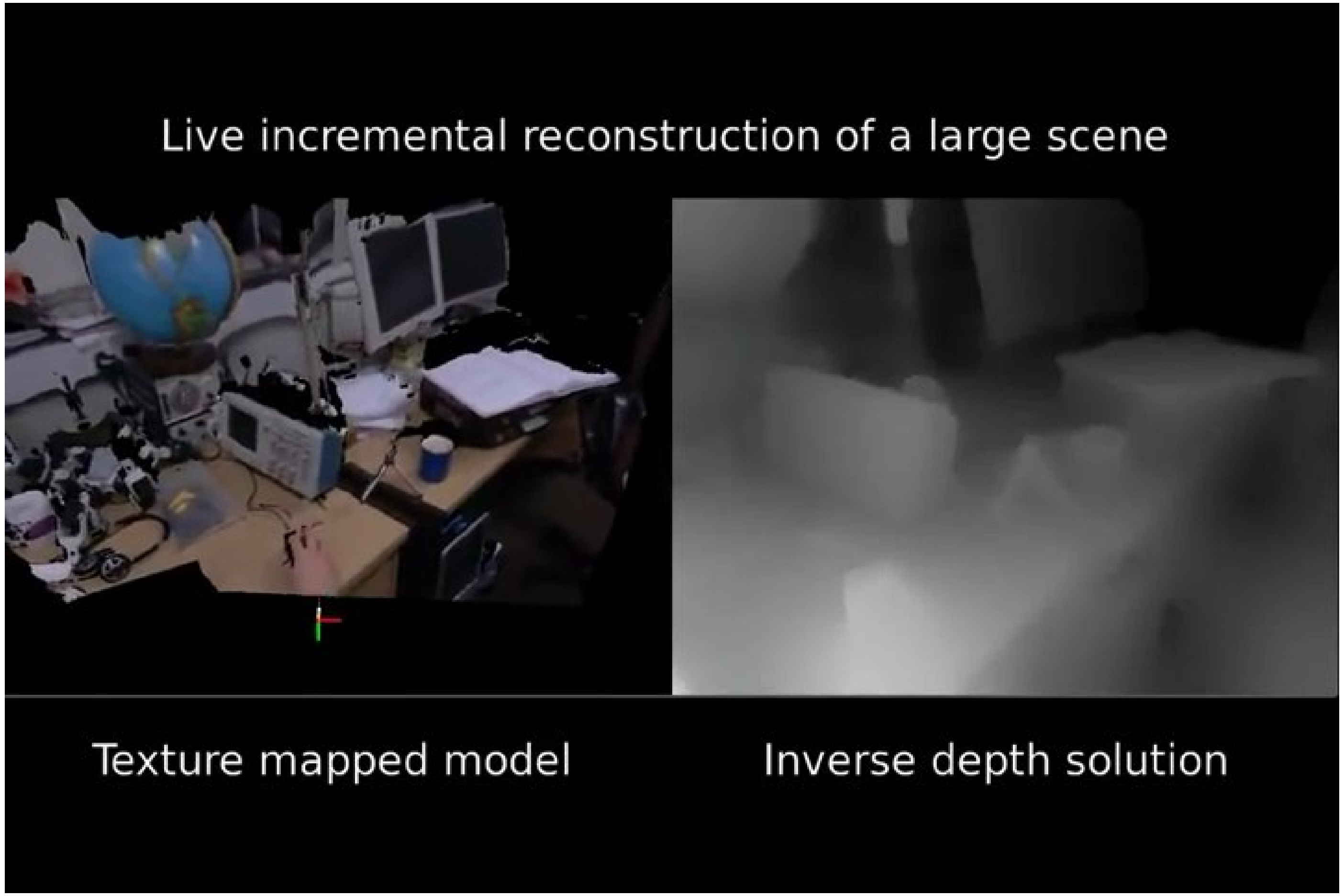
Vision Algorithms for Mobile Robotics

Lecture 12b
Dense 3D Reconstruction

Davide Scaramuzza

http://rpg.ifi.uzh.ch

DTAM: Dense Tracking and Mapping in Real-Time

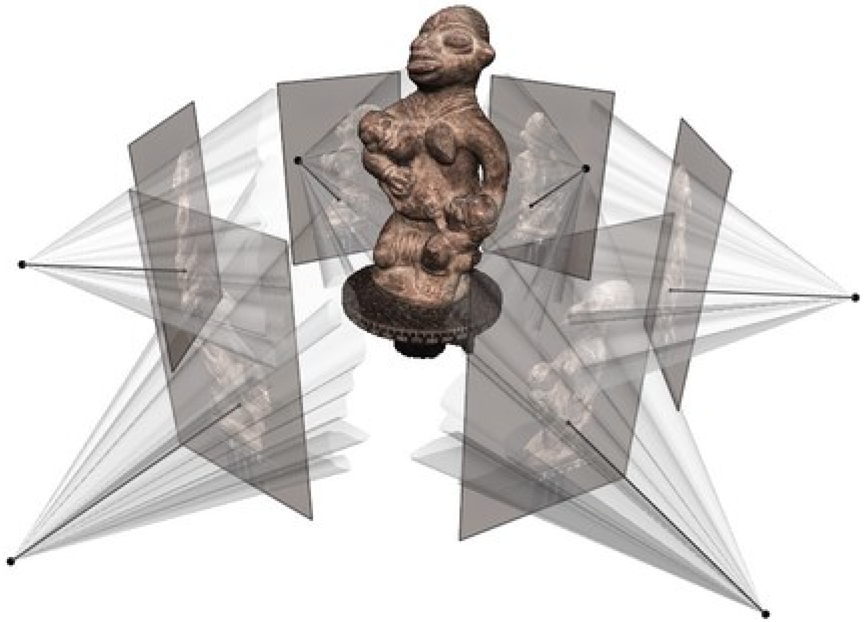


Newcombe, Lovegrove, Davison, *DTAM: Dense Tracking and Mapping in Real-Time*, International Conference on Computer Vision (ICCV), 2011. PDF.

Dense Reconstruction (or Multi-view stereo)

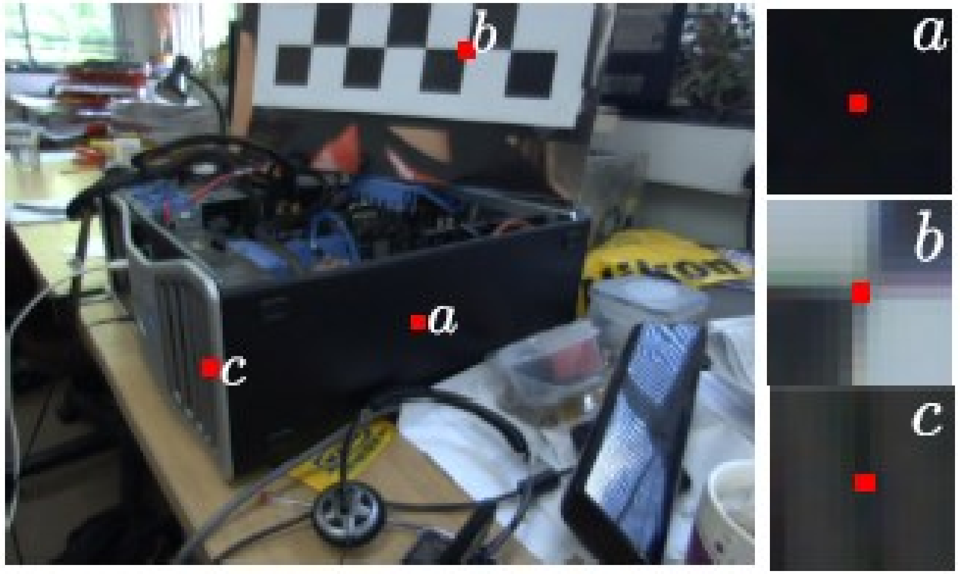
Problem definition:

- **Input**: calibrated images from several viewpoints (i.e., K, R, T are known for each camera, e.g., from SFM)
- **Output**: 3D object **dense** reconstruction (ideally of every pixel)



Challenges

- **Dense reconstruction** requires establishing **dense correspondences**
- But **not all pixels can be matched** reliably:
  - flat regions,
  - edges,
  - viewpoint and illumination changes,
  - occlusions

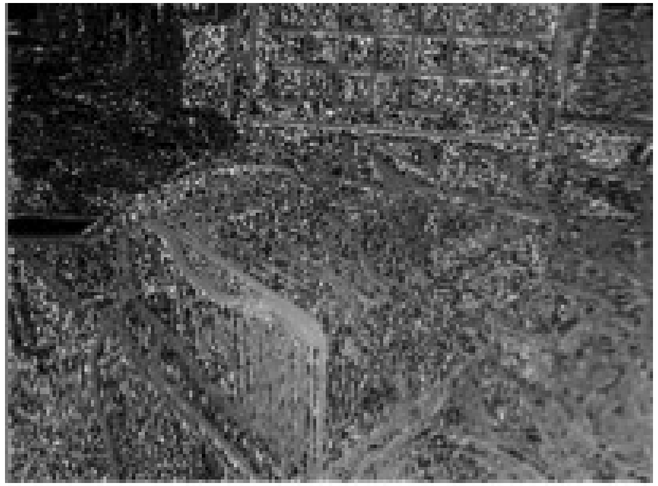


Idea: Take advantage of many small-baseline views where high-quality matching is possible

Dense reconstruction workflow

Step 1: Local methods

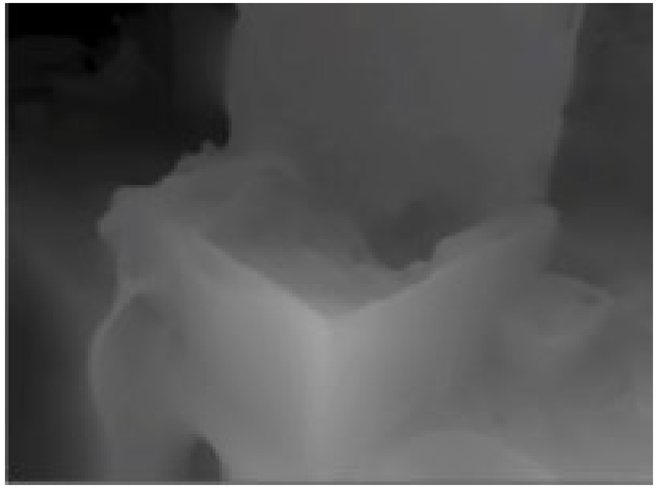
- Estimate depth independently for each pixel



How do we compute correspondences for *every* pixel?

Step 2: Global methods

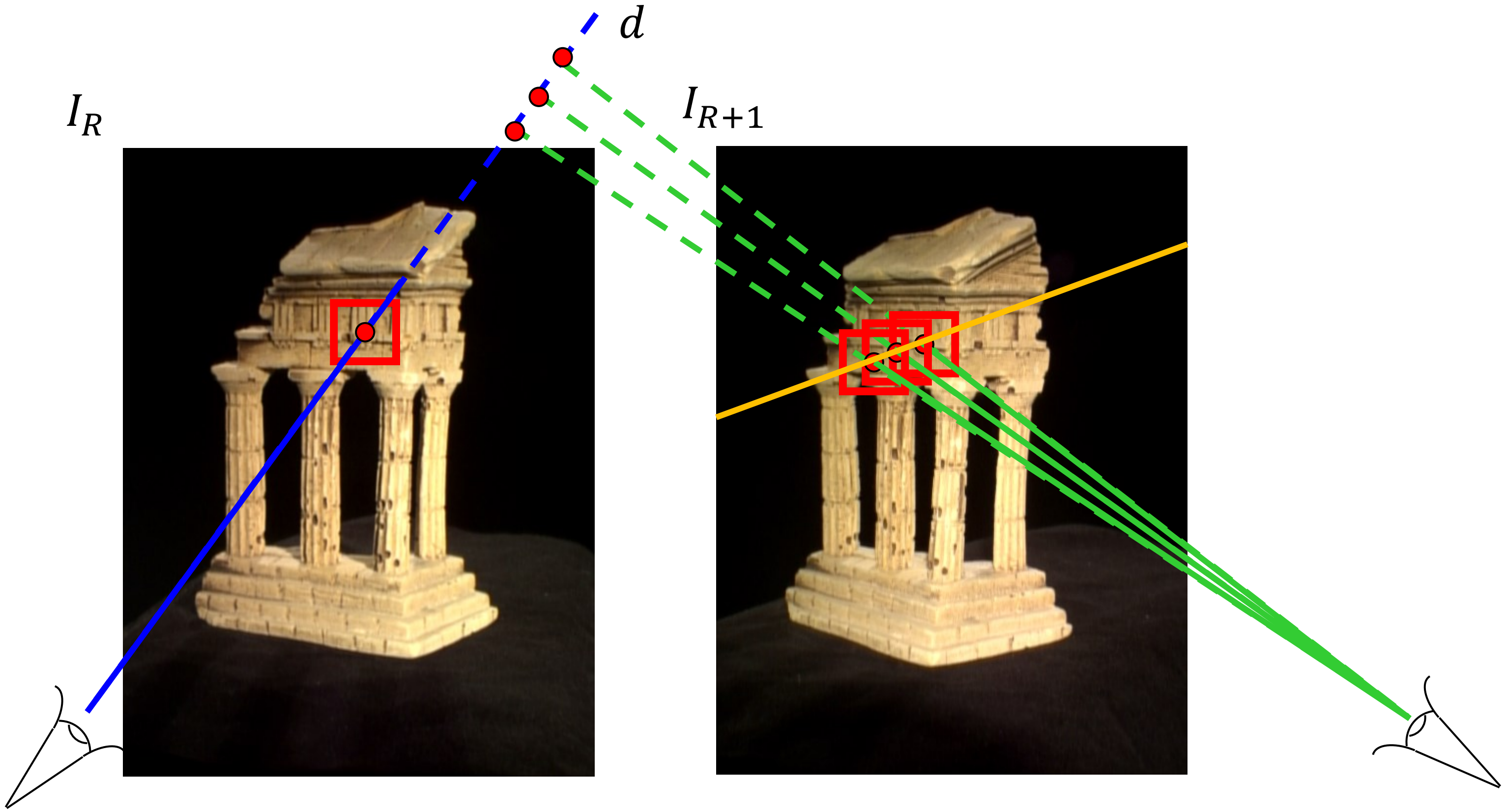
- Refine the *depth map as a* whole by enforcing smoothness. This process is called *regularization*



