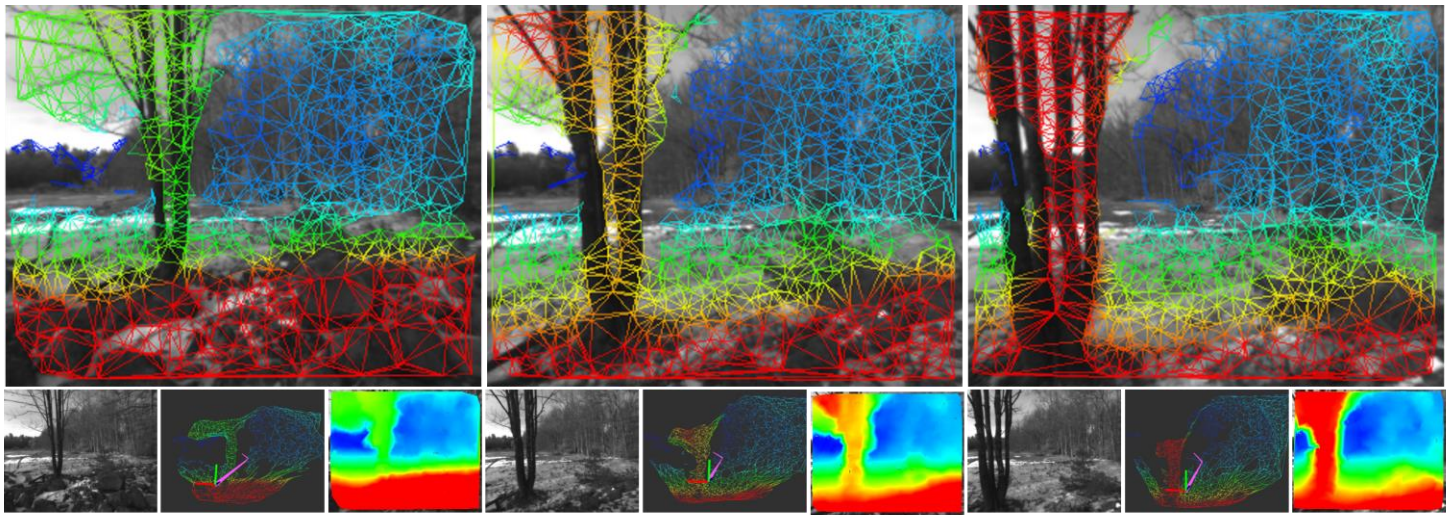
FLaME: Fast Lightweight Mesh Estimation using Variational Smoothing on Delaunay Graphs



W. Nicholas Greene

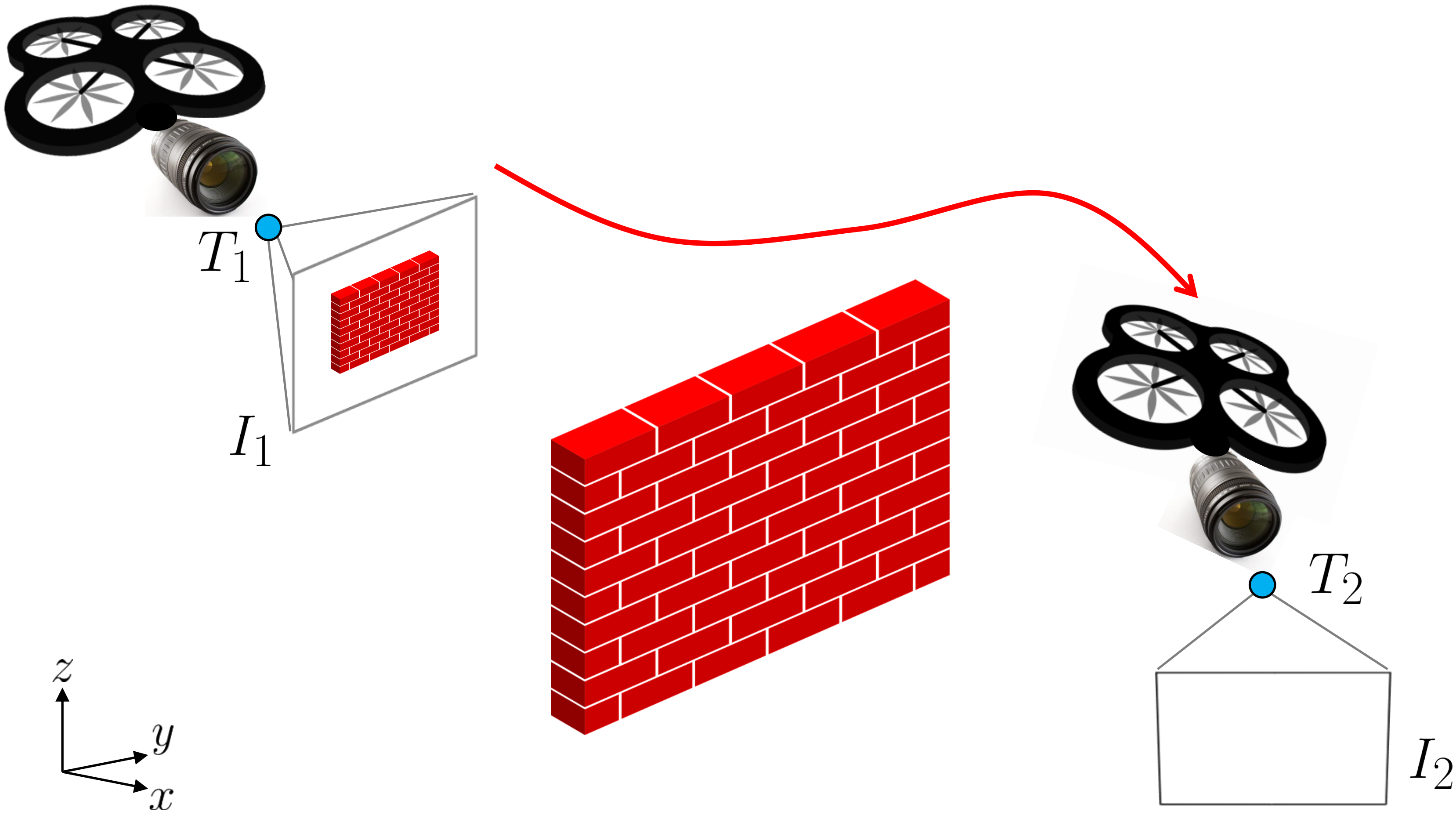
Robust Robotics Group, MIT CSAIL

LPM Workshop IROS 2017

September 28, 2017

with Nicholas Roy

We want Autonomous Vision-Based Navigation



https://commons.wikimedia.org/wiki/File%3AQuadcopter_Drone.png
https://commons.wikimedia.org/wiki/File%3ACanon_90-300mm_camera_lens.jpg

But Dense Monocular SLAM is Expensive

Sparse Methods

Mono-Slam (Davison, ICCV 2007)
PTAM (Klein and Murray, ISMAR 2007)
SVO (Forster et al., ICRA 2014)

Efficiency

Density

https://commons.wikimedia.org/wiki/File%3ANvidia_Titan_XP.jpg
https://shop-media.intel.com/api/v2/helperservice/getimage?url=http://images.icecat.biz/img/gallery/23221218_49.jpg&height=550&width=550
https://www.qualcomm.com/sites/ember/files/styles/optimize/public/component-item/flexible-block/thumb/chip_3.png?itok=XncLtDdQ

But Dense Monocular SLAM is Expensive

Sparse Methods

Mono-Slam (Davison, ICCV 2007)
PTAM (Klein and Murray, ISMAR 2007)
SVO (Forster et al., ICRA 2014)

Semi-Dense Methods

Engel et al. ICCV 2013
LSD-SLAM (Engel et al., ECCV 2014)
Mur-Artal and Tardos (RSS 2015)
Pillai et al. ICRA 2016

Efficiency

Density

https://commons.wikimedia.org/wiki/File%3ANvidia_Titan_XP.jpg
https://shop-media.intel.com/api/v2/helperservice/getimage?url=http://images.icecat.biz/img/gallery/23221218_49.jpg&height=550&width=550
https://www.qualcomm.com/sites/ember/files/styles/optimize/public/component-item/flexible-block/thumb/chip_3.png?itok=XncLtDdQ

But Dense Monocular SLAM is Expensive

Efficiency

Sparse Methods

Mono-Slam (Davison, ICCV 2007)
PTAM (Klein and Murray, ISMAR 2007)
SVO (Forster et al., ICRA 2014)

Semi-Dense Methods

Engel et al. ICCV 2013
LSD-SLAM (Engel et al., ECCV 2014)
Mur-Artal and Tardos (RSS 2015)
Pillai et al. ICRA 2016

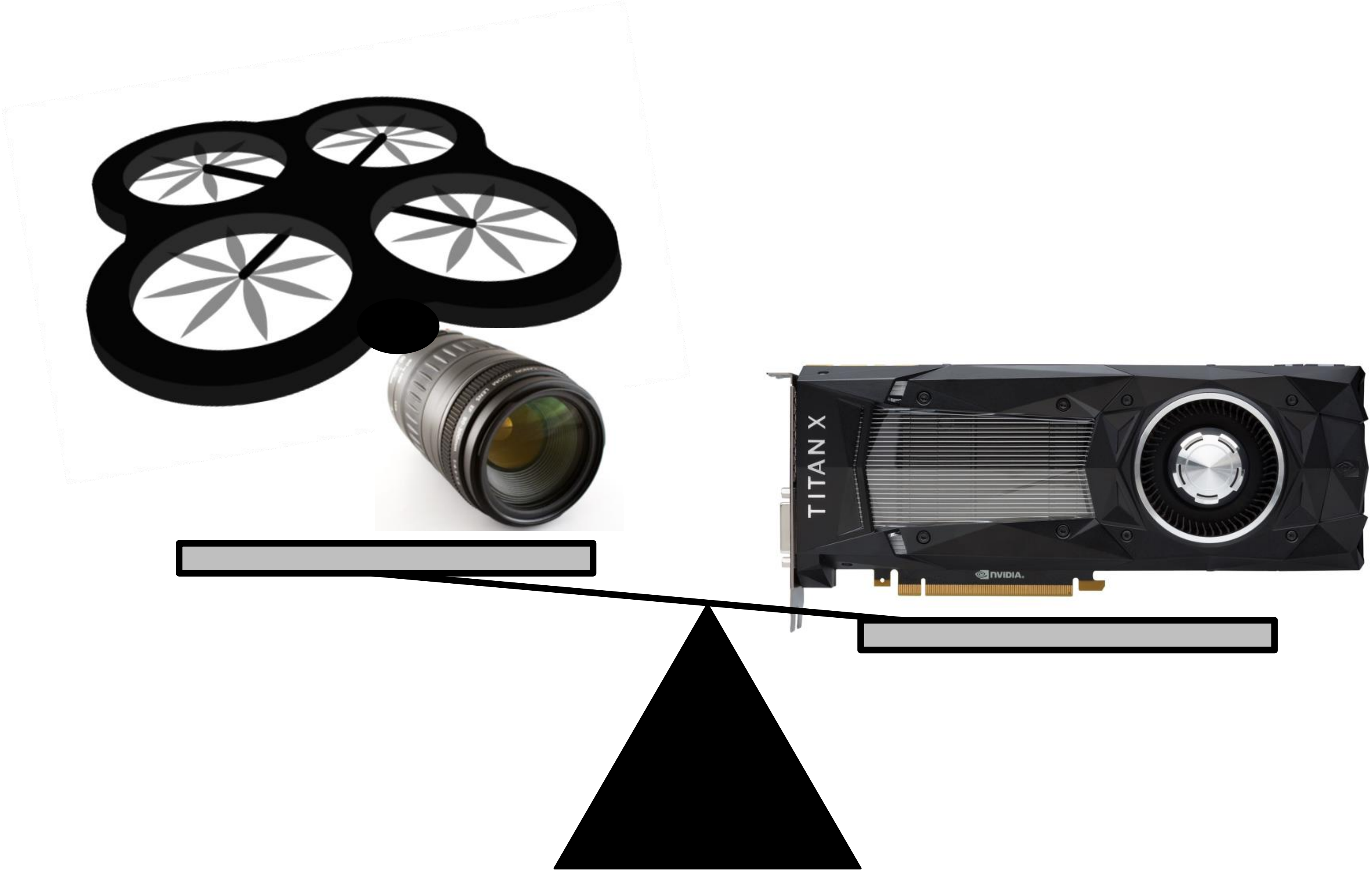
Dense Methods

DTAM (Newcombe et al., ICCV 2011)
Graber et al. (ICCV 2011)
MonoFusion (Pradeep et al., ISMAR 2013)
REMODE (Pizzoli et al., ICRA 2014)

