Computer Vision on Rolling Shutter Cameras

PART IV: RS & the Kinect

Per-Erik Forssén, Erik Ringaby, Johan Hedborg

Computer Vision Laboratory
Dept. of Electrical Engineering
Linköping University

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

The Kinect sensor

- Designed for player interaction with the Xbox 360
- "You are the Controller"
- 3D scanner
- Accelerometer sensor
- Skeletal tracking

- Stationary in your living room

The Kinect sensor

- The H/W gained popularity in the research community

- Quasi-dense depth maps in 30 Hz

- Cheap (~100 USD)

The Kinect sensor

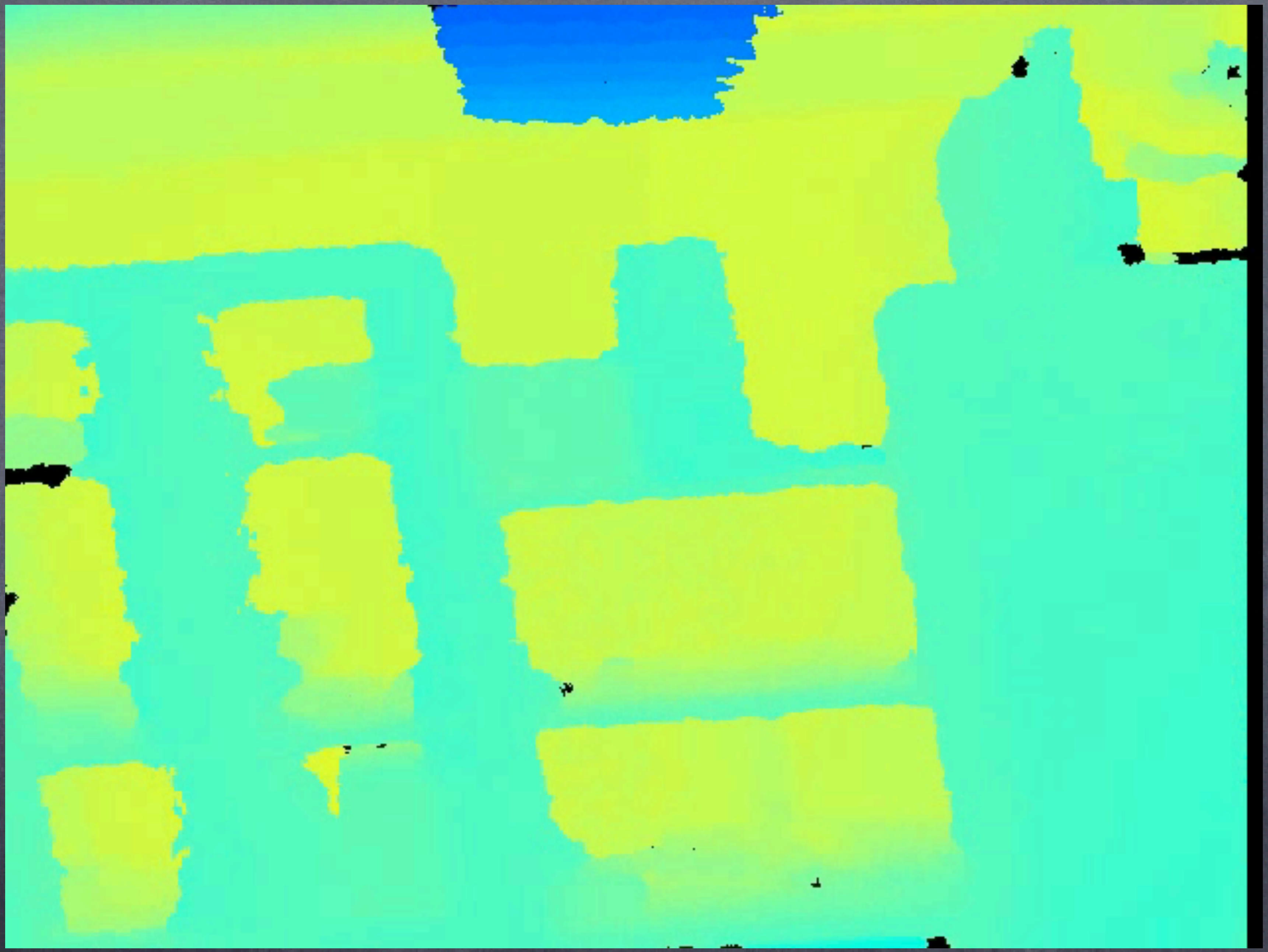
- Range estimation by triangulation

- A - structured NIR laser projector
- B - CMOS Colour camera
- C - CMOS NIR camera