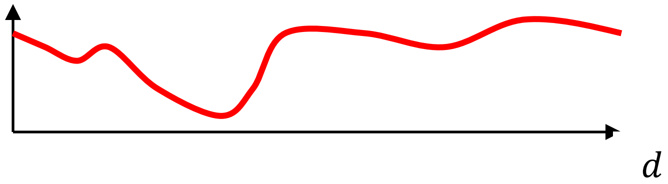
Solution: Aggregated Photometric Error

Set the first image as reference and estimate depth at each pixel by minimizing the Aggregated Photometric Error in all subsequent frames

Solution: Aggregated Photometric Error

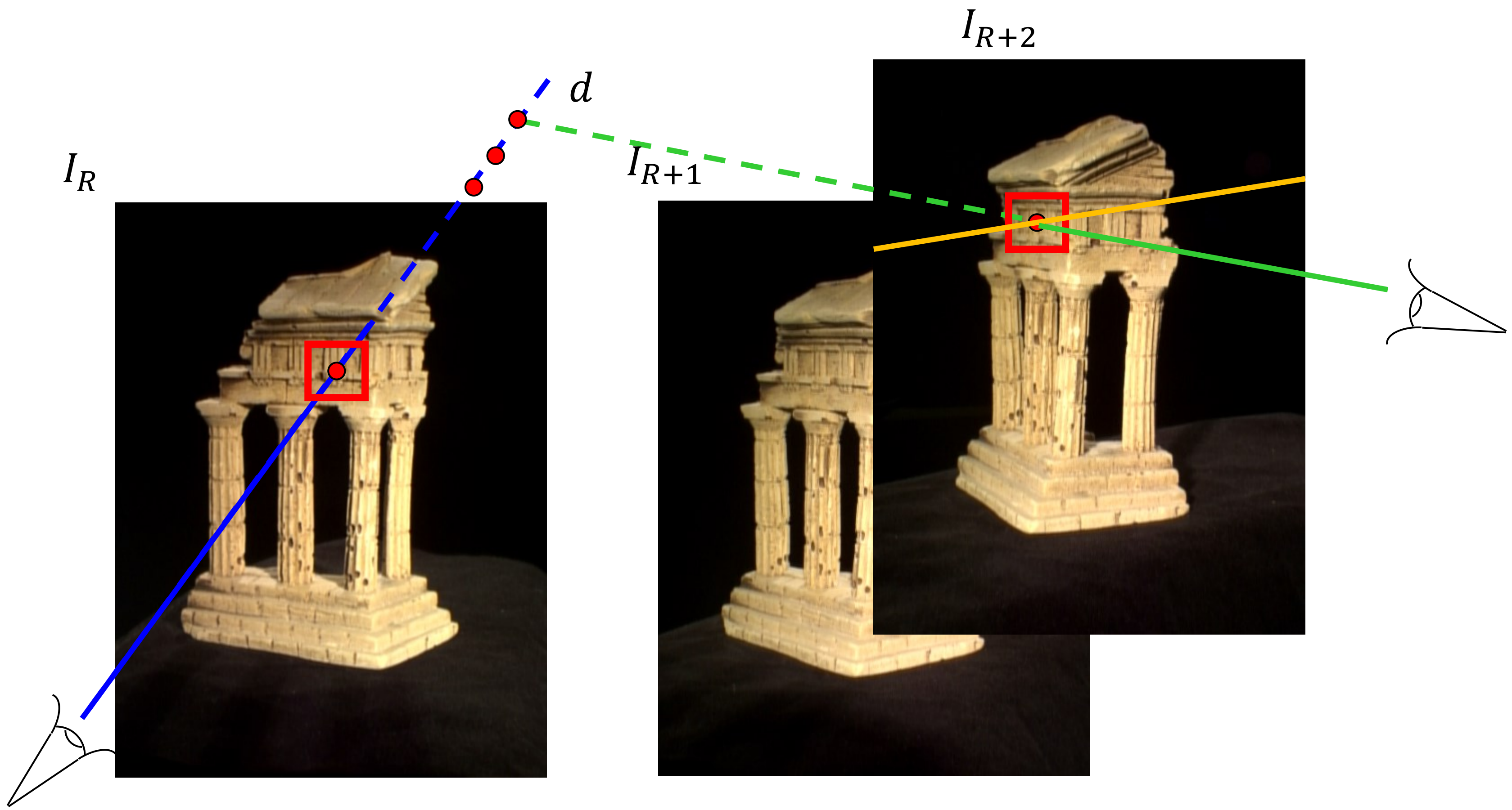


Photometric error: $\rho\big(I_R(u,v) - I_{R+1}(u'(d), v'(d))\big)$

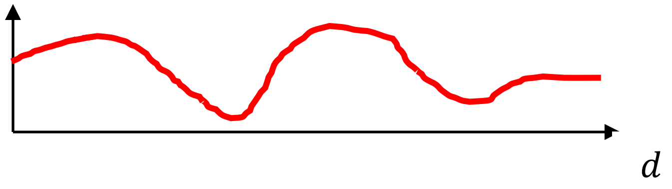


This error term is computed between the reference image and each subsequent frame. The sum of these error terms is called Aggregated Photometric Error (see next slide)