Density

https://commons.wikimedia.org/wiki/File%3ANvidia_Titan_XP.jpg
https://shop-media.intel.com/api/v2/helperservice/getimage?url=http://images.icecat.biz/img/gallery/23221218_49.jpg&height=550&width=550
https://www.qualcomm.com/sites/ember/files/styles/optimize/public/component-item/flexible-block/thumb/chip_3.png?itok=XncLtDdQ

But Dense Monocular SLAM is Expensive



https://commons.wikimedia.org/wiki/File%3ANvidia_Titan_XP.jpg
https://commons.wikimedia.org/wiki/File%3AQuadcopter_Drone.png
https://commons.wikimedia.org/wiki/File%3ACanon_90-300mm_camera_lens.jpg

But Dense Monocular SLAM is Expensive

Sparse Methods

Mono-Slam (Davison, ICCV 2007)
PTAM (Klein and Murray, ISMAR 2007)
SVO (Forster et al., ICRA 2014)

Semi-Dense Methods

Engel et al. ICCV 2013
LSD-SLAM (Engel et al., ECCV 2014)
Mur-Artal and Tardos (RSS 2015)
Pillai et al. ICRA 2016

Dense Methods

DTAM (Newcombe et al., ICCV 2011)
Graber et al. (ICCV 2011)
MonoFusion (Pradeep et al., ISMAR 2013)
REMODE (Pizzoli et al., ICRA 2014)

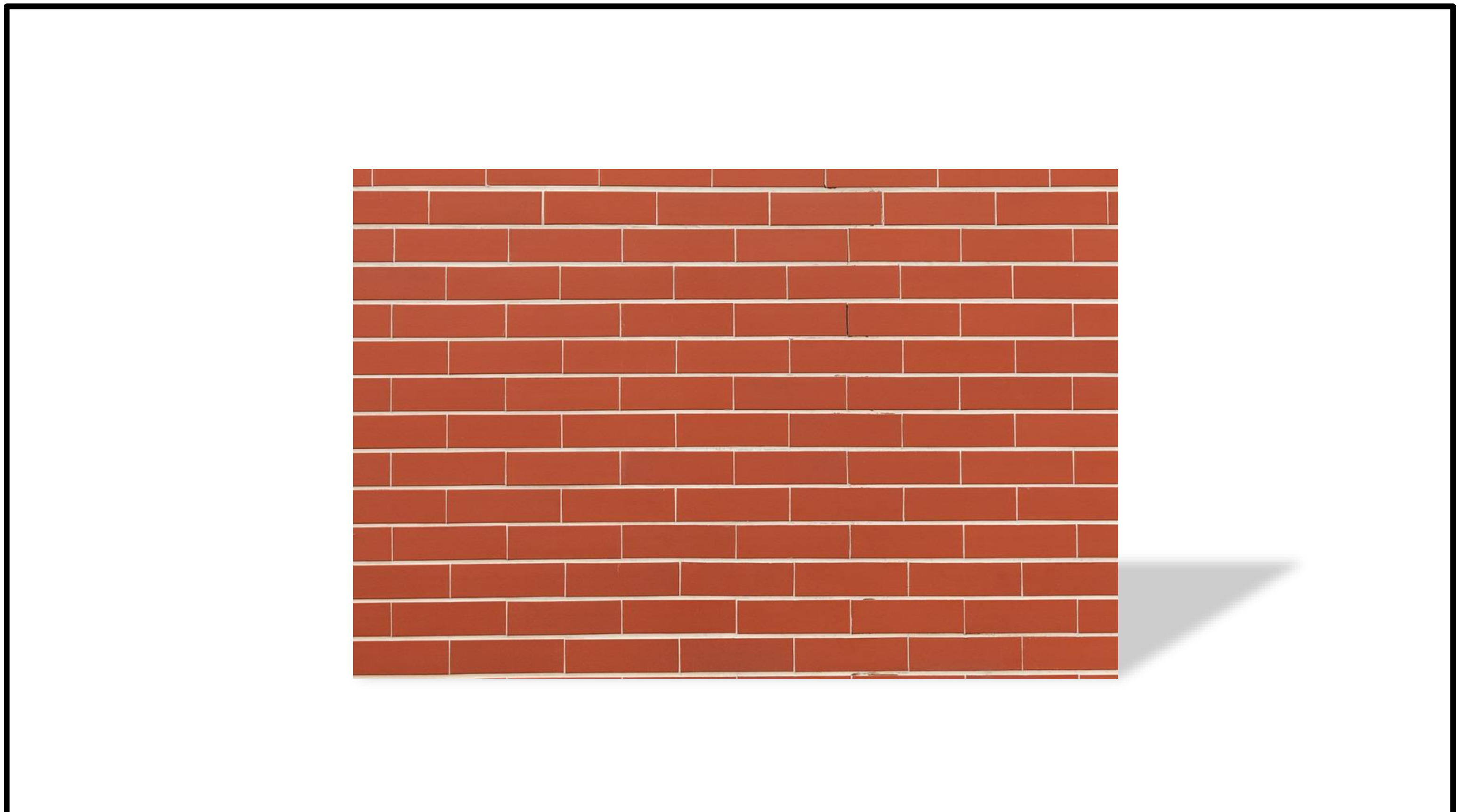
?

Efficiency

Density

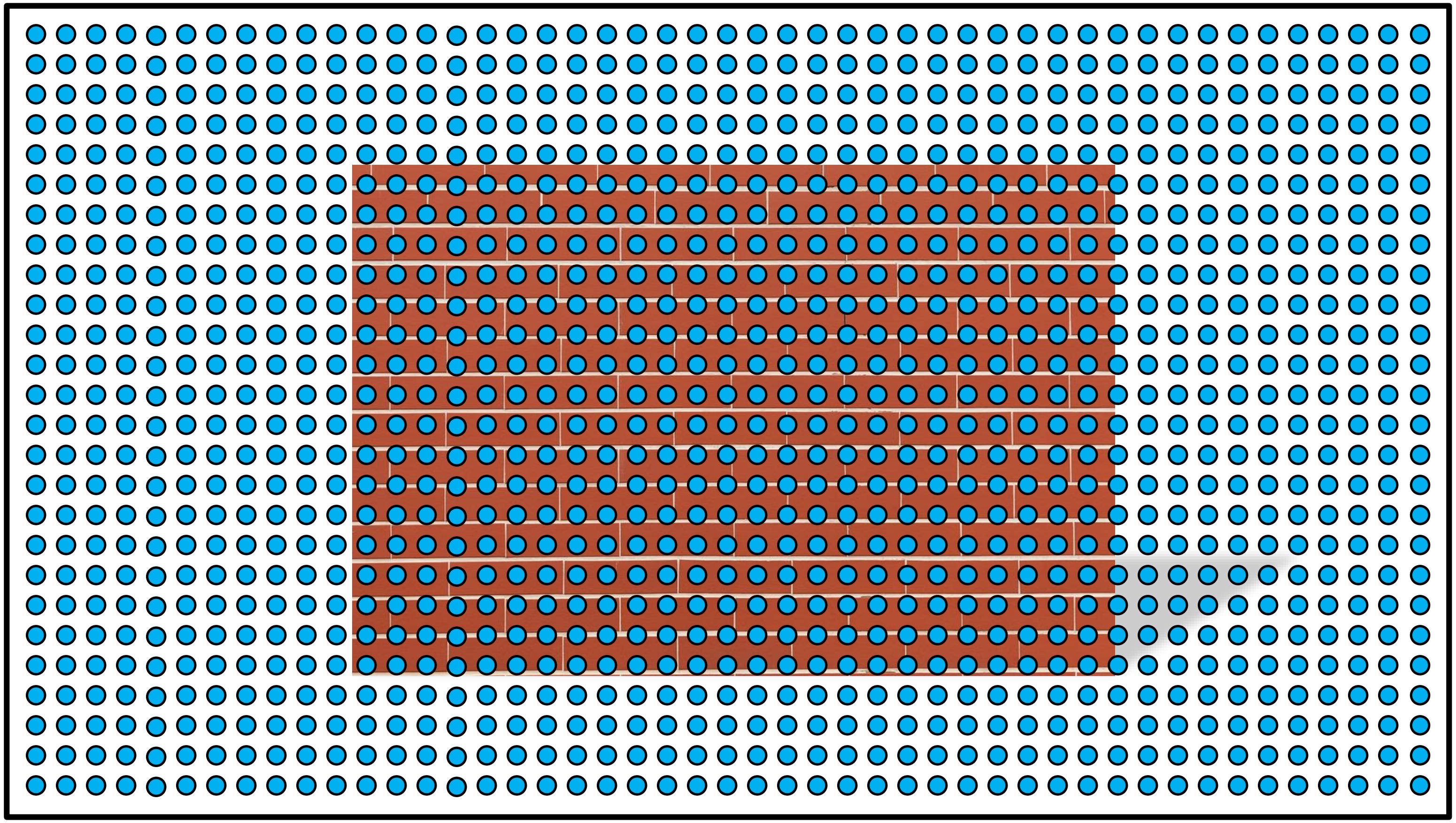
Can we more efficiently estimate dense geometry?

What Would a Dense Method Do?