RS on Kinect

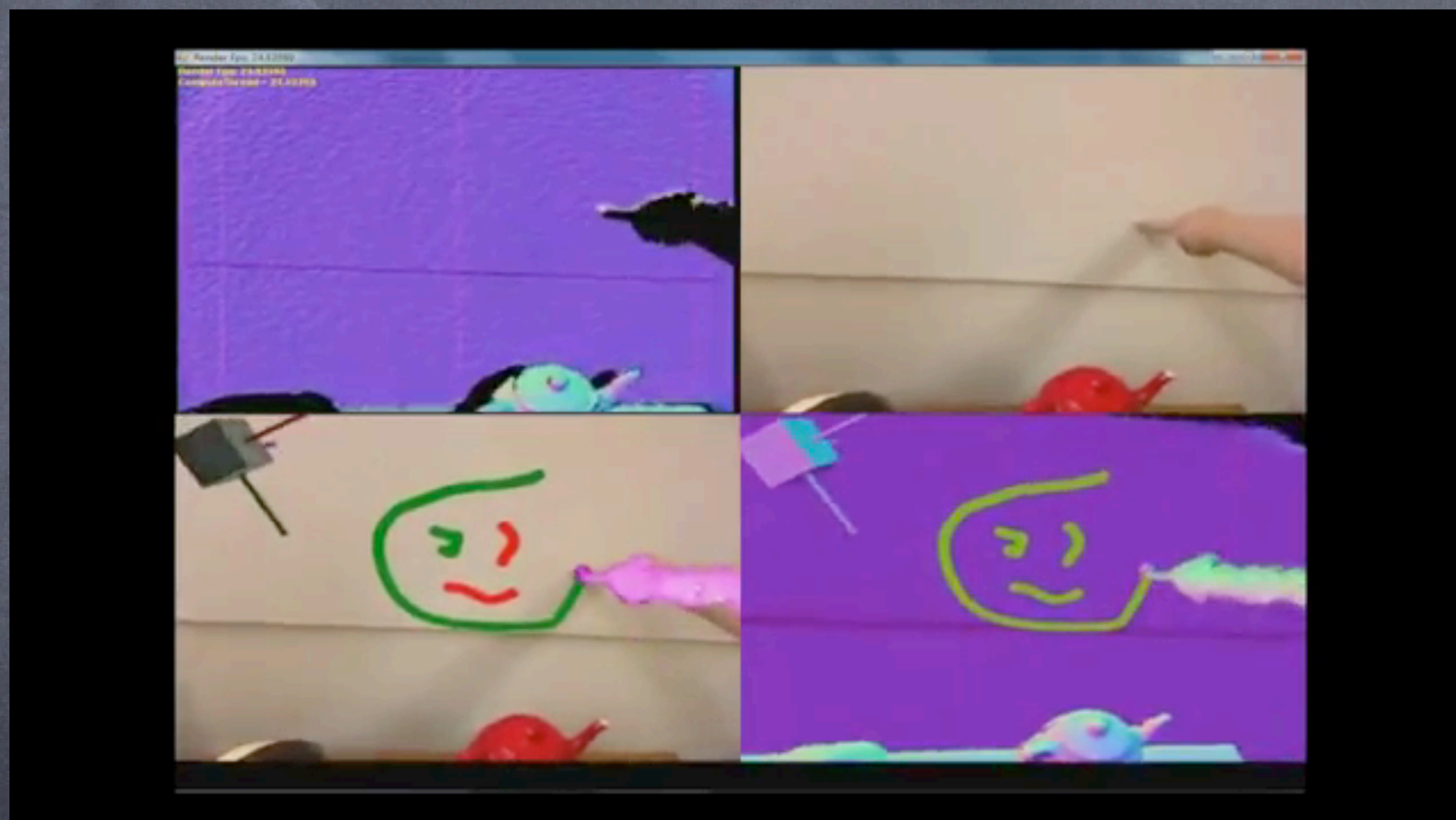
- Both Kinect cameras make use of rolling shutters
- Not a big problem when it is used for gaming:
  - Stationary in your living room
  - People are moving quite slowly far away from the sensor
- It is however also popular to use the Kinect sensor on mobile platforms
  → RS problems