Solution: Aggregated Photometric Error

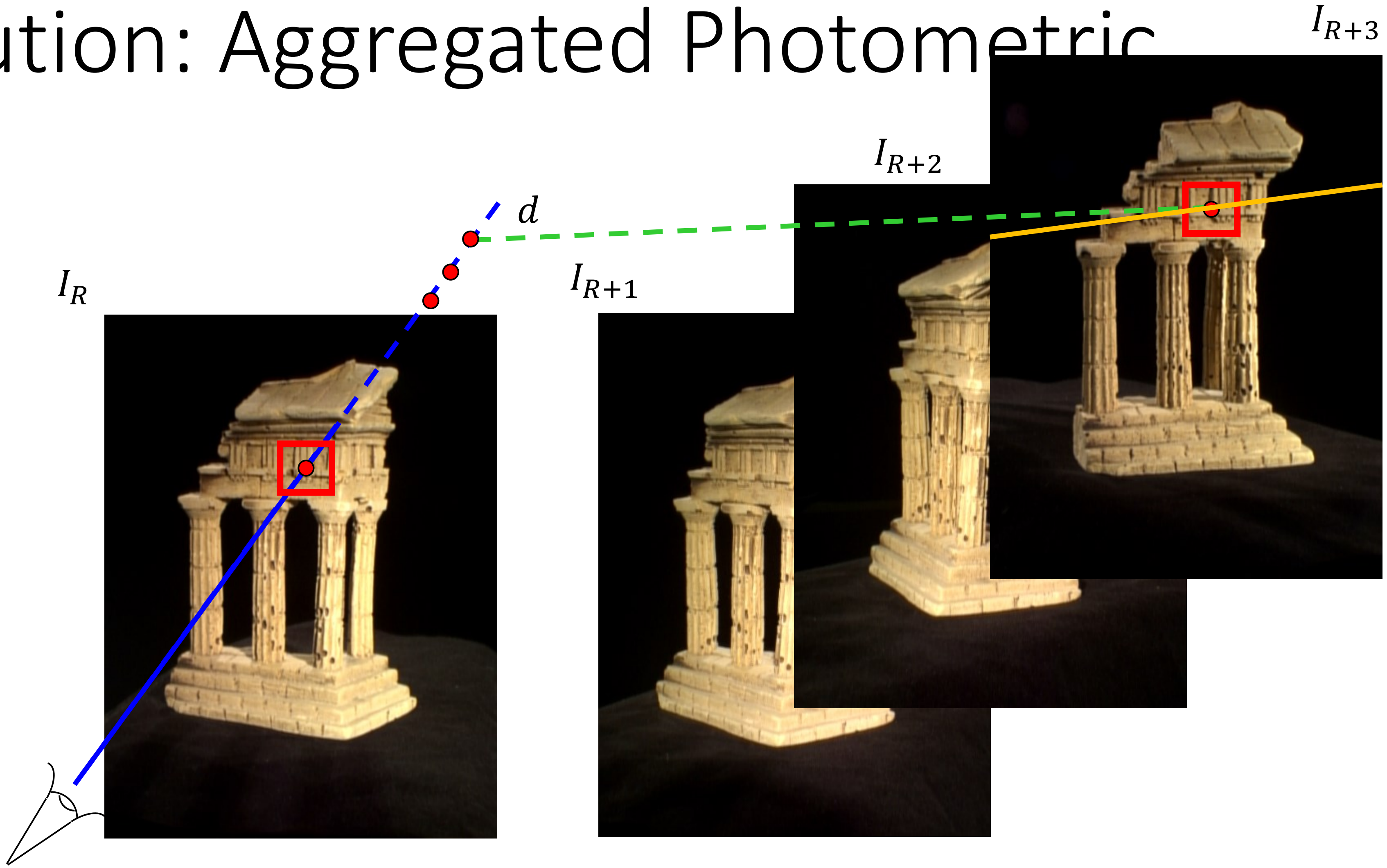


Photometric error: $\rho\left(I_R(u,v) - I_{R+2}(u'(d), v'(d))\right)$

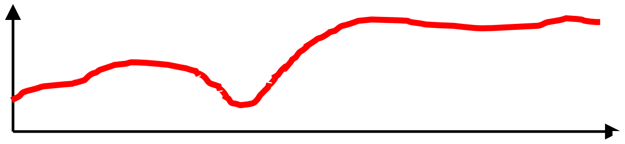


This error term is computed between the reference image and each subsequent frame. The sum of these error terms is called Aggregated Photometric Error (see next slide)

Solution: Aggregated Photometric



Photometric error: $\rho\big(I_R(u,v) - I_{R+3}(u'(d), v'(d))\big)$



This error term is computed between the reference image and each subsequent frame. The sum of these error terms is called Aggregated Photometric Error (see next slide)

Disparity Space Image (DSI)

- For a given image point (u, v) and for discrete depth hypotheses d, the **Aggregated Photometric Error** $C(u, v, d)$ with respect to the reference image I_R can be stored in a volumetric 3D grid called the **Disparity Space Image (DSI)**, where each voxel has value:

$$C(u, v, d) = \sum_{k=R+1}^{R+n-1} \rho\big(I_R(u, v) - I_k(u', v', d)\big)$$

  where n is the number of images considered and $I_k(u', v', d)$ is the patch of intensity values in the k-th image centered on the pixel (u', v') corresponding to the patch $I_R(u, v)$ in the reference image I_R and depth hypothesis $d;$ thus, formally:

$$I_k(u', v', d) = I_k\left(\pi\left(T_{k,R}(\pi^{-1}(u, v) \cdot d)\right)\right)$$

  where $T_{k,R}$ is the relative pose between frames R and K

- $\rho(\cdot)$ is the photometric error (SSD) (e.g. L_1, L_2, Tukey, or Huber norm)

Depth estimation

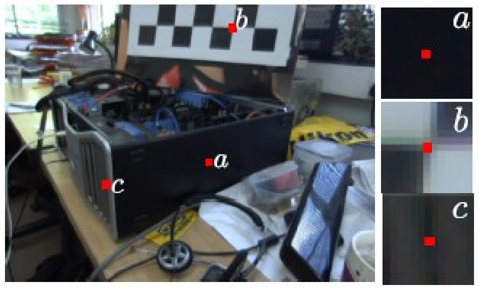
The solution to the depth estimation problem is to find a **function** $\boldsymbol{d(u,v)}$ (called **depth map**) in the DSI that minimizes the **aggregated photometric error:**

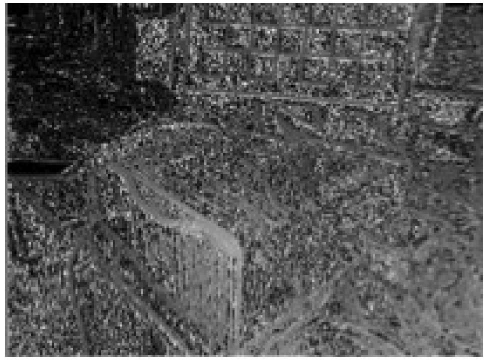
$$depth\,map = d(u,v) = arg\min_{d} C(u,v,d(u,v))$$

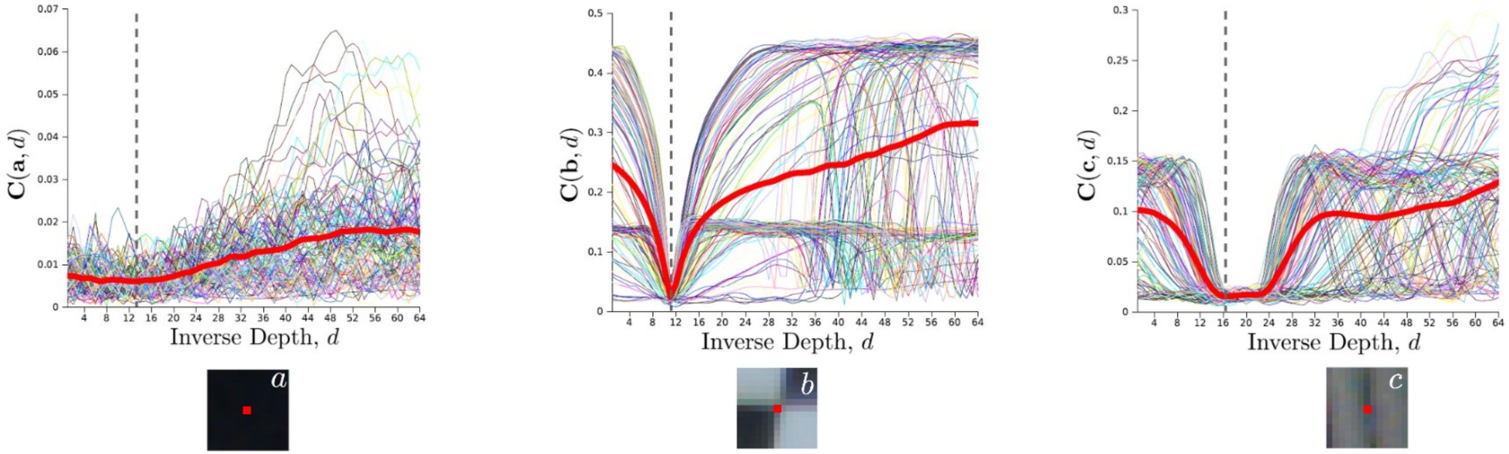
Depth map: each pixel intensity encodes the depth of that pixel. Dark = far, bright = close.