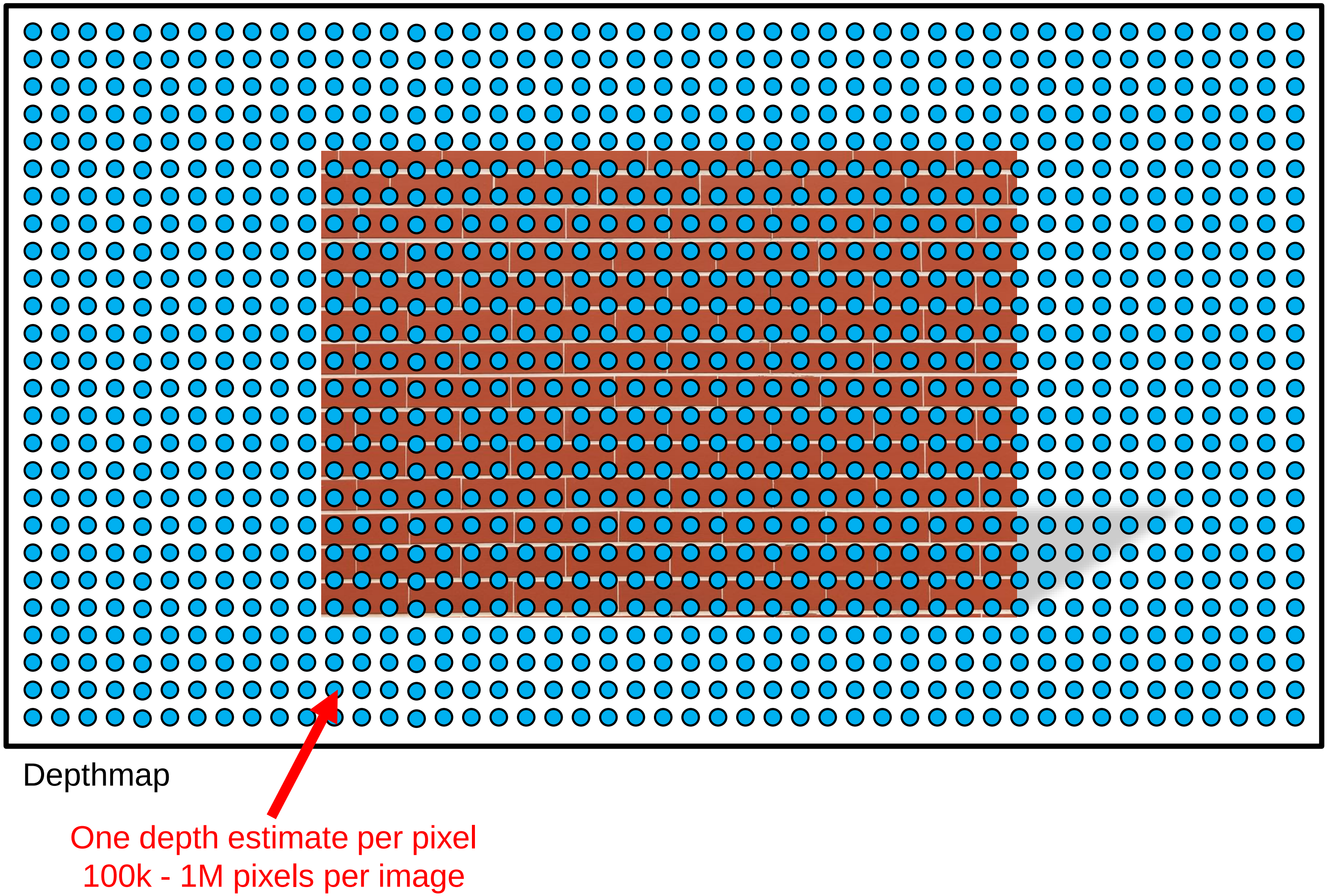
Image

Estimate Depth for Every Pixel

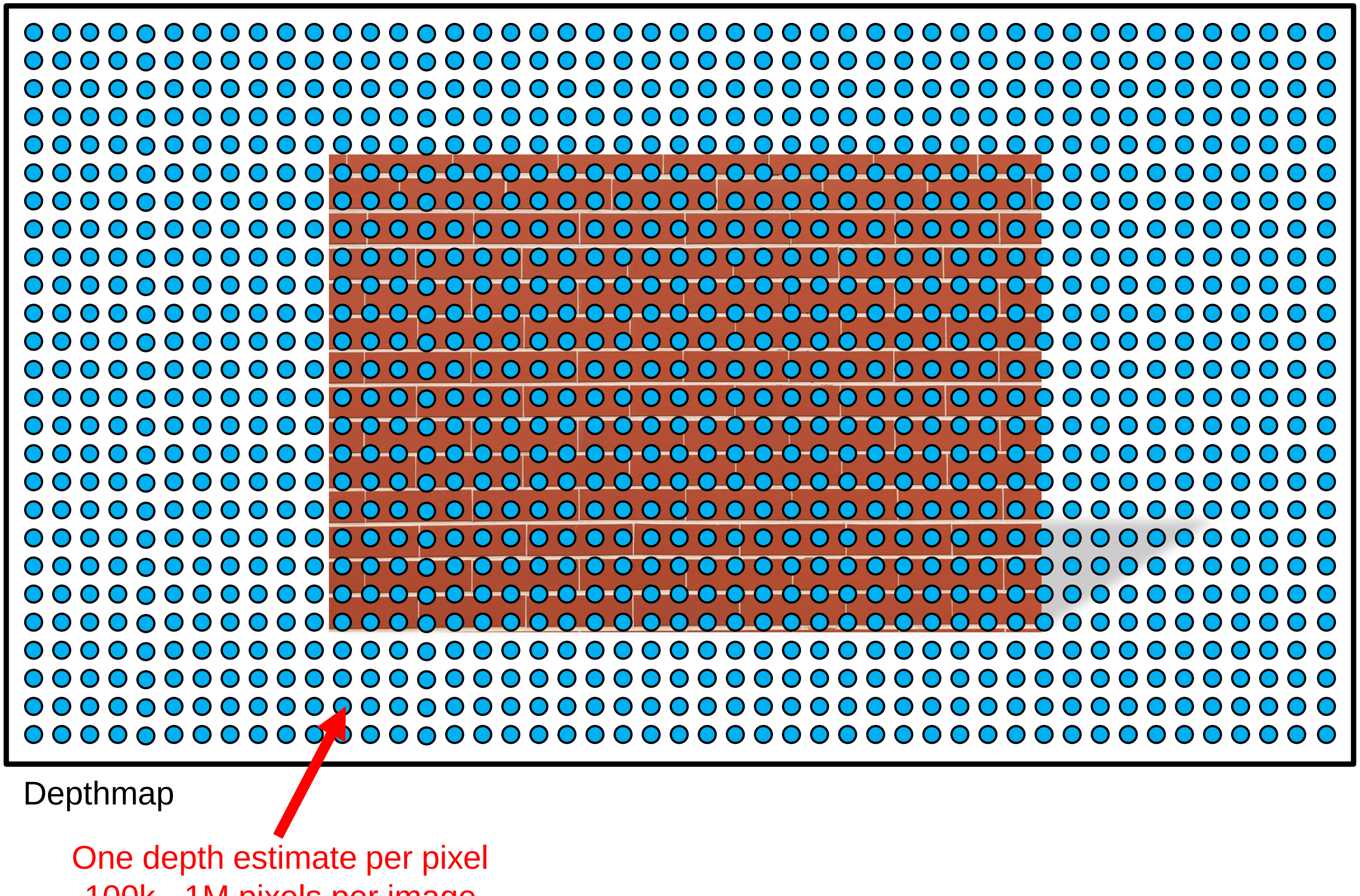


Depthmap

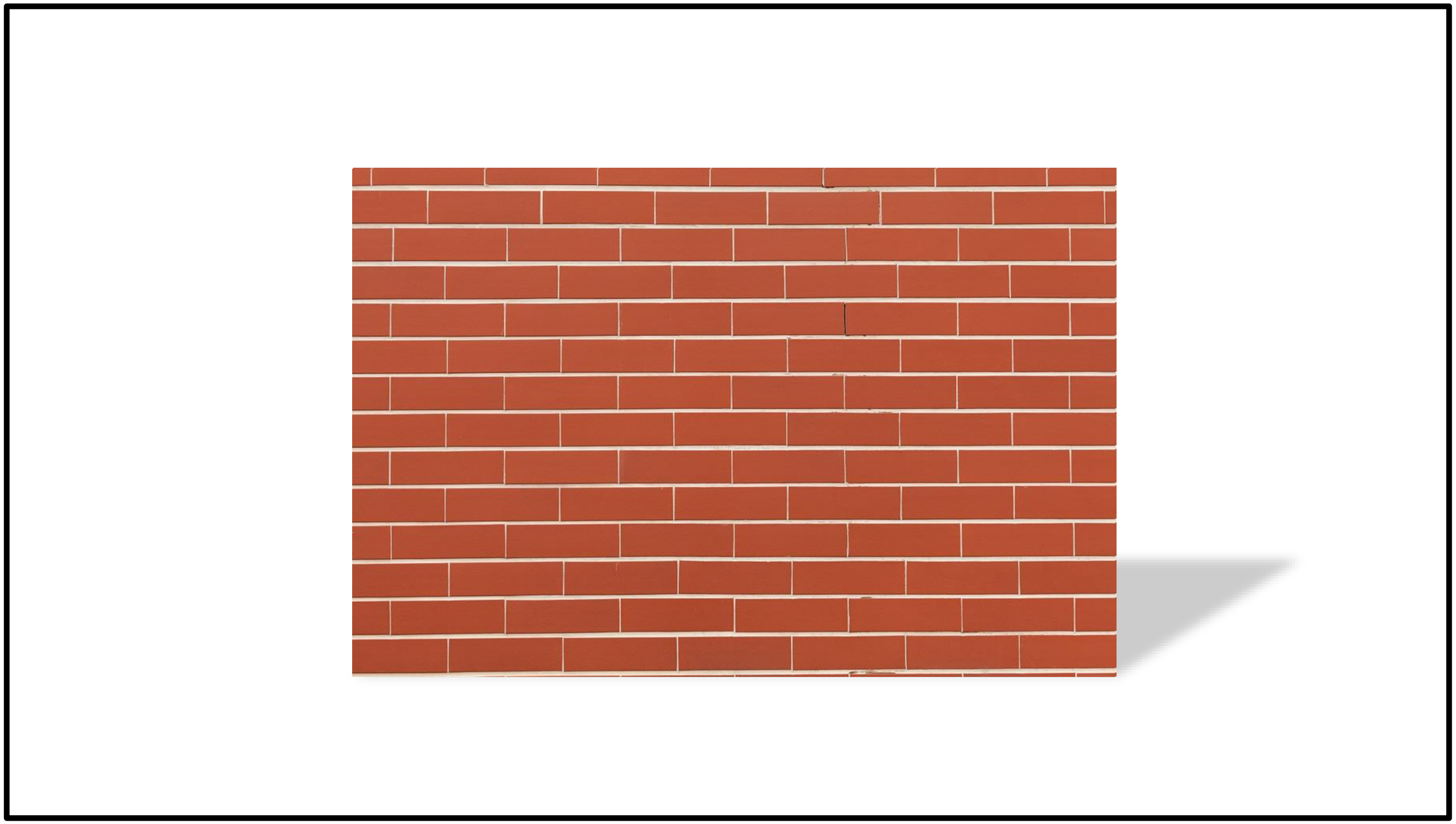
Dense Methods Oversample Geometry



Dense Methods Make Regularization Hard(er)

