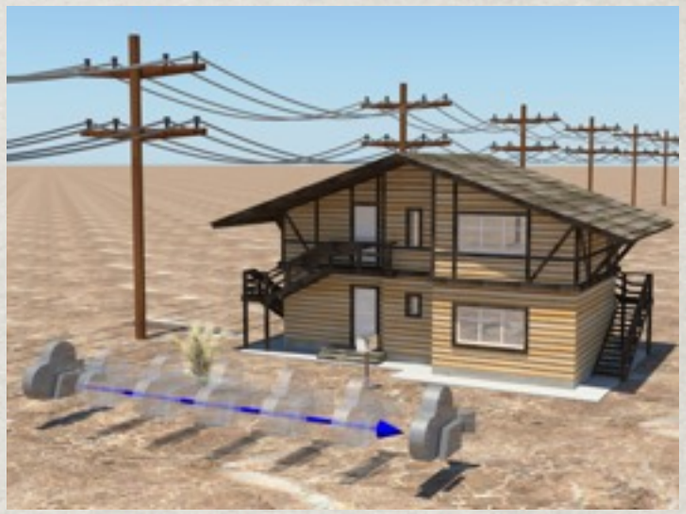
http://www.cvl.isy.liu.se/research/rs-dataset/

SYNTHETIC DATASET

- Four categories

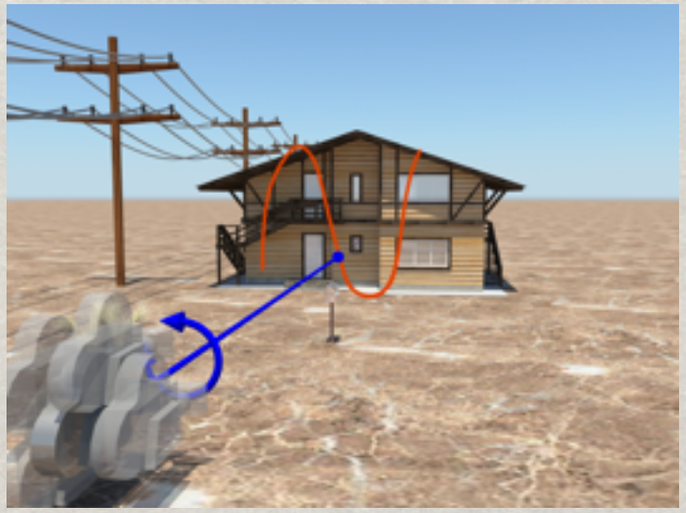


Rotation (aim)



Translation

Rotation (aim+in plane)



Rotation+translation

(c) 2010 Per-Erik Forssén

SYNTHETIC DATASET

- Measure of accuracy

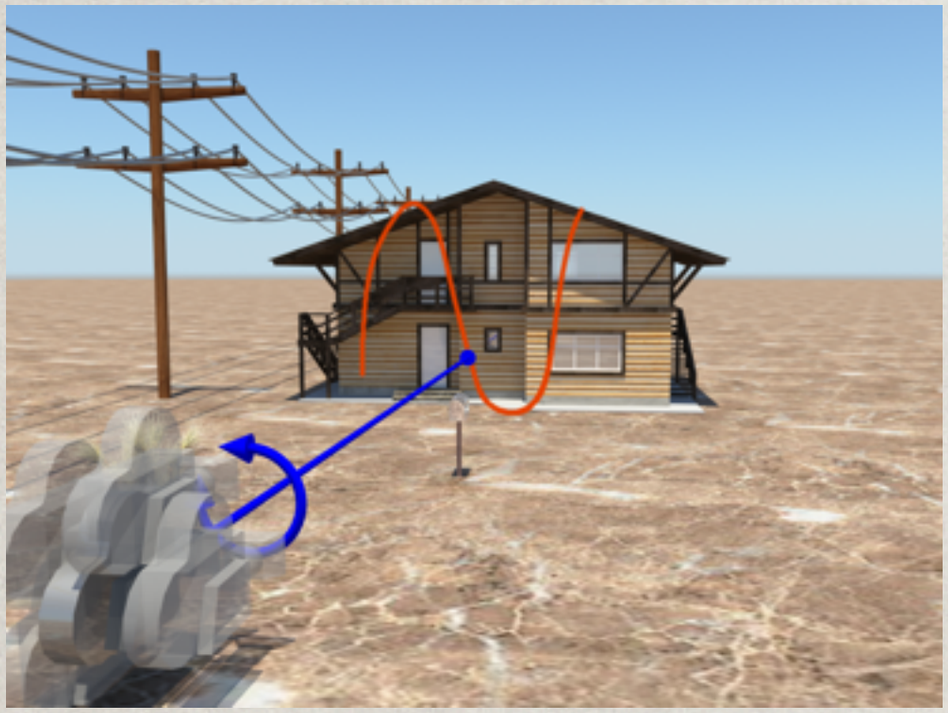
$$\texttt{accuracy} = \frac{\#\texttt{correct pixels}}{\texttt{total } \#\texttt{pixels}}$$