Influence of the patch appearance

- Aggregated photometric error for **flat regions** (a) and **edges parallel to epipolar lines** (c) **show flat valleys (plus noise)**
- For **textured areas** (e.g., corners (b) or blobs), the aggregated photometric error has one **distinctive minimum**
- **Repetitive texture** shows **multiple minima**

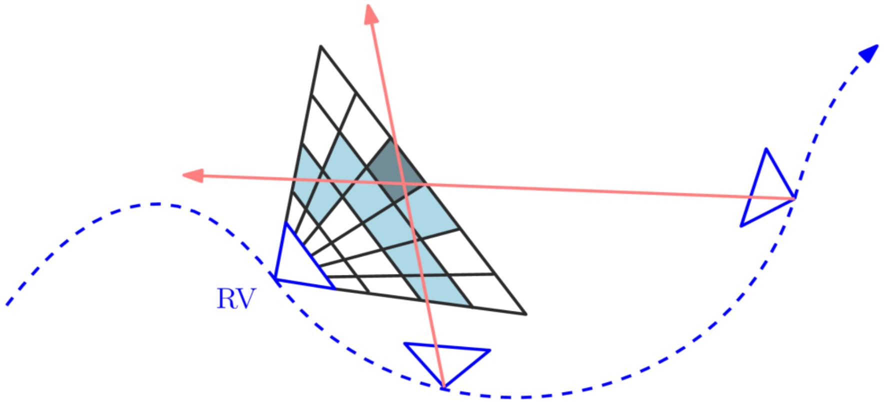






Disparity Space Image (DSI)

- Image resolution: 240×180 pixels
- Number of disparity (depth) levels: 100
- DSI:
  - size: $240 \times 180 \times 100$ voxels; each voxel contains the Aggregated Photometric Error $C(u,v,d)$
  - white = high Aggregated Photometric Error
  - blue = low Aggregated Photometric Error



Non-uniform, projective grid,
centered on the reference frame I_R

Reference image

Disparity Space Image (DSI)

- Image resolution: 240×180 pixels
- Number of disparity (depth) levels: 100
- DSI:
  - size: $240 \times 180 \times 100$ voxels; each voxel contains the Aggregated Photometric Error $C(u,v,d)$
  - white = high Aggregated Photometric Error
  - blue = low Aggregated Photometric Error

DSI

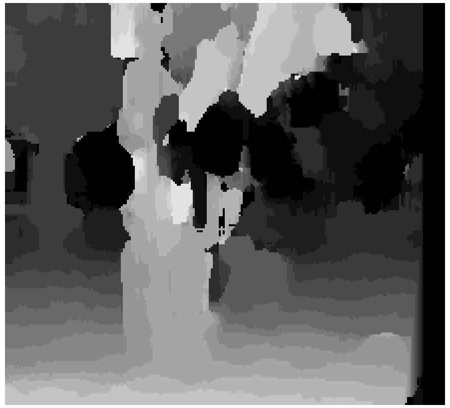
Reference image

Influence of patch size

- Smaller window
  - + More detail
  - • More noise

- Larger window
  - + Smoother disparity maps
  - • Less detail

$W = 3$



$W = 20$

Can we use a patch size of 1×1 pixels?

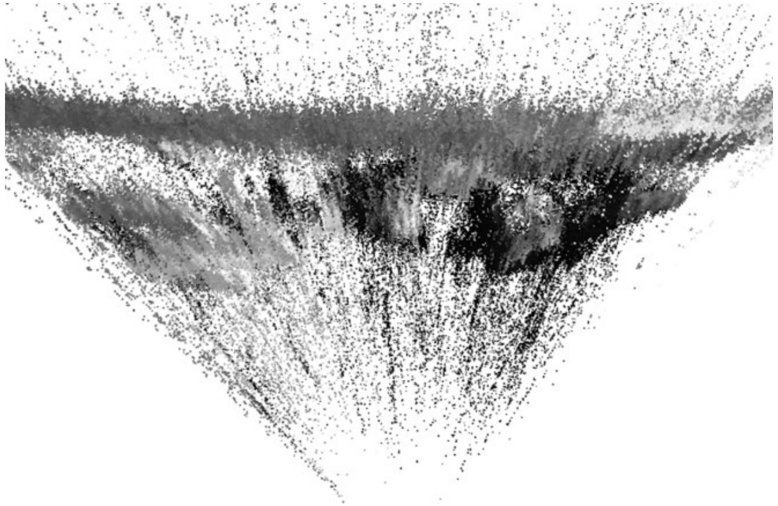
Regularization

To penalize wrong reconstruction due to image noise and ambiguous texture, we add a smoothing term (called regularization term) to the optimization:

$$d(u,v) = arg\min_{d} C(u,v,d(u,v)) \quad \text{(local methods)}$$

subject to

Piecewise smooth (global methods)



First reconstruction via local methods

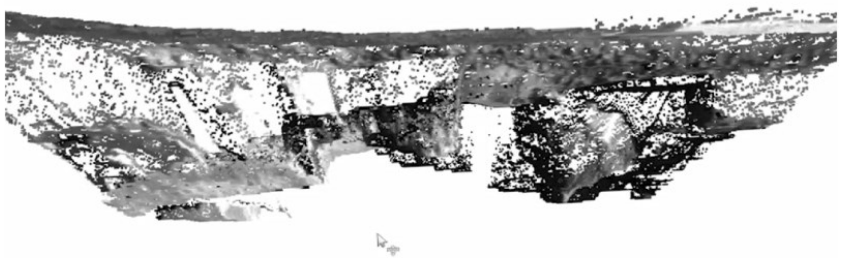
Regularization

To penalize wrong reconstruction due to image noise and ambiguous texture, we add a smoothing term (called regularization term) to the optimization:

$$d(u,v) = arg\min_{d} C(u,v,d(u,v)) \quad \text{(local methods)}$$

subject to

Piecewise smooth (global methods)



After applying global methods

Regularization

- Formulated in terms of energy minimization
- The objective is to find a **surface** $d(u, v)$ that minimizes a global energy functional:

$$E(d) = \underbrace{E_d(d)}_{\text{Data term}} + \underbrace{\lambda E_s(d)}_{\text{Regularization term (i.e., smoothing)}}$$