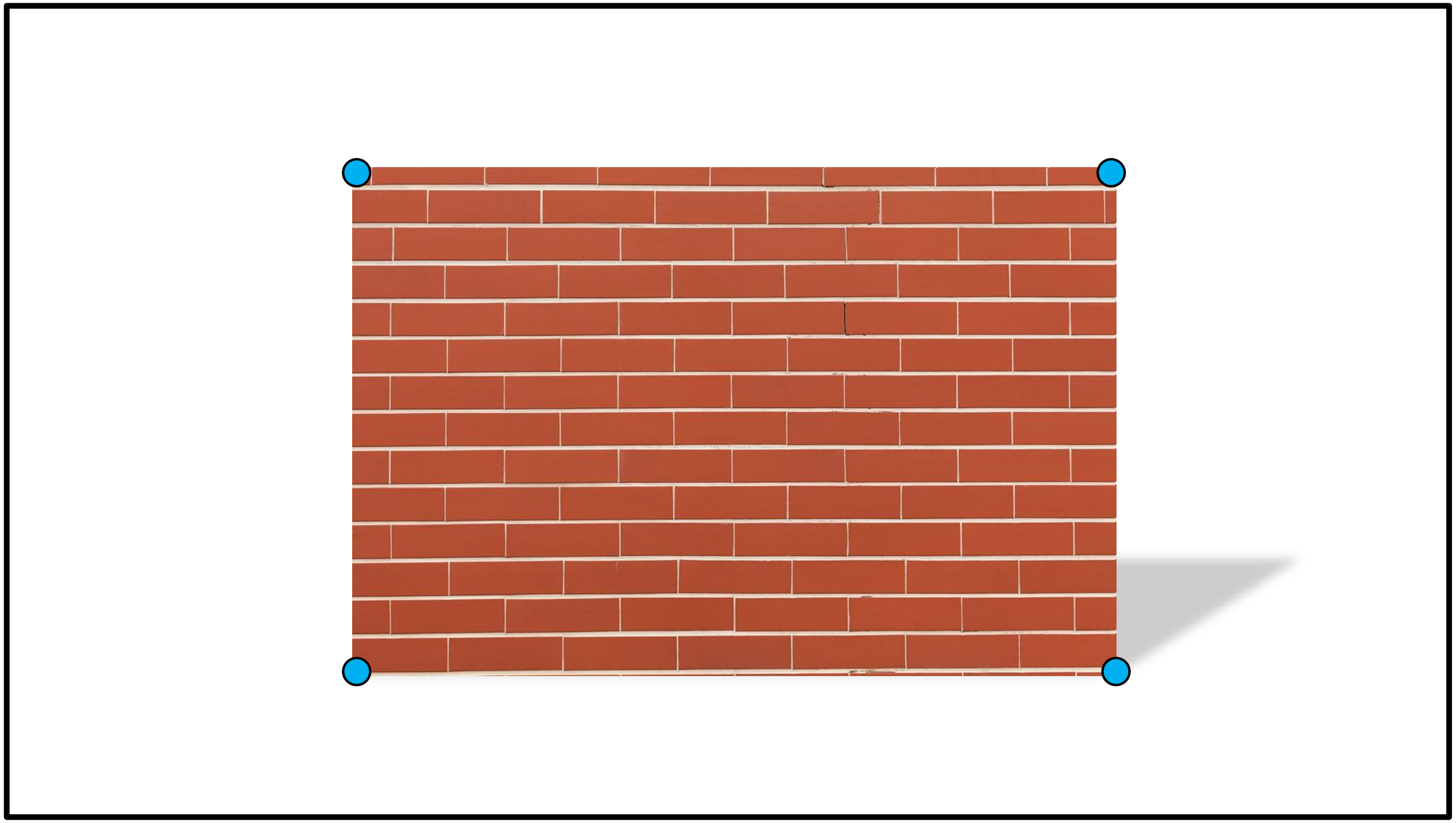


Meshes Encode Geometry with Fewer Depths



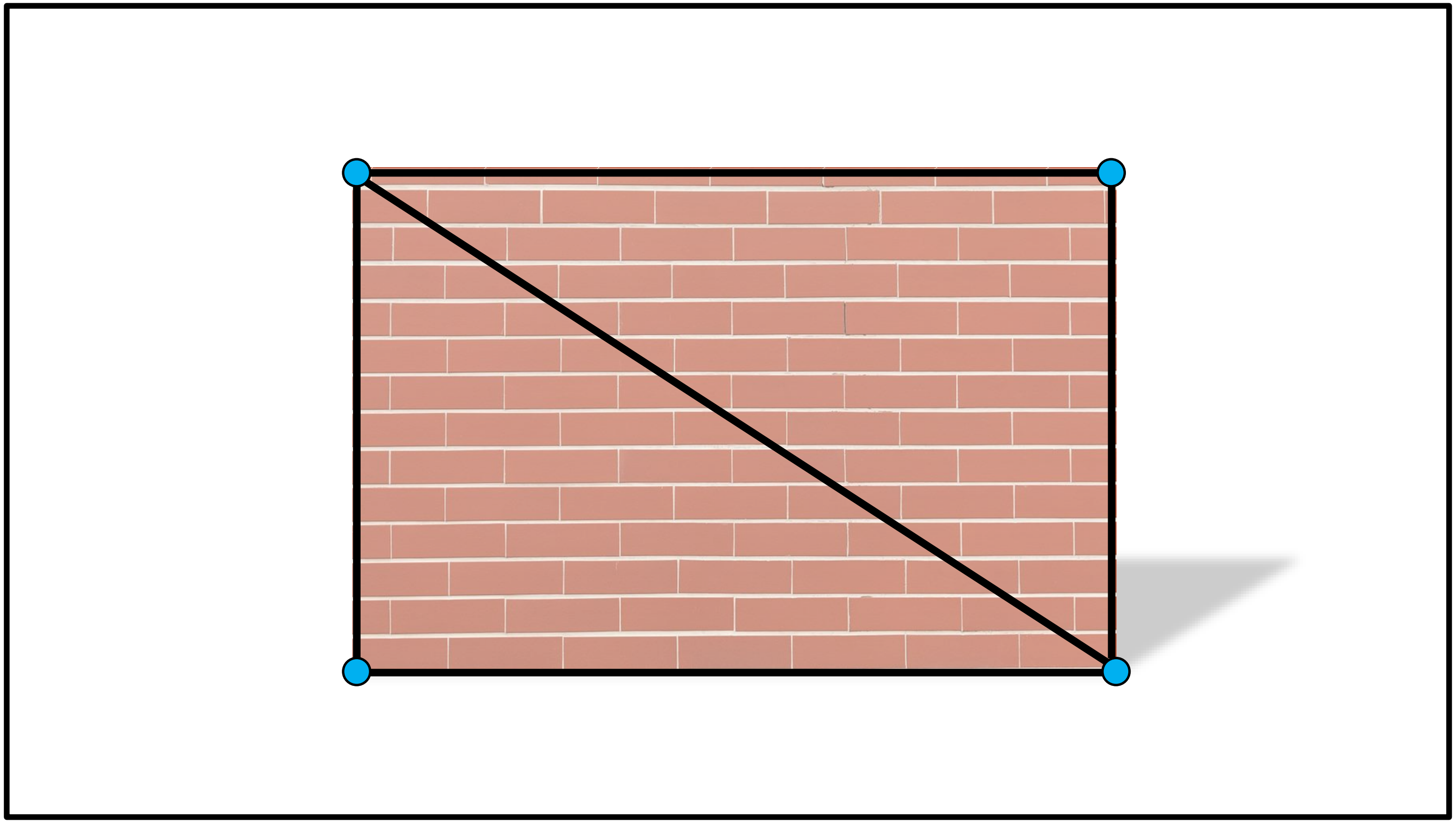
Depthmap

Meshes Encode Geometry with Fewer Depths



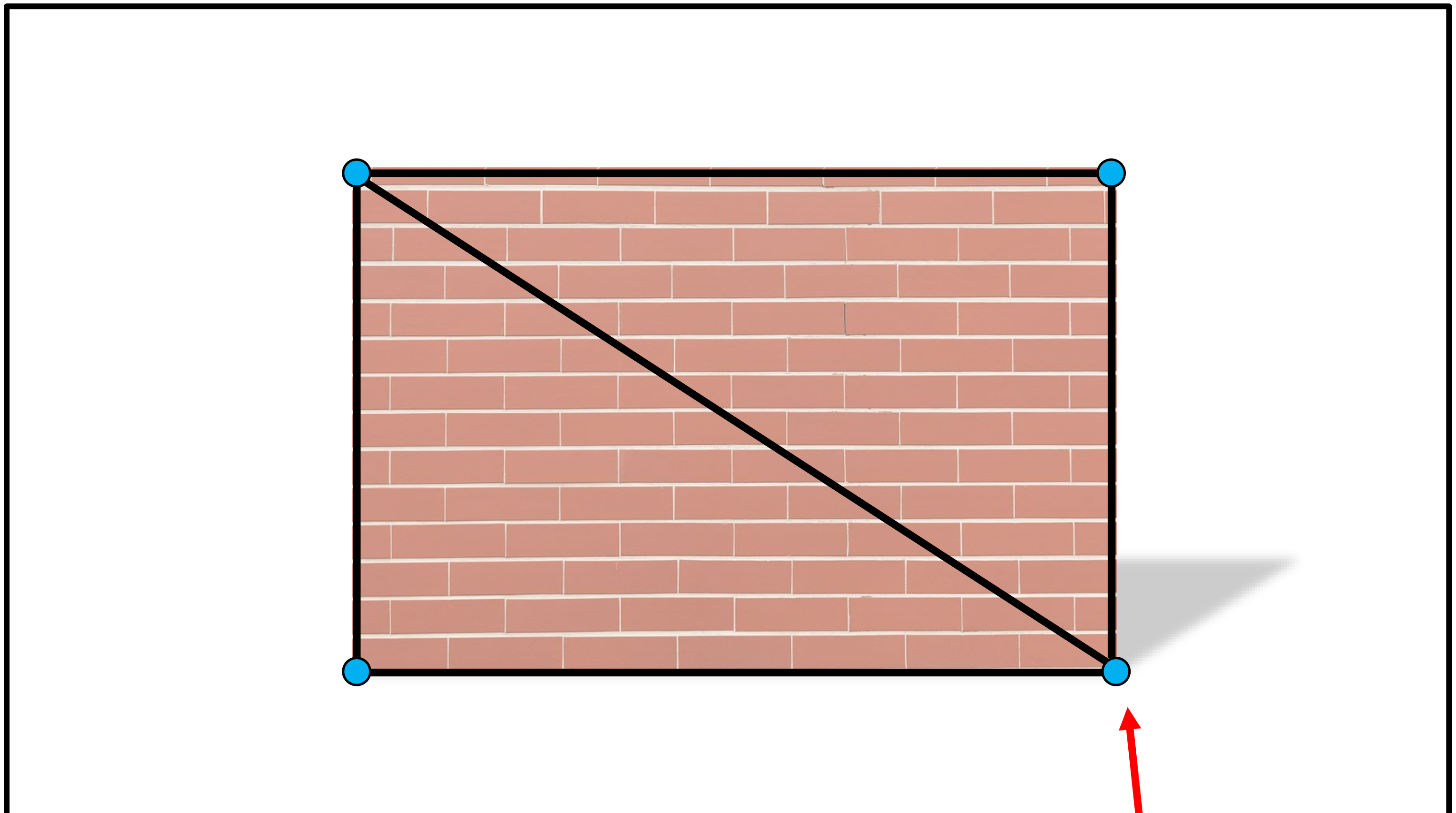
Depthmap

Meshes Encode Geometry with Fewer Depths