Kinect footage



Kinect footage

Augmented reality by KinectFusion



[Izadi et al. SIGGRAPH'11]

RS on Kinect

- A similar approach as for the video case, but now we also have the depth
- 3D point correspondences enables us to estimate the full 3D motion (rotation and translation)
- The 3D point cloud can be rectified

[Ringaby & Forssén, ICCV'11]

Synchronization problem

- RGB and depth map:
  - Different readout time
  - Different fields-of-view
  - Correspondence problematic under camera motion

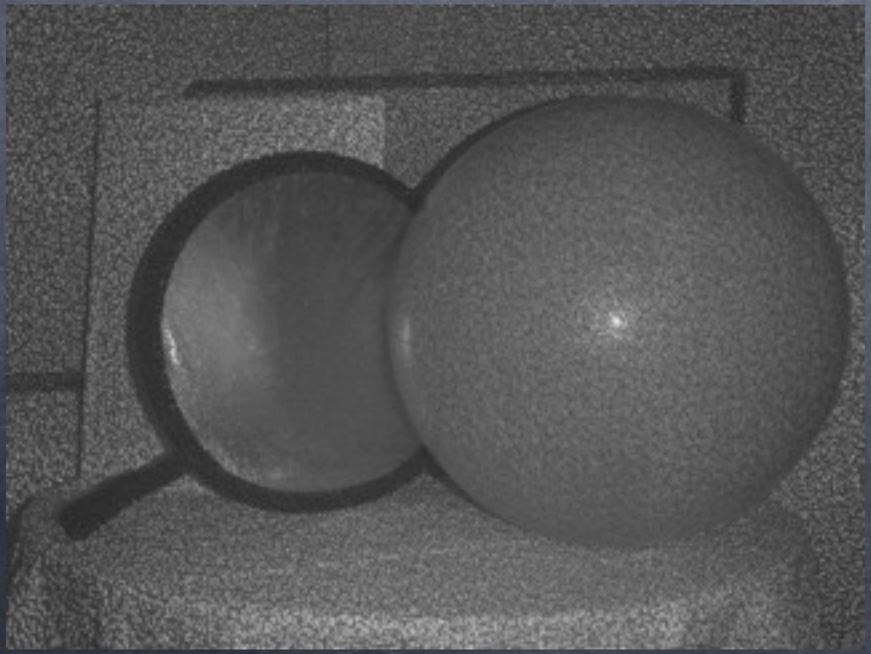
Synchronization problem

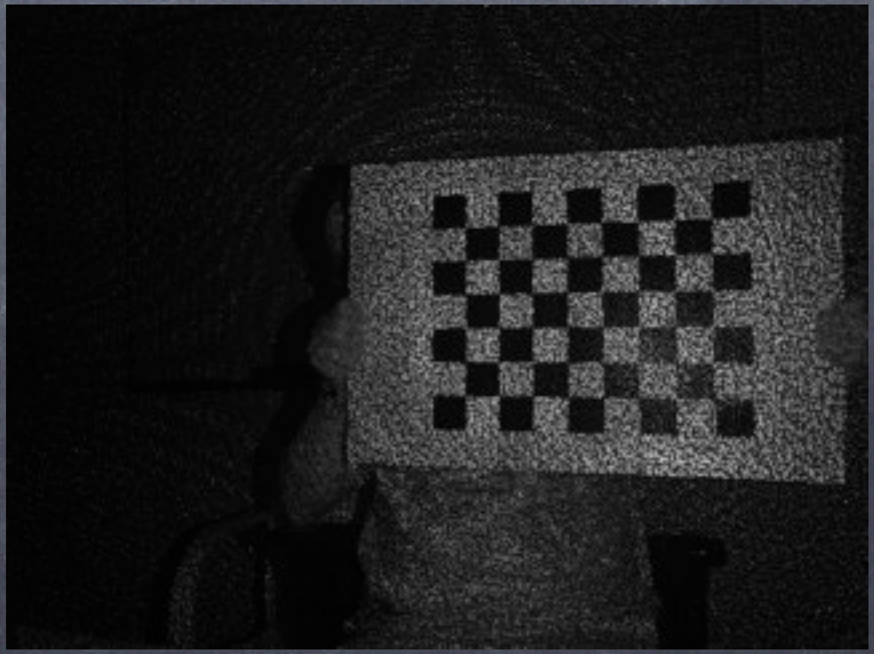
- RGB and depth map:
  - Different readout time
  - Different fields-of-view
  - Correspondence problematic under camera motion