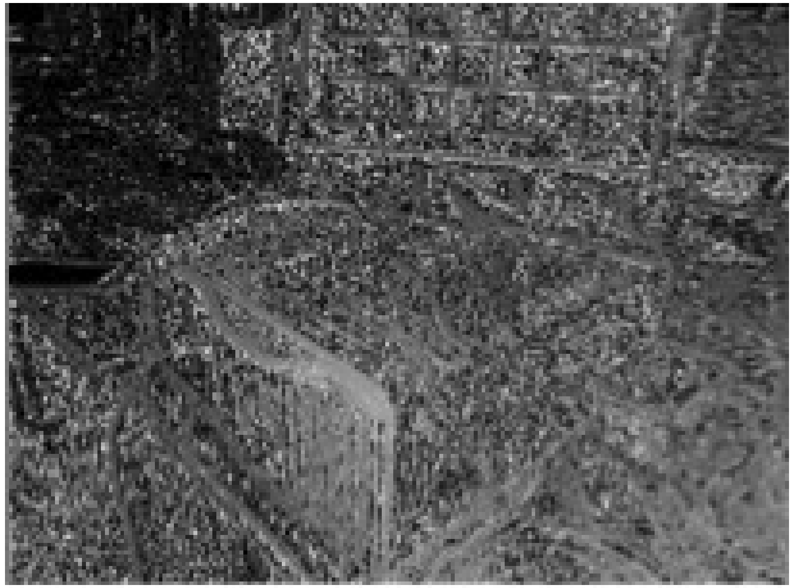
Data term: $E_d(d) = C(u, v, d(u, v))$

Regularization term: $E_s(d) = \sum_{(u,v)} \left(\frac{\partial d}{\partial u}\right)^2 + \left(\frac{\partial d}{\partial v}\right)^2$

where λ controls the tradeoff between data and regularization. What happens as λ increases?

Regularized depth maps

The regularization term $E_s(d)$ basically fills the holes: it **smooths the depth map** by softing discontinuities



Final depth image for increasing λ

Regularized depth maps

The regularization term $E_s(d)$ basically fills the holes: it **smooths the depth map** by softing discontinuities

Final depth image for increasing λ

Regularized depth maps

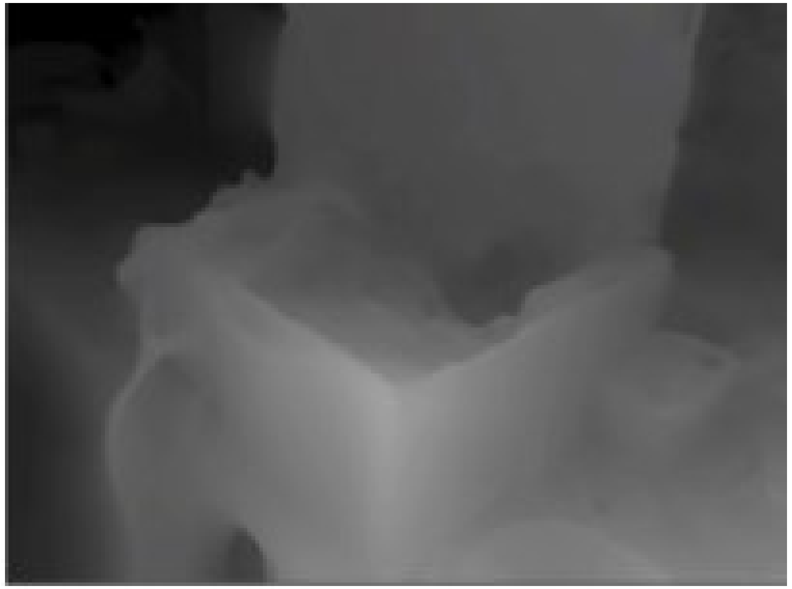
The regularization term $E_s(d)$ basically fills the holes: it **smooths the depth map** by softing discontinuities

Final depth image for increasing λ

Regularized depth maps

The regularization term $E_s(d)$ basically fills the holes: it **smooths the depth map** by softing discontinuities

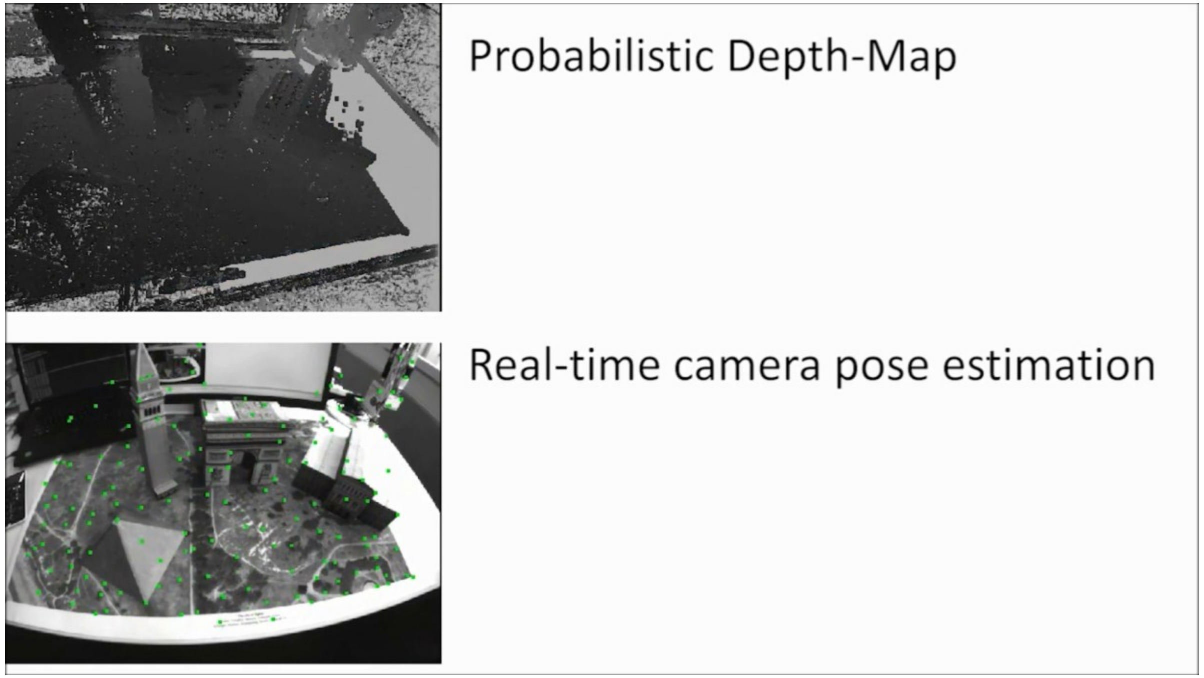
How can we deal with depth discontinuities between separate objects?

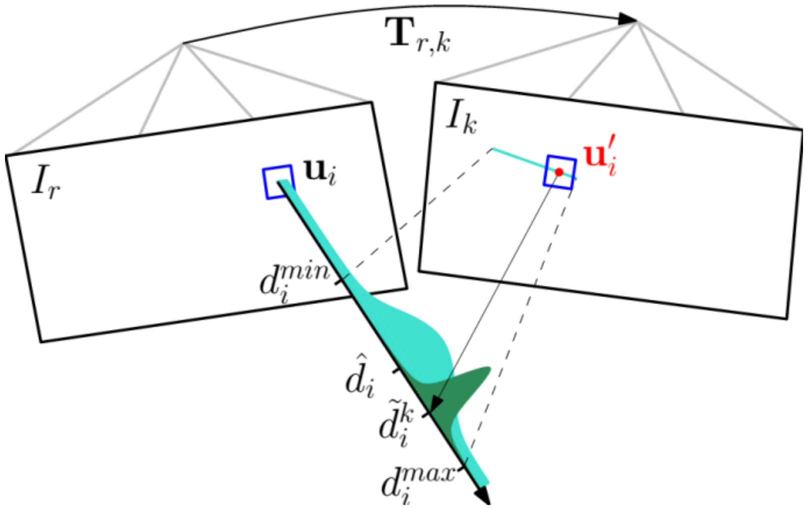


Final depth image for increasing λ

Probabilistic Monocular Dense Reconstruction

- Estimate depth uncertainty
- **Regularize only 3D points** with **low depth uncertainty** (does not fill holes if present, which is good for robotic applications)