Depthmap

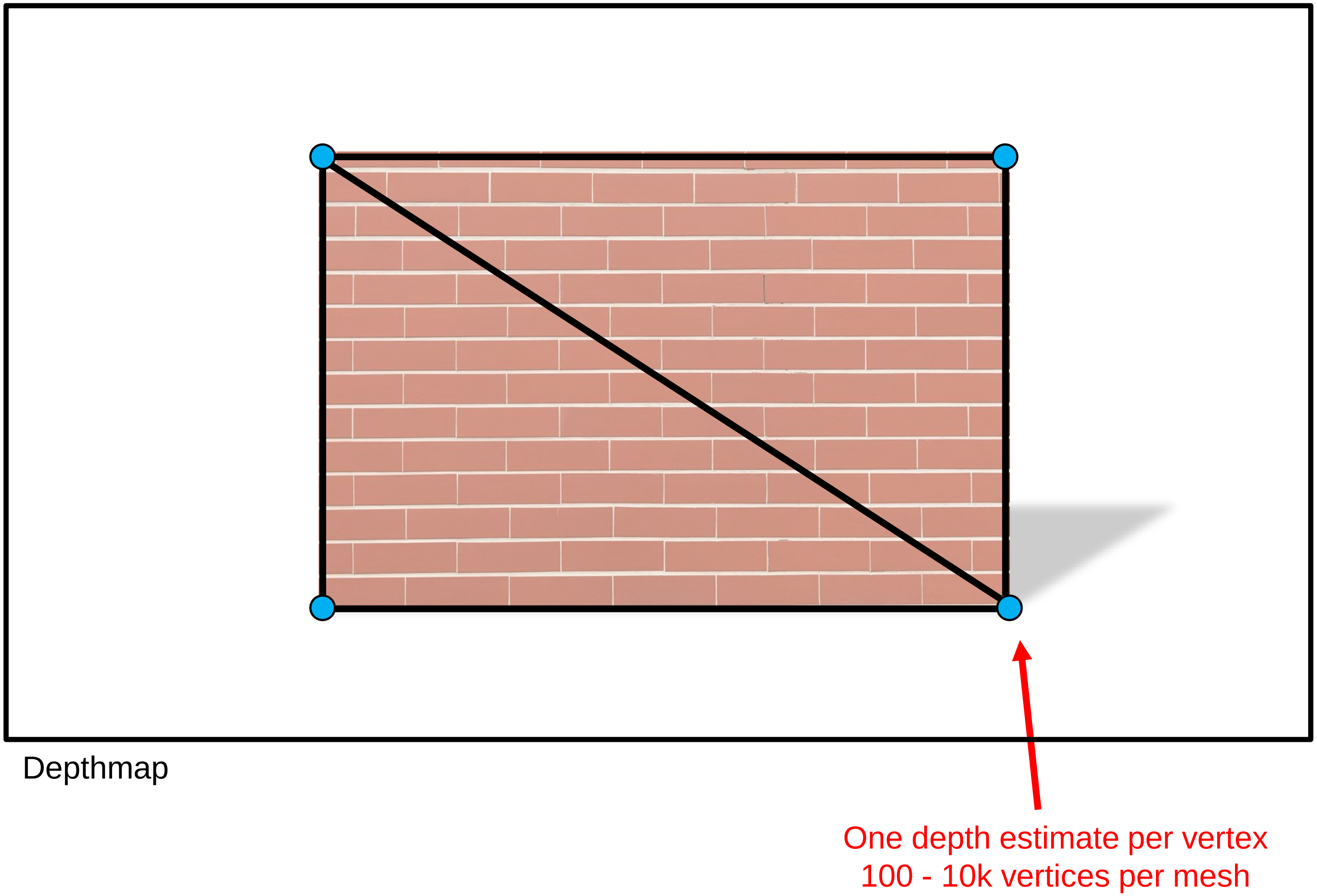
Meshes Encode Geometry with Fewer Depths



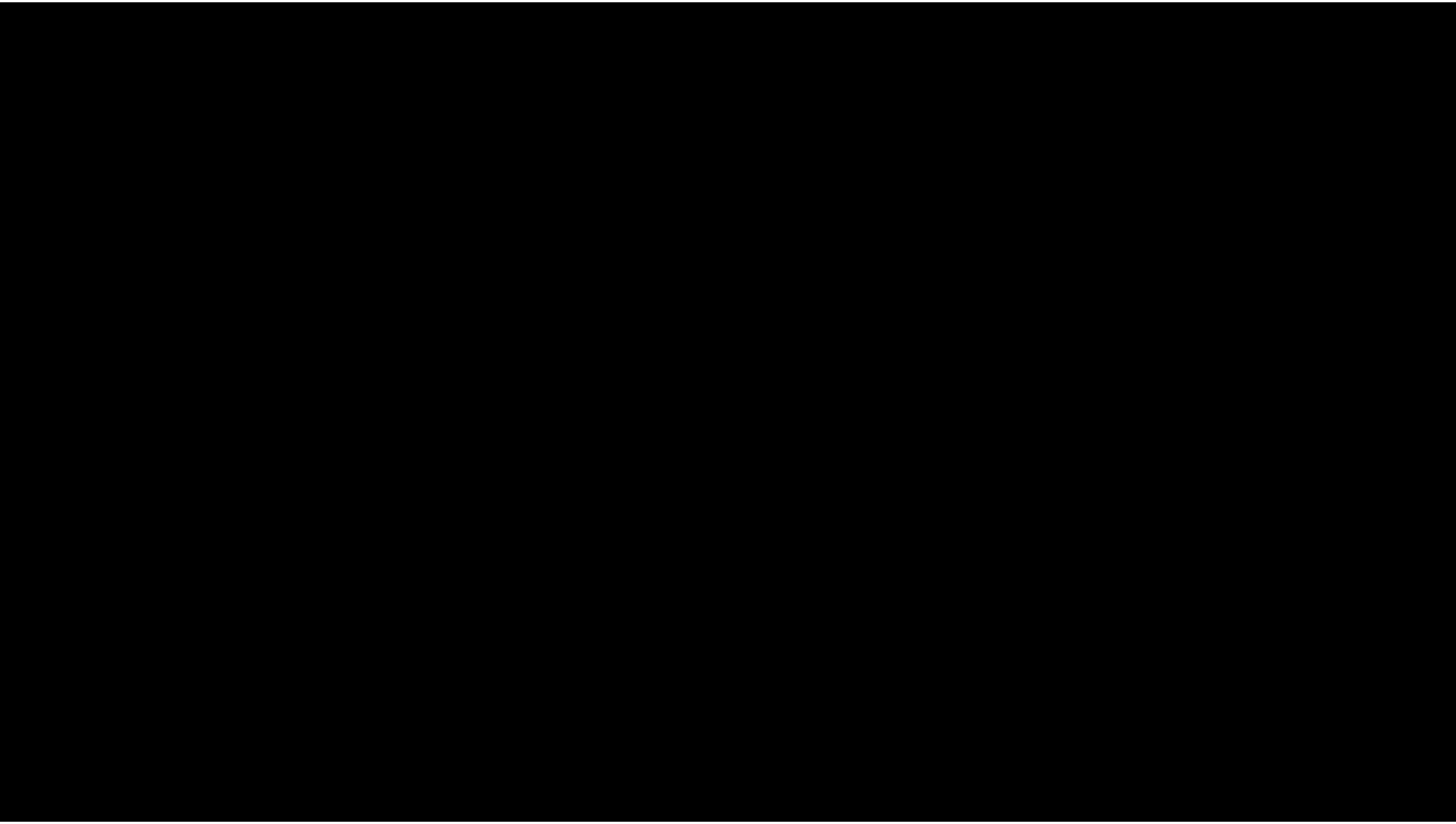
Depthmap

One depth estimate per vertex
100 - 10k vertices per mesh

Meshes Make Regularization Easy(er)