- Solution:
  Use NIR images

NIR images

- NIR and depth map from the same sensor
- NIR camera uses shorter shutter speeds, less motion blur
- Drawback, we need to suppress the structured light pattern (SLP)





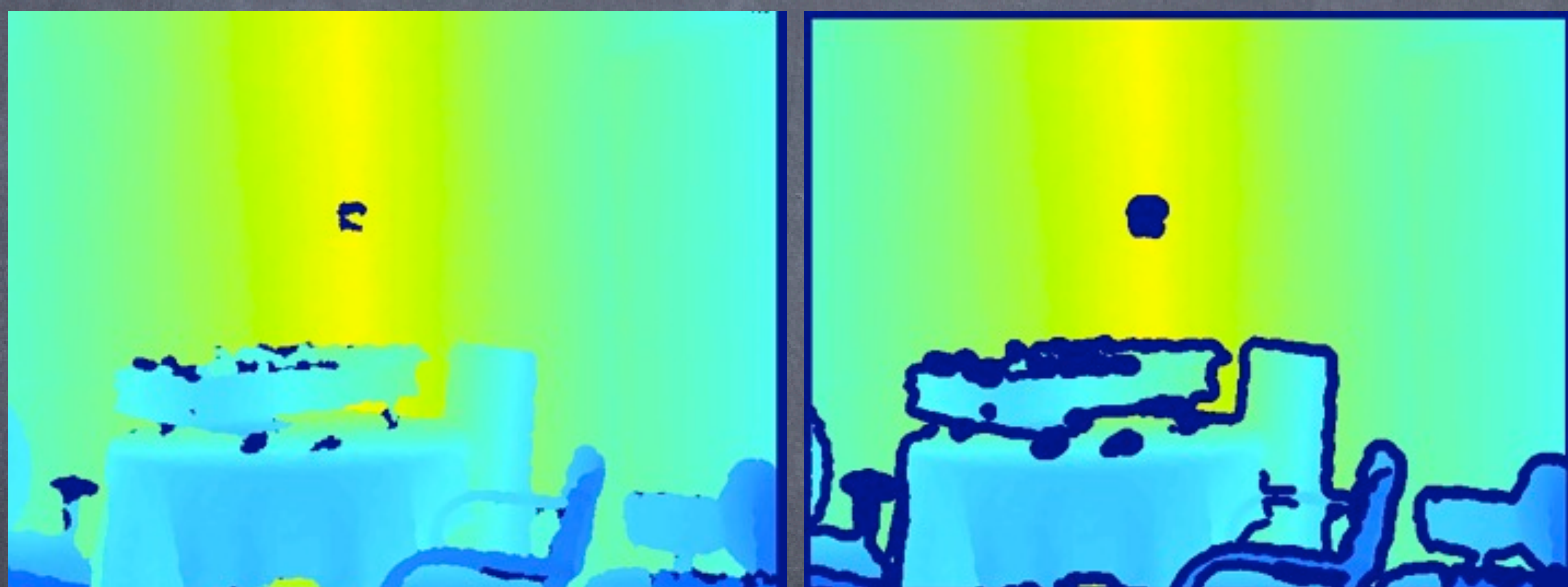
Suppressing the SLP

Source code available at: http://users.isy.liu.se/cvl/perfo/software/

Outlier rejection

- Optimisation sensitive to point corr. outliers
- Three rejection steps:
  - KLT cross-checking
  - Depth map edge detection
  - Procrustes RANSAC