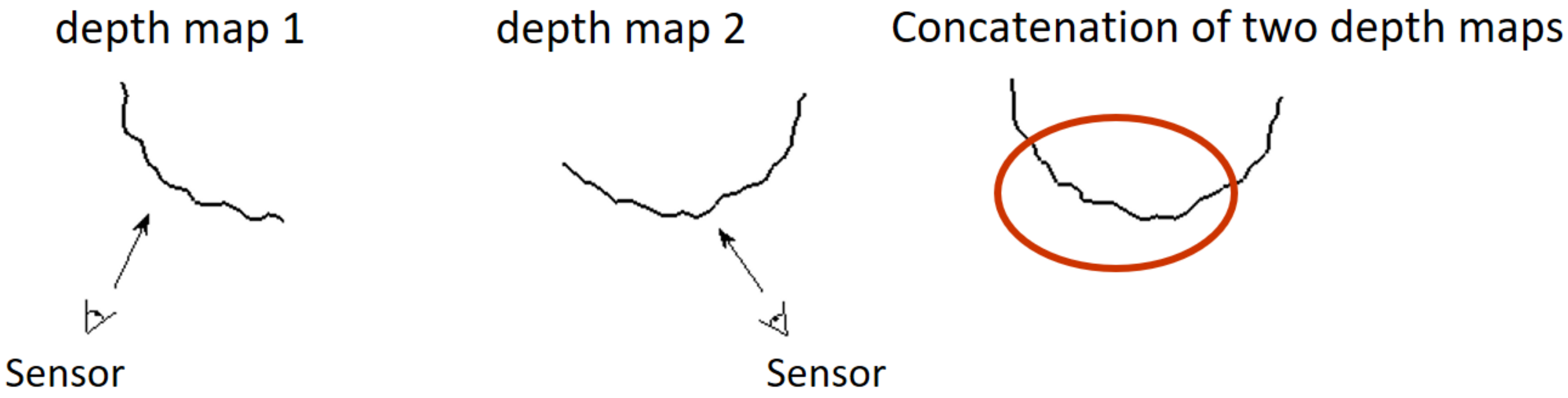




Pizzoli, Forster, Scaramuzza, *REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time*, IEEE International Conference on Robotics and Automation (ICRA), 2014. PDF. Video. Code.

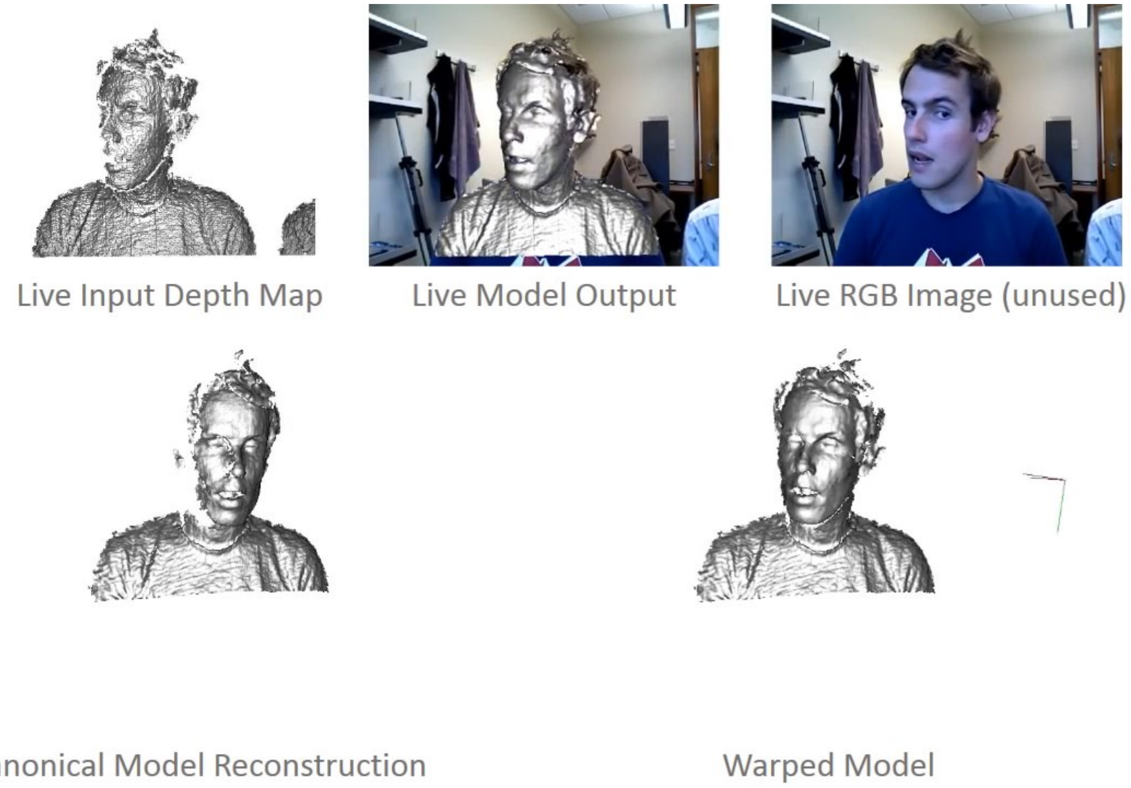
What about large motions?

- When the **distance between the current frame and the reference frame** gets too large (e.g., $> 10\%$ of the average scene depth), then the Aggregated Photometric Error starts to diverge due to the **large view point changes**
- Solution: **create a new reference frame** (keyframe) and start a new depth map computation



What about dynamic objects?