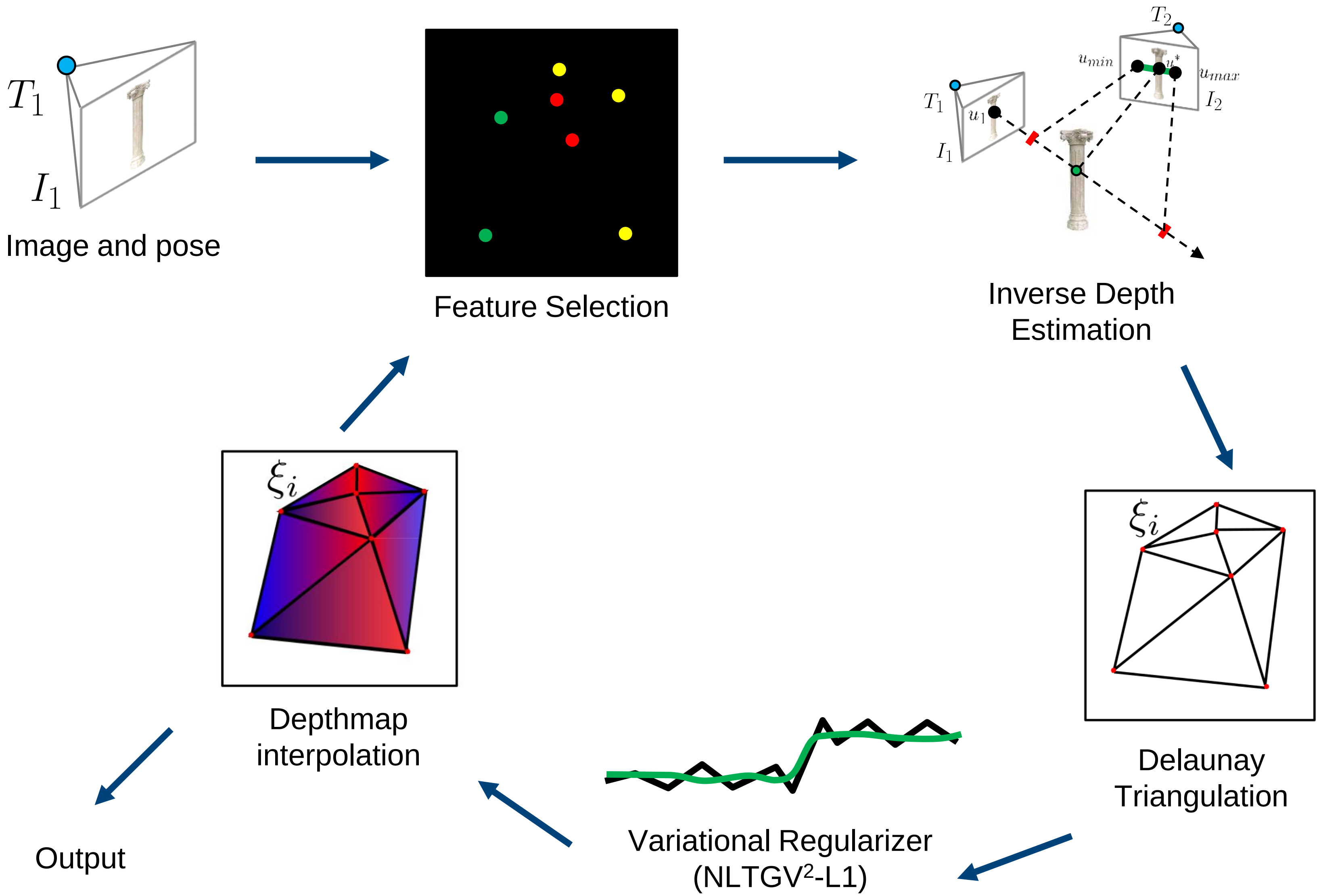


FLaME: Fast Lightweight Mesh Estimation

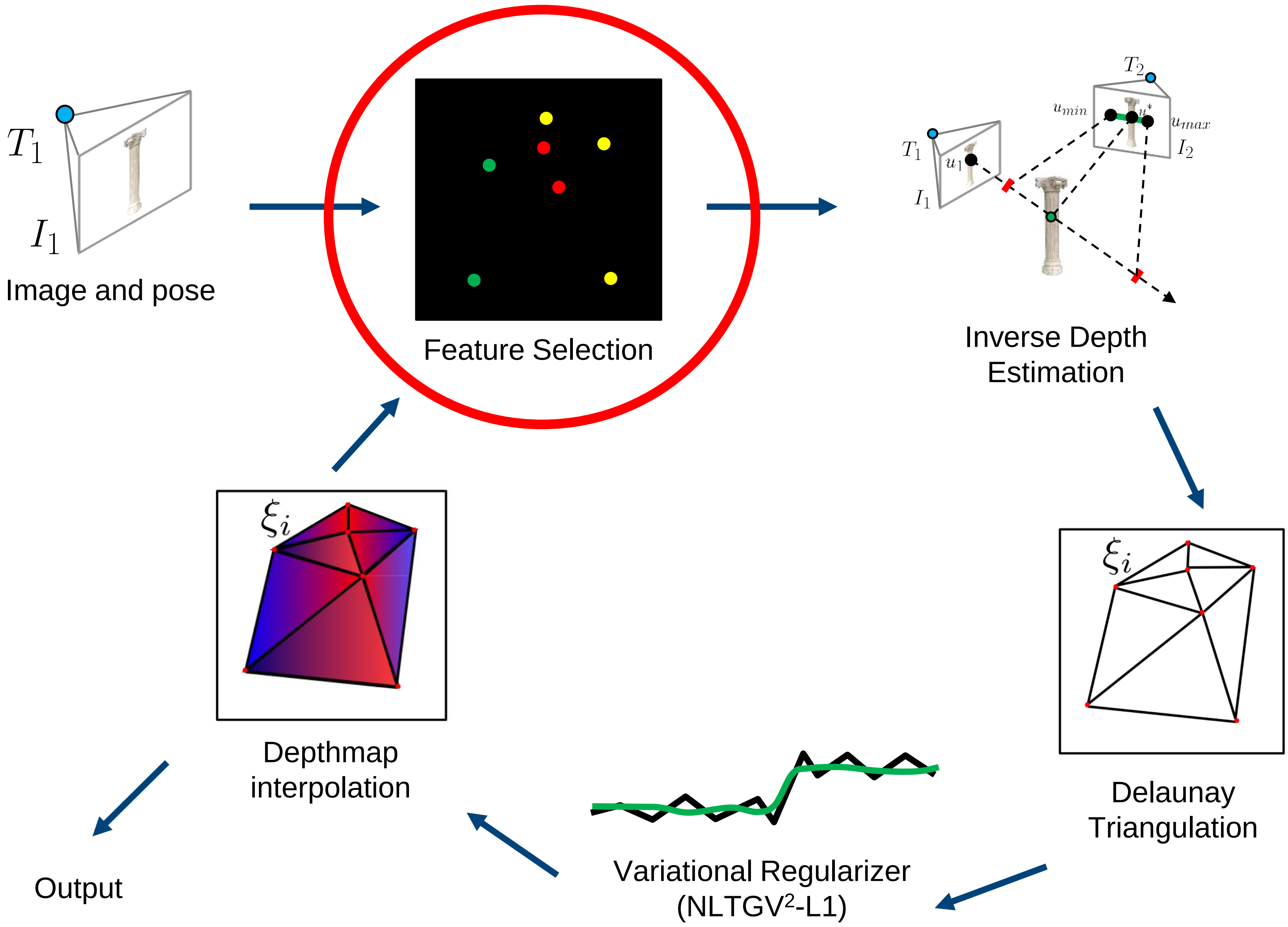


Fast (< 5 ms/frame) ***Lightweight*** (< 1 Intel i7 CPU core) ***Runs Onboard MAV***

FLaME: Fast Lightweight Mesh Estimation

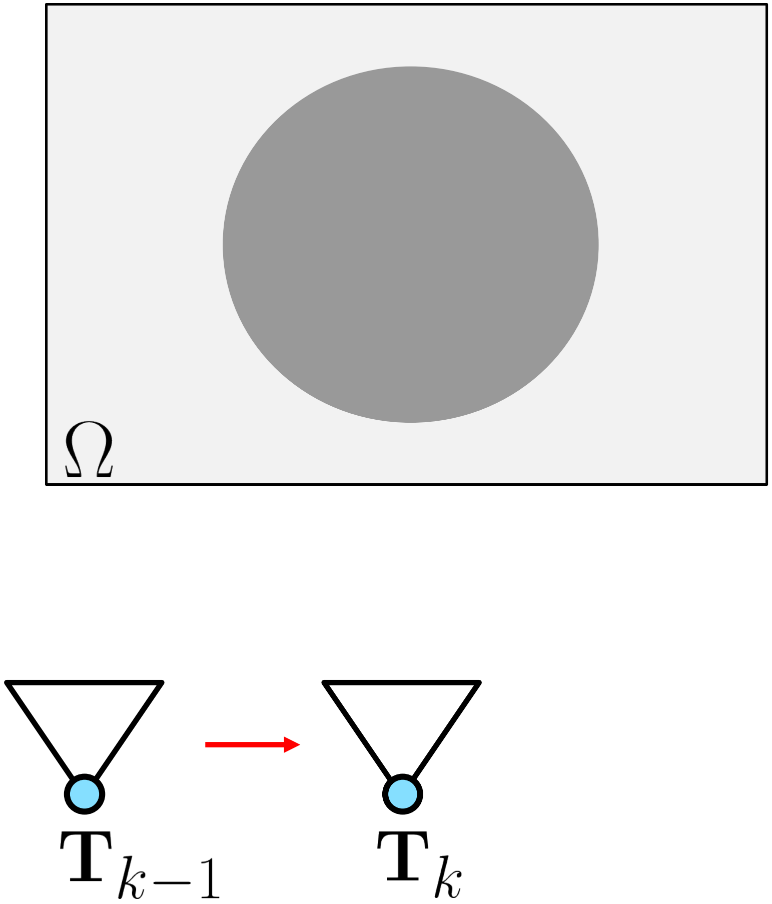


FLaME: Fast Lightweight Mesh Estimation



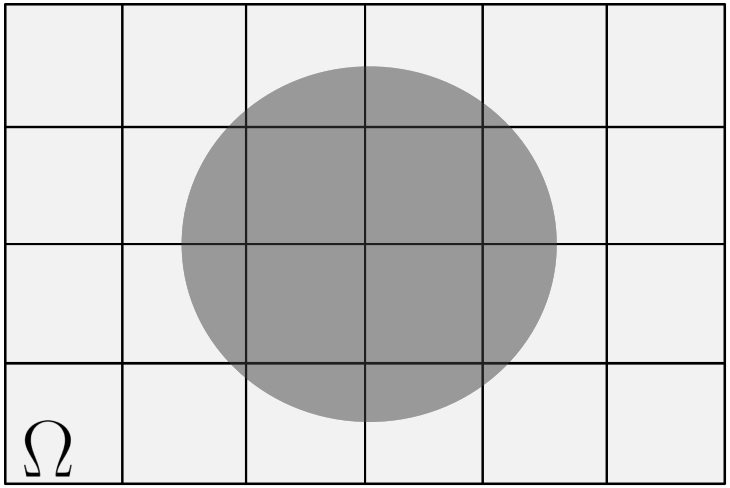
Select Easily Trackable Features

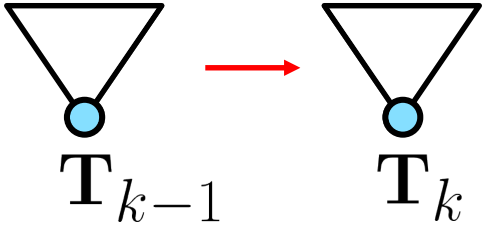
- At each frame we sample ***trackable*** pixels or ***features*** over the image
- These ***features*** will serve as ***potential vertices*** to add to the mesh



Select Easily Trackable Features

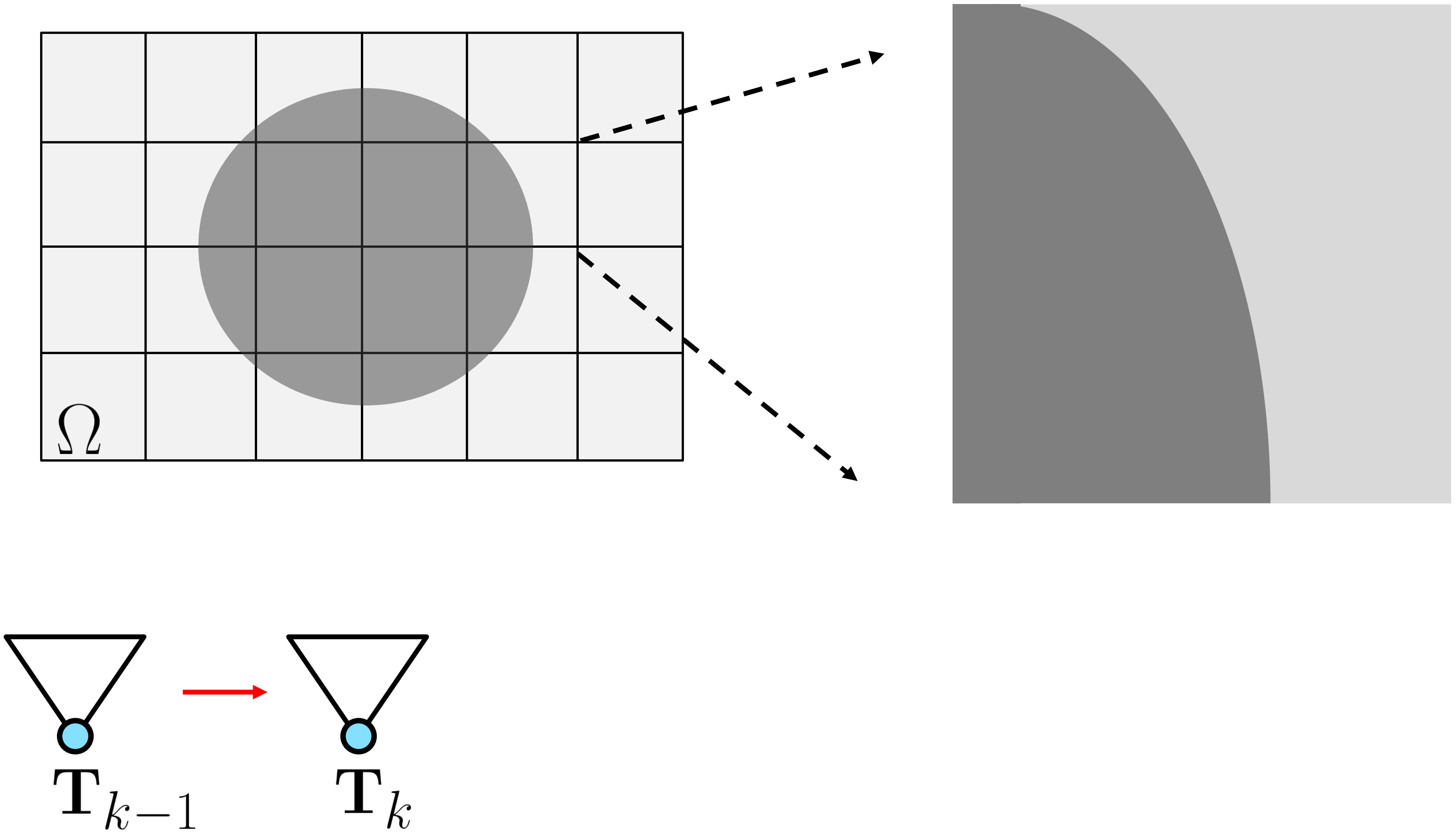
- At each frame we sample ***trackable*** pixels or ***features*** over the image
- These ***features*** will serve as ***potential vertices*** to add to the mesh