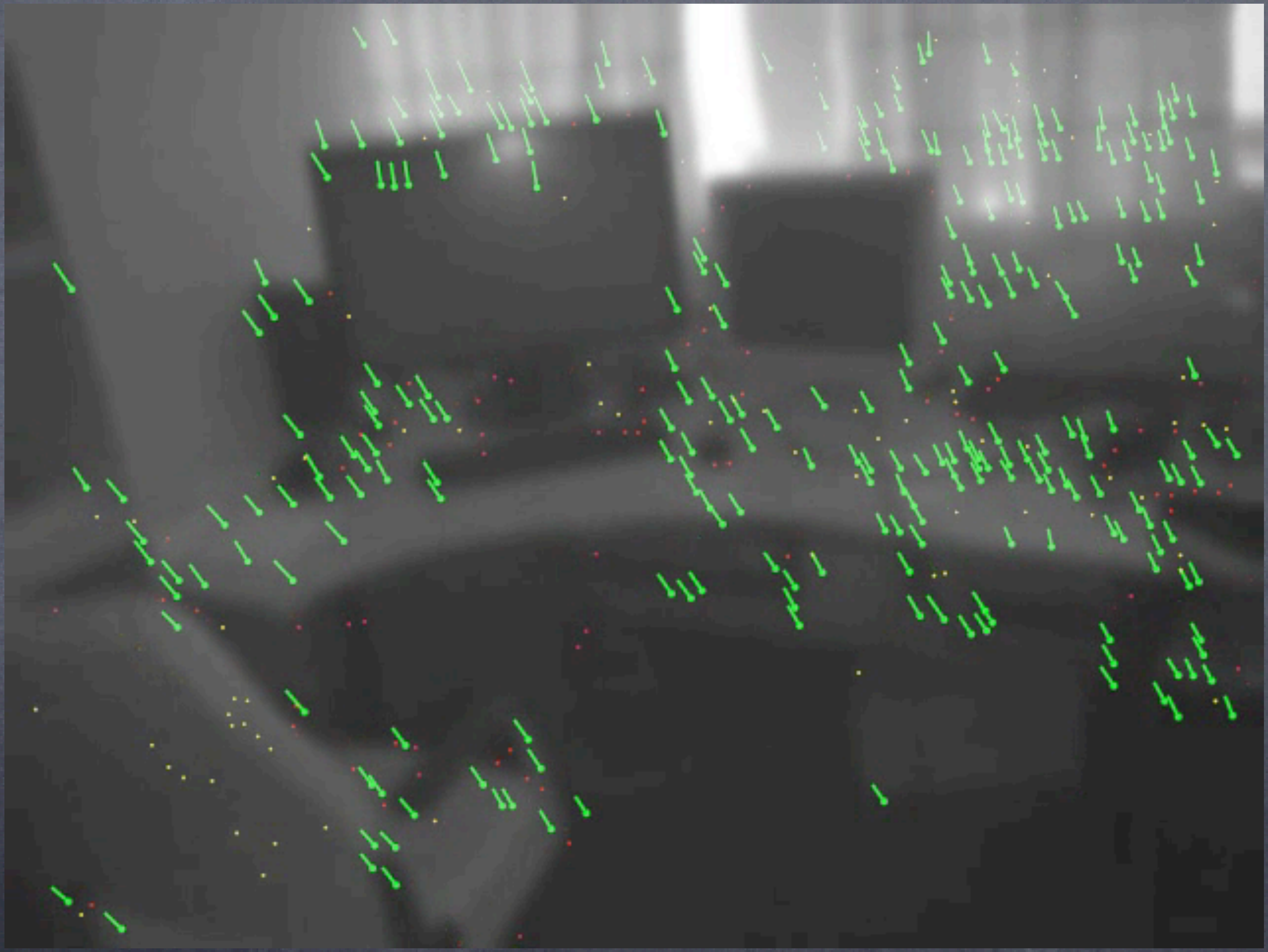
Depth map edge detection



Procrustes RANSAC

- Kinect depth map is noisy
- Estimate global translation and rotation between point clouds with Procrustes alg. [Viklands06] (RANSAC)
- When finished, reject those point correspondences which are above a threshold

Point correspondences



Sensor Geometry

- A 3D point X relates to its corresponding homogenous image point x as:

$$\mathbf{x} = \mathbf{K}\mathbf{X}\ , \ \text{and} \quad \mathbf{X} = z(\mathbf{x})\mathbf{K}^{-1}\mathbf{x}$$

where **K** is the intrinsic camera matrix and z(x) is the point's value in the depth image

- We model the camera motion as a seq. of rotation matrices **R**(t) and translation vectors **d**(t)

Sensor motion estimation

- A point in image 1, at row N_x corr. to 3D point X_1 and
- A point in image 2, at row N_y corr. to 3D point X_2. They can be transformed to X_0:

$$\mathbf{X_0} = \mathbf{R}(N_1)\mathbf{X}_1 + \mathbf{d}(N_1)$$
$$\mathbf{X_0} = \mathbf{R}(N_2)\mathbf{X}_2 + \mathbf{d}(N_2)$$