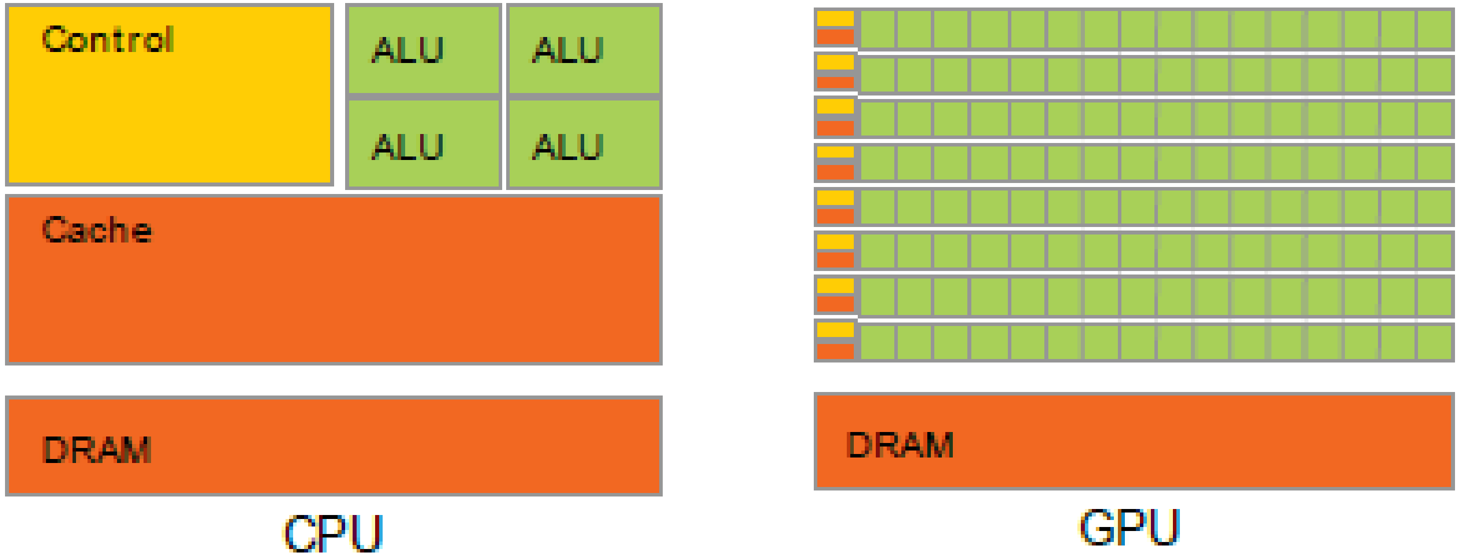
Simultaneous Reconstruction of non-rigid scenes and 6-DOF camera pose tracking using an RGBD camera



Newcombe, Fox, Seitz, *DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time*,
International Conference on Computer Vision and Pattern Recognition (CVPR) 2015, Best Paper Award. PDF.

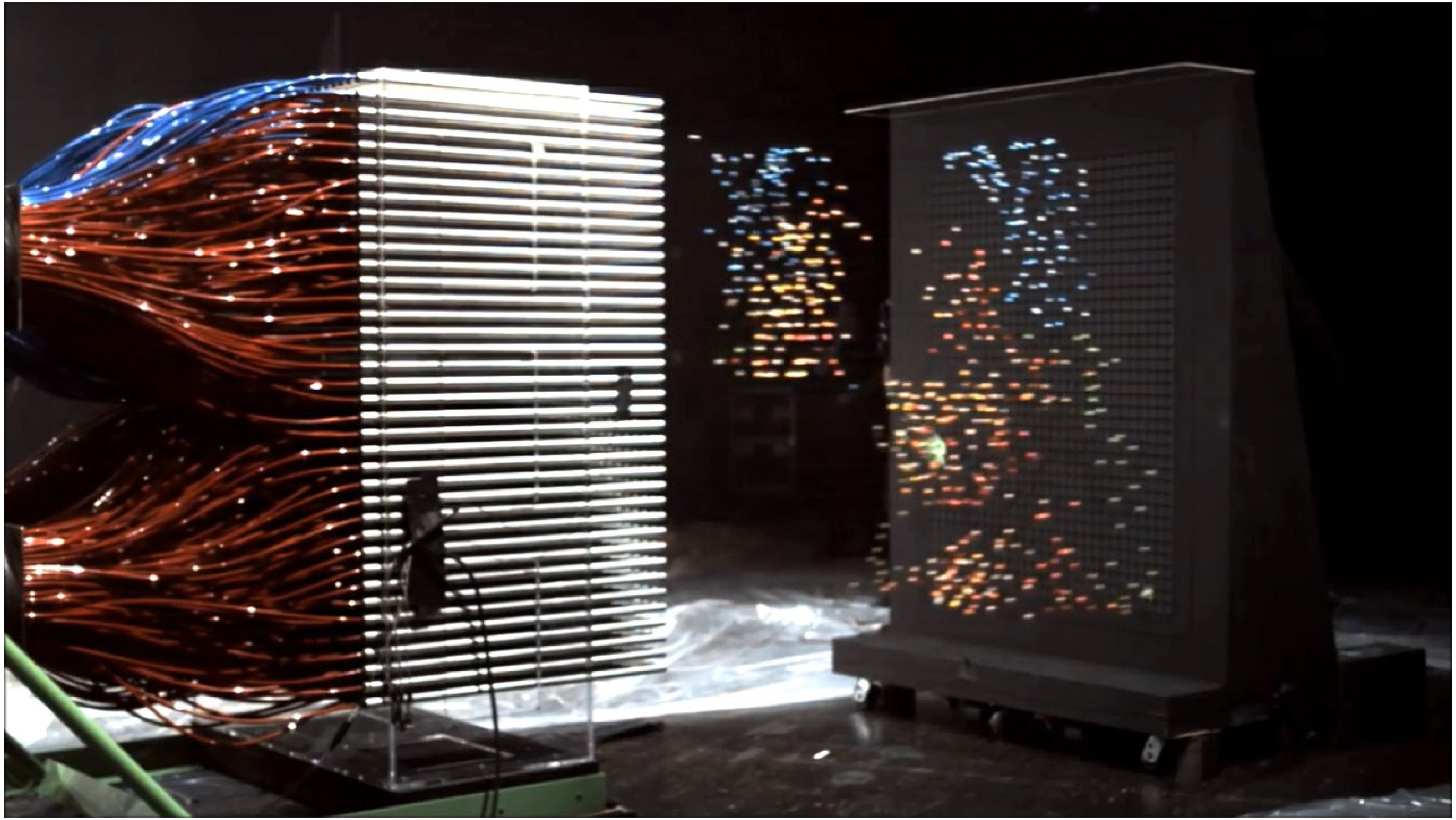
GPU: Graphics Processing Unit

- A CPU is optimized for **serial processing** (i.e., only a few instructions can be executed during the same clock cycle)
- GPU performs calculations **in parallel** on thousands of cores (i.e., thousands of instructions can be executed during the same clock cycle)



ALU: Arithmetic Logic Unit

GPU: Graphics Processing Unit



https://www.youtube.com/watch?v=-P28LKWTzrI

GPU for 3D Dense Reconstruction

- **Image processing**
  - Filtering & Feature extraction (i.e., **convolutions**)
  - Warping (e.g., **epipolar rectification**, **homography**)
- **Multiple-view geometry**
  - Search for **dense correspondences**
    - Pixel-wise operations (SAD, **SSD**, NCC)
    - **Matrix and vector operations** (epipolar geometry)
  - **Aggregated Photometric Error** for multi-view stereo
- **Global optimization**
  - Variational methods (i.e., **regularization** (smoothing))
    - **Parallel, in-place** operations for gradient / divergence computation
- **Deep learning**

Things to remember

- Aggregated Photometric Error
- Disparity Space Image
- Effects of regularization
- Handling discontinuities and large motions
- GPU

Readings

- Chapter: 12.7 of Szeliski's book, 2nd edition

Understanding Check

Are you able to answer the following questions?

- Are you able to describe the multi-view stereo working principle? (aggregated photometric error)
- What are the differences in the behavior of the aggregated photometric error for corners, flat regions, and edges?
- What is the disparity space image (DSI) and how is it built in practice?
- How do we extract the depth from the DSI?
- How do we enforce smoothness (regularization)?
- What happens if we increase lambda (the regularization term)? What if lambda is 0? And if lambda is too big?
- How do we handle depth discontinuities and large motions?
- What are the advantages of GPUs?