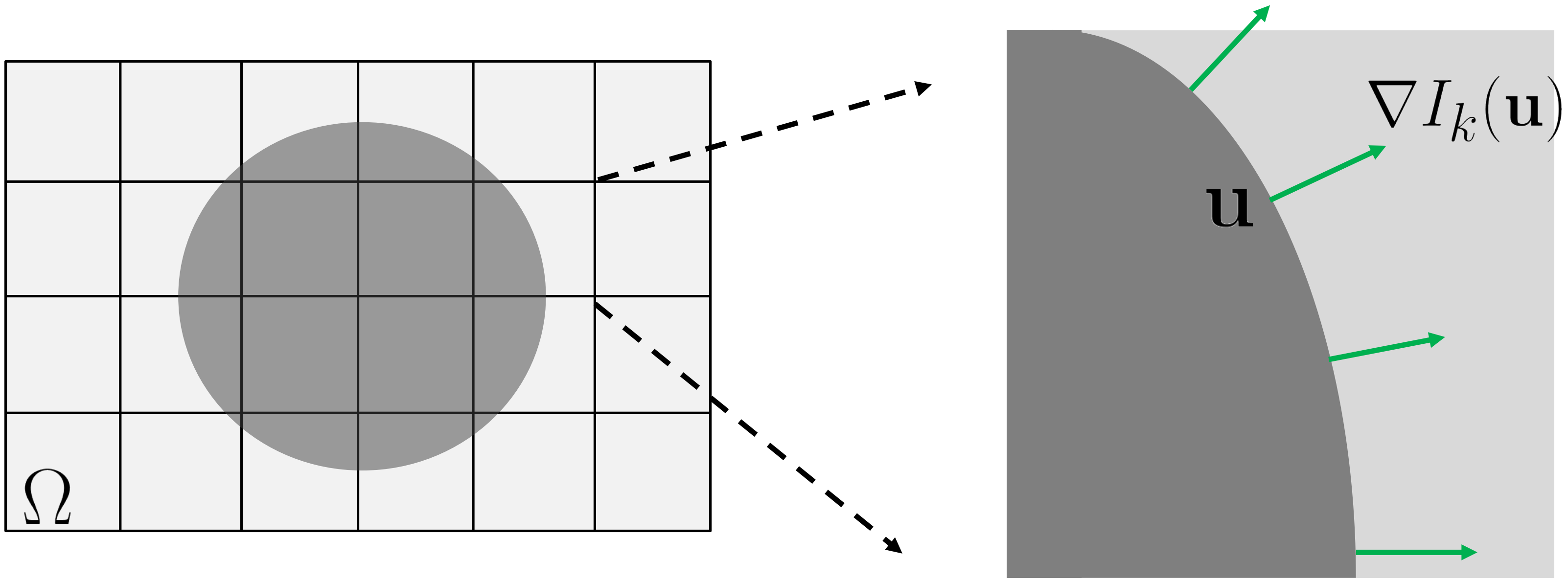
Select Easily Trackable Features

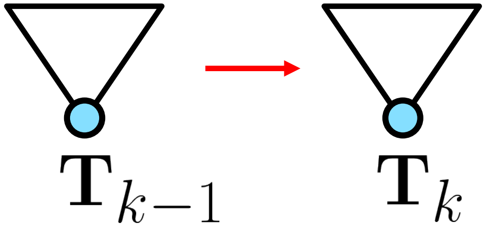
- At each frame we sample ***trackable*** pixels over the image domain
- These ***features*** will serve as ***potential vertices*** to add to the mesh



Select Easily Trackable Features

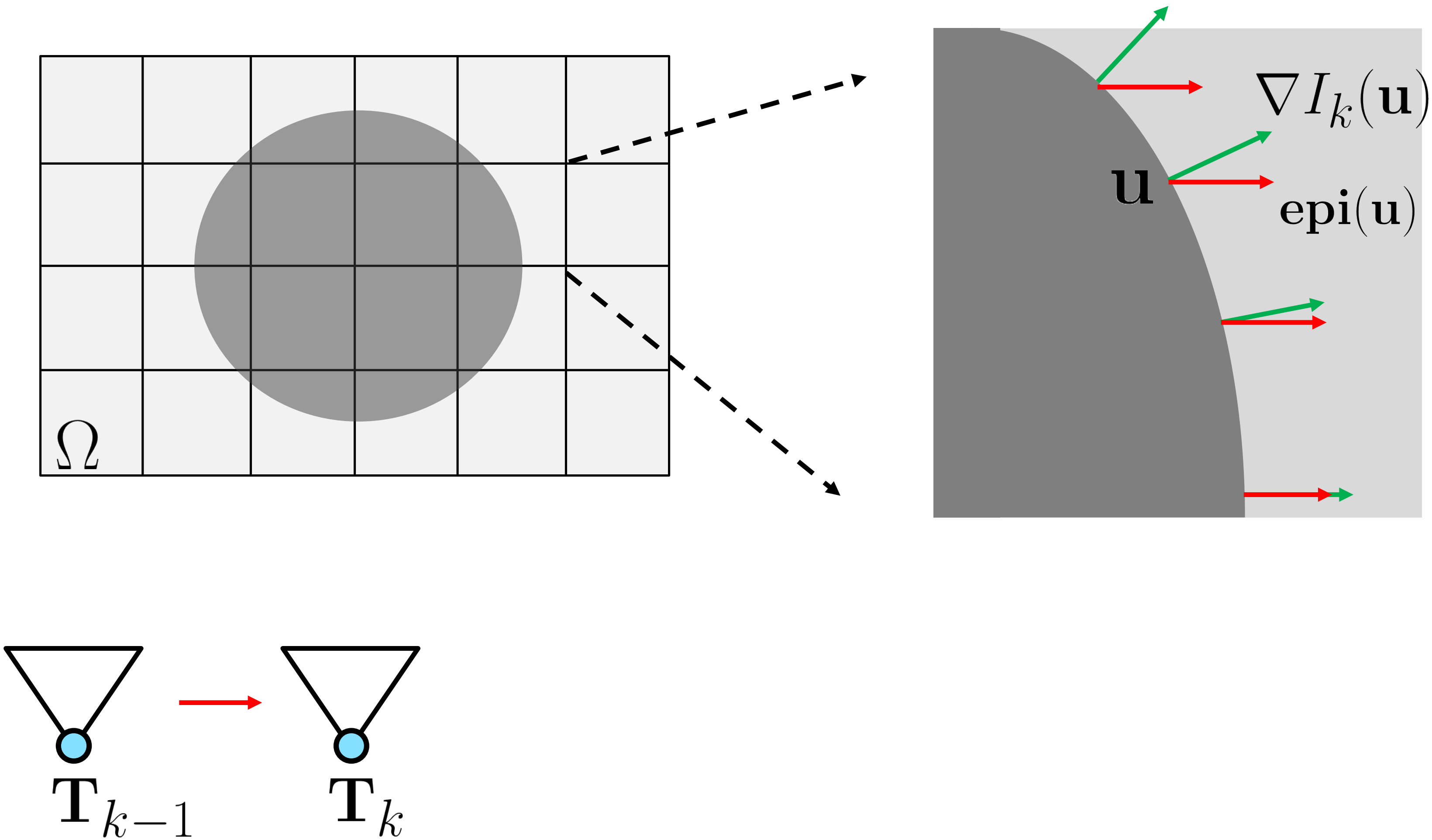
- At each frame we sample ***trackable*** pixels over the image domain
- These ***features*** will serve as ***potential vertices*** to add to the mesh





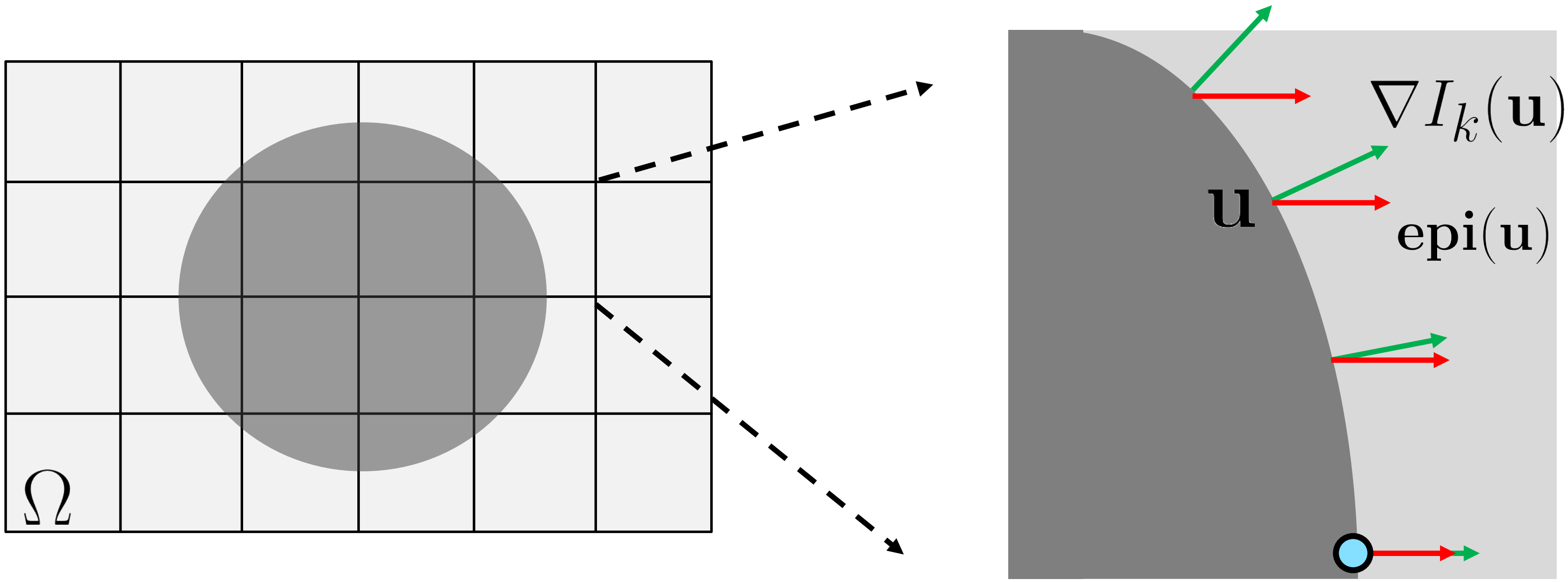
Select Easily Trackable Features

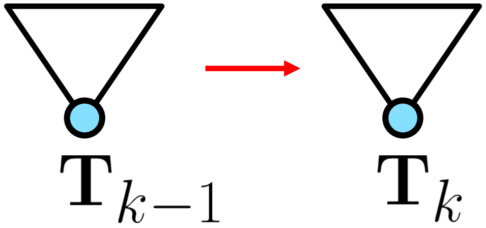
- At each frame we sample ***trackable*** pixels over the image domain
- These ***features*** will serve as ***potential vertices*** to add to the mesh