- X_0 is the position the point should have, if it was imaged the same time as the first row in image 1

Sensor motion estimation

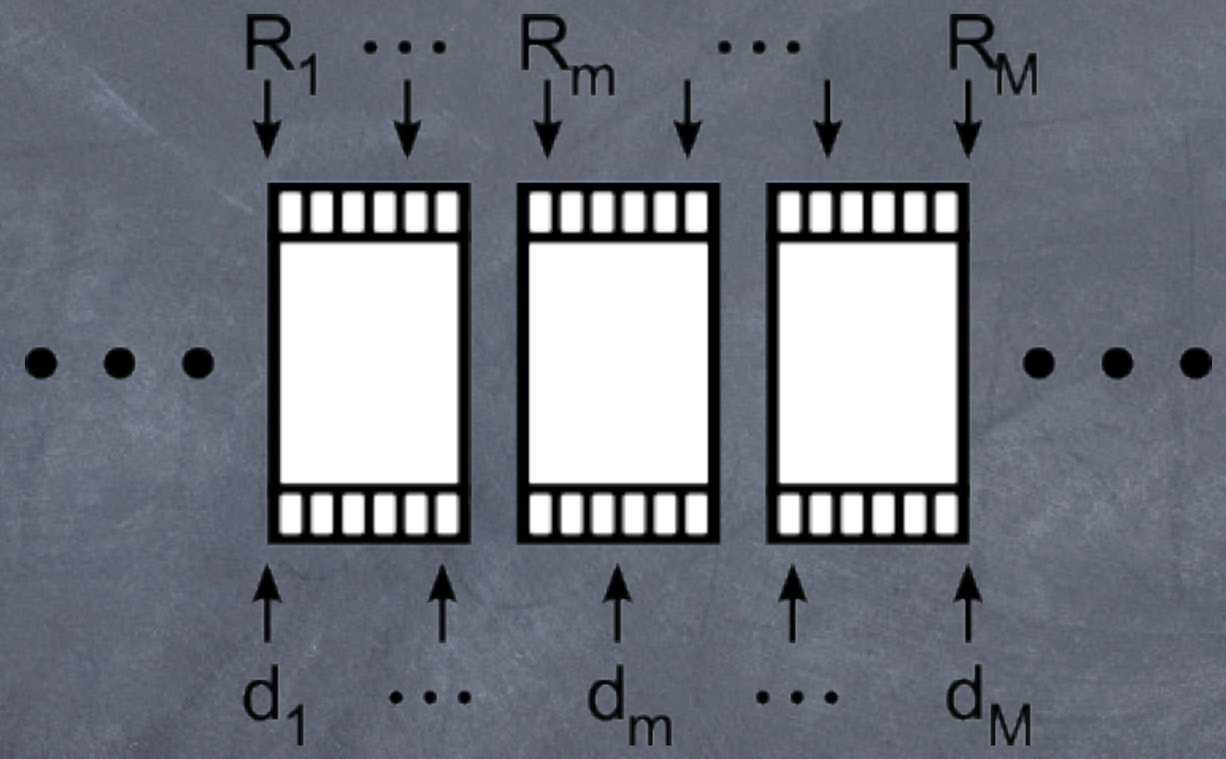
- We use this cost function to solve for the rotation and translation:

$$J = \sum_{k=1}^{K} ||\mathbf{R}(N_{1,k})\mathbf{X}_{1,k} + \mathbf{d}(N_{1,k}) - \mathbf{R}(N_{2,k})\mathbf{X}_{2,k} - \mathbf{d}(N_{2,k})||^2$$

where K is the number of point corr.