- Thresholded Euclidean RGB error

  correct if $d(\mathbf{I}_{\text{gt}}(x,y), \mathbf{I}_{\text{rect}}(x,y)) < 0.3$

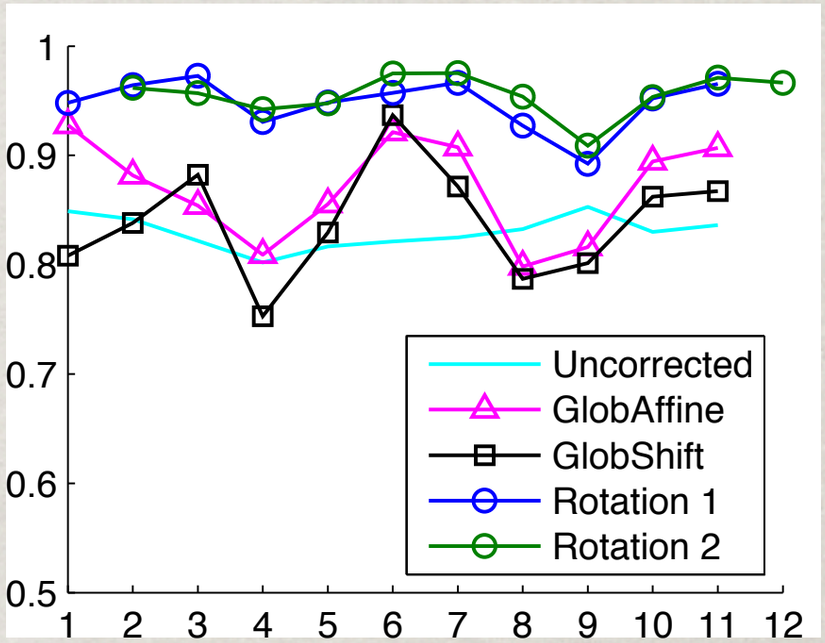
- Only counted within visibility mask

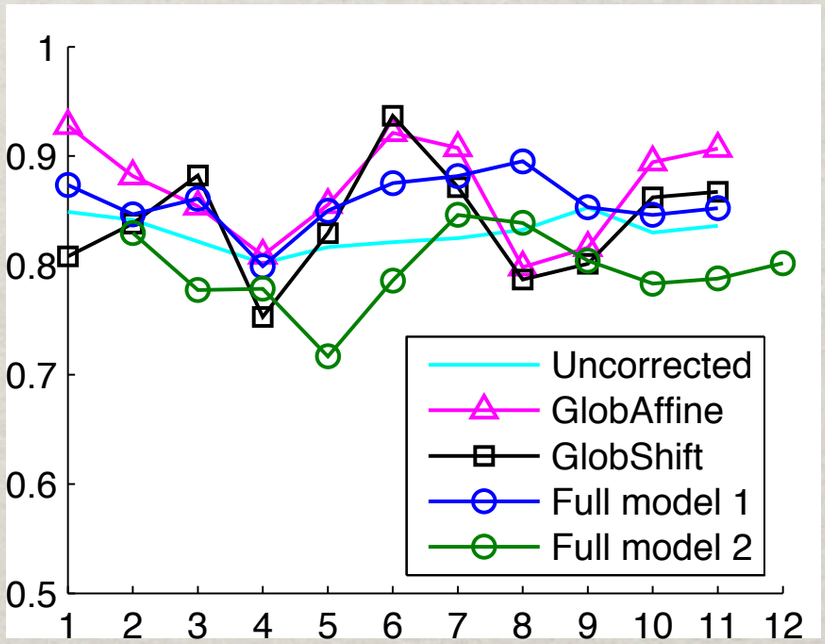
RESULTS



Rotation+translation

More experiments in the paper





(c) 2010 Per-Erik Forssén

ROLLING SHUTTER RECTIFICATION

- Frame from HTC Desire camera (input)

ROLLING SHUTTER RECTIFICATION

- Frame from HTC Desire camera (rectified)

(c) 2010 Per-Erik Forssén

ROLLING SHUTTER RECTIFICATION

- Frame from iPhone 3GS camera (input)

(c) 2010 Per-Erik Forssén

ROLLING SHUTTER RECTIFICATION

- Frame from iPhone 3GS camera (rectified)

(c) 2010 Per-Erik Forssén

ROLLING SHUTTER RECTIFICATION

- Stabilised video comparison

(c) 2010 Per-Erik Forssén

ROLLING SHUTTER RECTIFICATION

- Stabilised video comparison (3x slowmotion)

Stabilised Rectified and stabilised

ROLLING SHUTTER RECTIFICATION

- Rectified video comparison (2.5x slowmotion)

Input

Rectified

(c) 2010 Per-Erik Forssén

Original
Rectified
Stabilised
Stabilised&Rectified

Final remarks

- Modelling the 3D camera motion improves **rectification accuracy**
- We get the camera trajectory for free
- Useful for **video stabilisation**
- Fusion with MEMS gyro measurements is straightforward (as available in e.g. the iPhone 4).