- 12 unknowns, 3 equations per point corr.
- **R** "key-rotations" as before, and **d** "key-translations"

Sensor motion estimation



”Key-rotations” and ”key-translations”

- SLERP for rotations and
- Linear interpolation for translations

Rectification

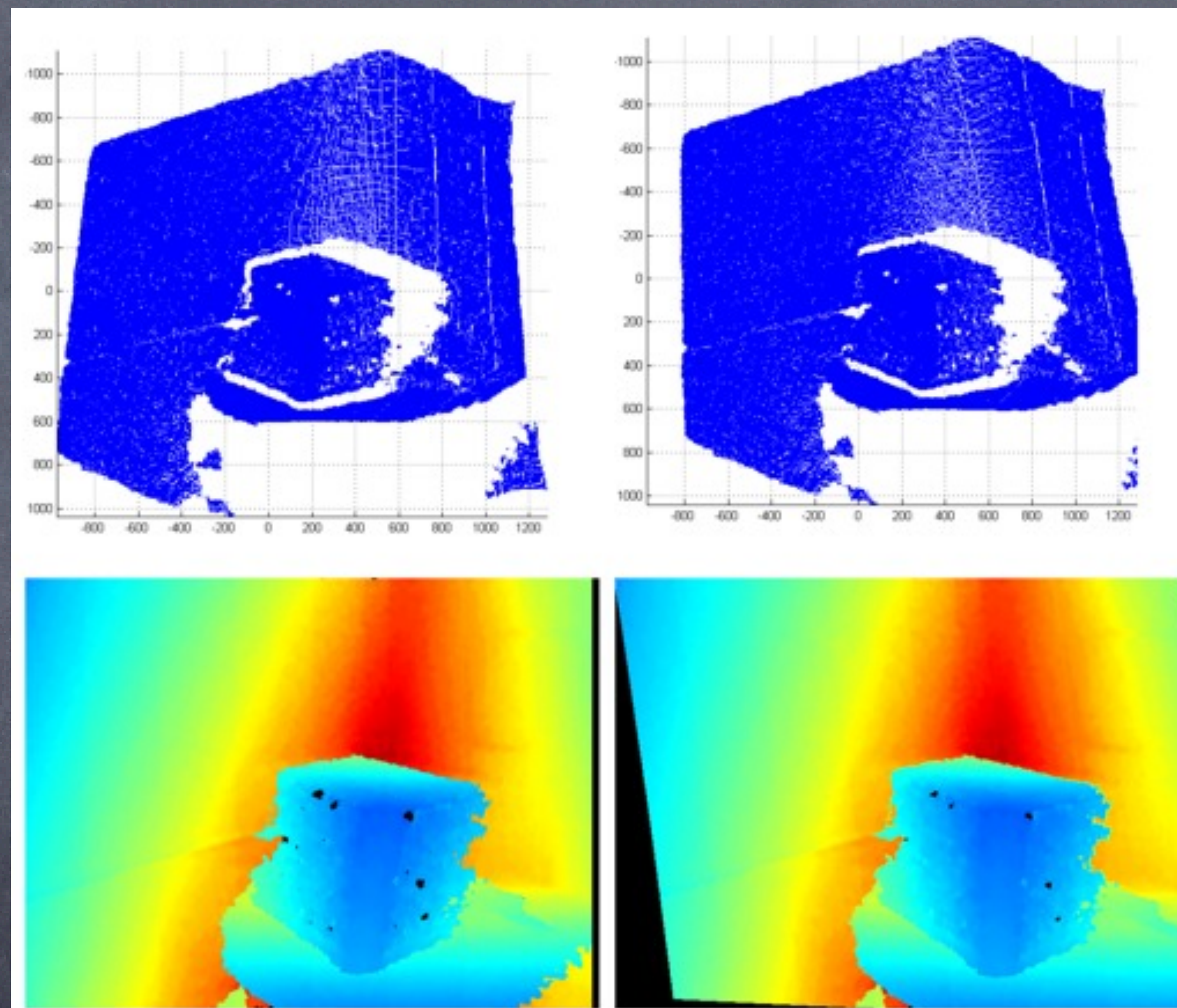
- When the camera motion has been estimated the 3D point clouds can be rectified with

$$\mathbf{X}' = \mathbf{R}_{\mathrm{ref}}(\mathbf{R}(N_1)\mathbf{X}_1 + \mathbf{d}(N_1)) + \mathbf{d}_{\mathrm{ref}}$$

- By projecting the points through the camera, depth map and video frames can also be rectified:

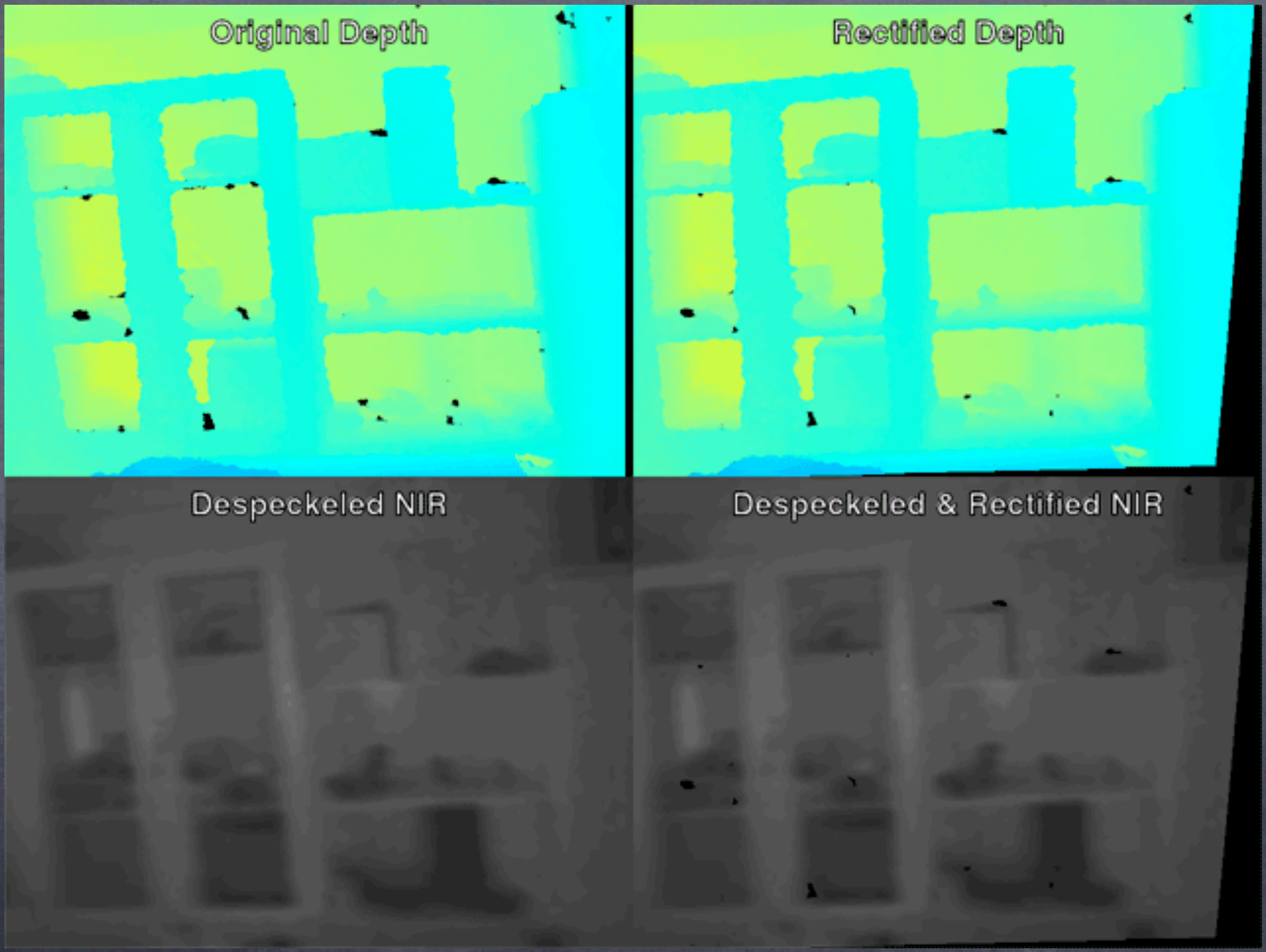
$$\mathbf{x}' = \mathbf{K}[\mathbf{R}_{\mathrm{ref}}(\mathbf{R}(N_1)\mathbf{X}_1 + \mathbf{d}(N_1)) + \mathbf{d}_{\mathrm{ref}}]$$

Rectification



Original Rectified

Rectification



Summary

- When the Kinect is used on a mobile platform rolling-shutter distortions will be present
- Using correspondences in the NIR images avoids depth-to-image registration problem
- Depth map noisy, optimisation in 3D more sensitive than "video approach"
- Full 6-DOF motion can be estimated and corrected for

References

- Viklands, "Algorithms for the Weighted Orthogonal Procrustes Problem and Other Least Squares Problems", Umeå University 2006
- Ringaby, Forssén, "Scan Rectification for Structured Light Range Sensors with Rolling Shutters", ICCV'11
- Izadi et-al "KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction", SIGGRAPH'11