Select Easily Trackable Features

- At each frame we sample ***trackable*** pixels over the image domain
- These ***features*** will serve as ***potential vertices*** to add to the mesh

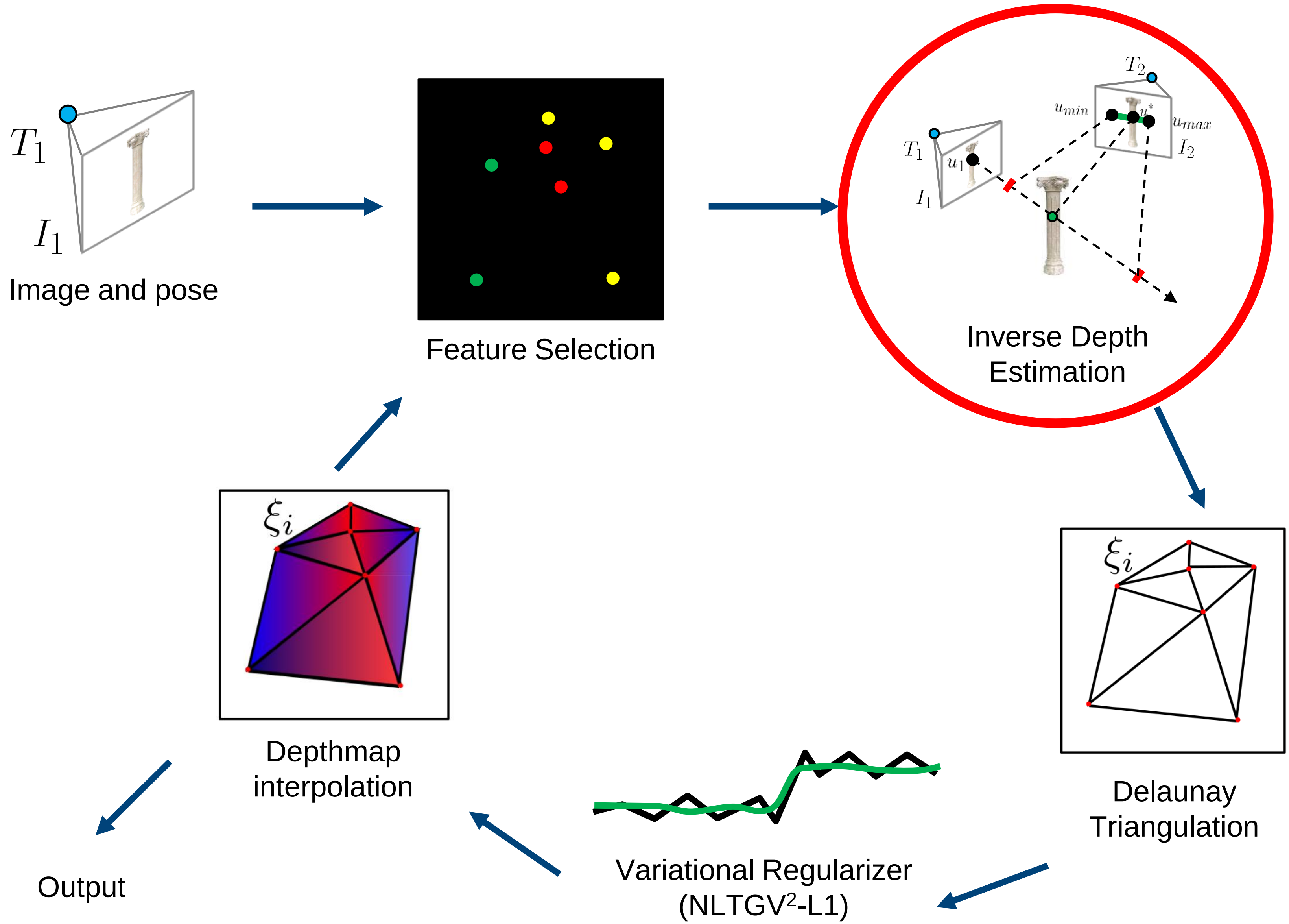




Select pixel in each grid cell with maximum score:

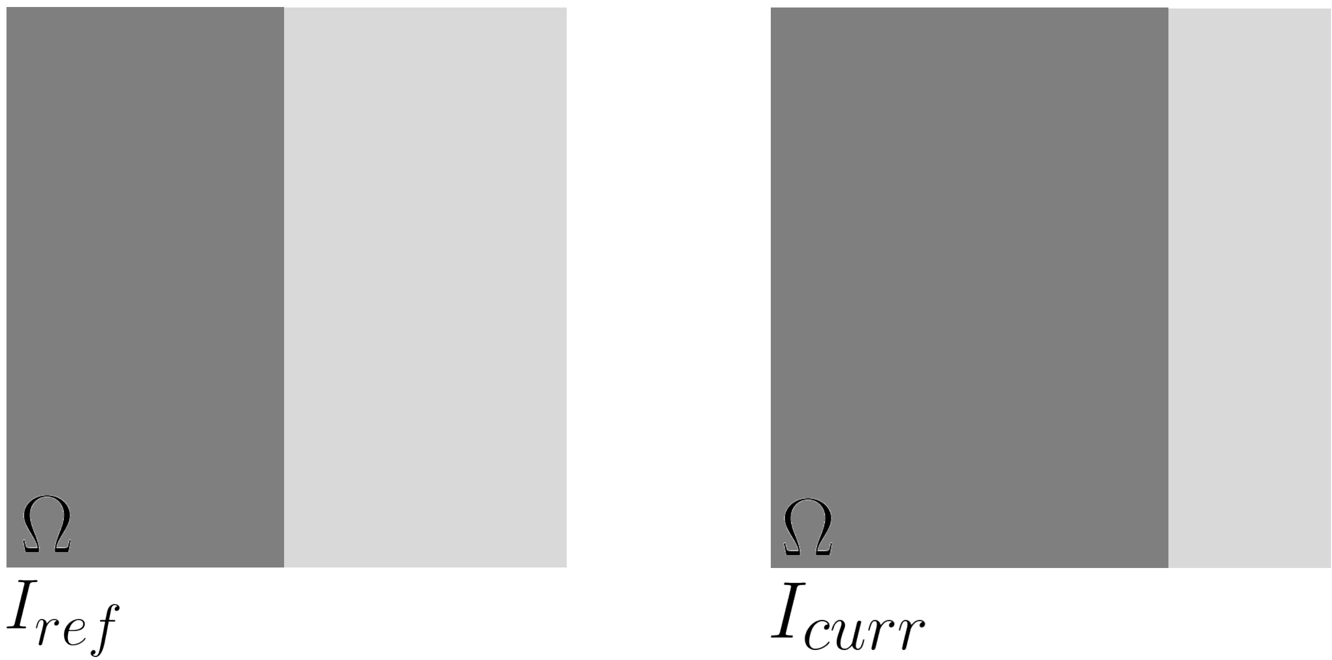
$$s(\mathbf{u}) = |\nabla I_k(\mathbf{u}) \cdot \mathbf{epi}(\mathbf{u})|$$

FLaME: Fast Lightweight Mesh Estimation



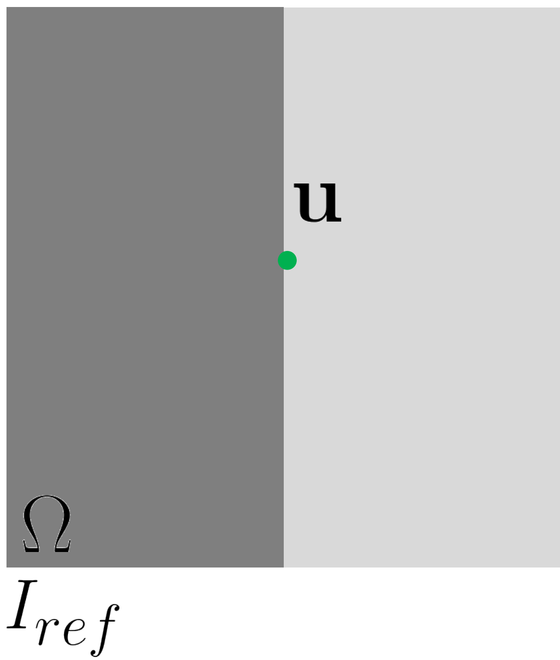
Estimate Inverse Depths for Features

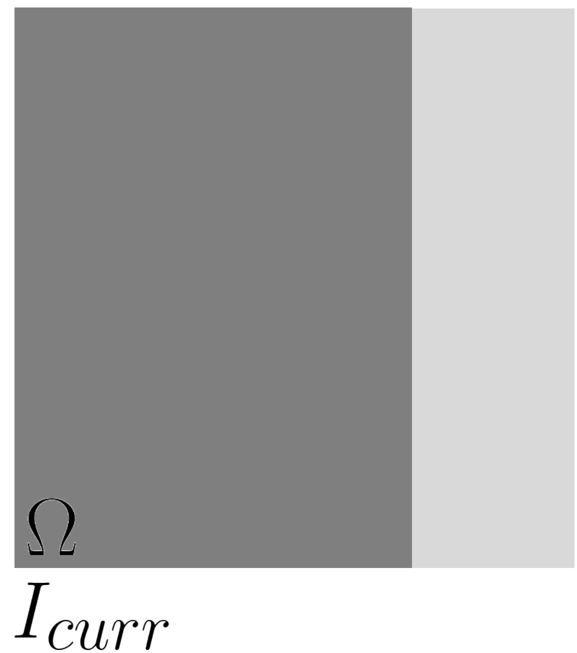
- We ***track*** features across new images using ***direct epipolar stereo comparisons***



Estimate Inverse Depths for Features

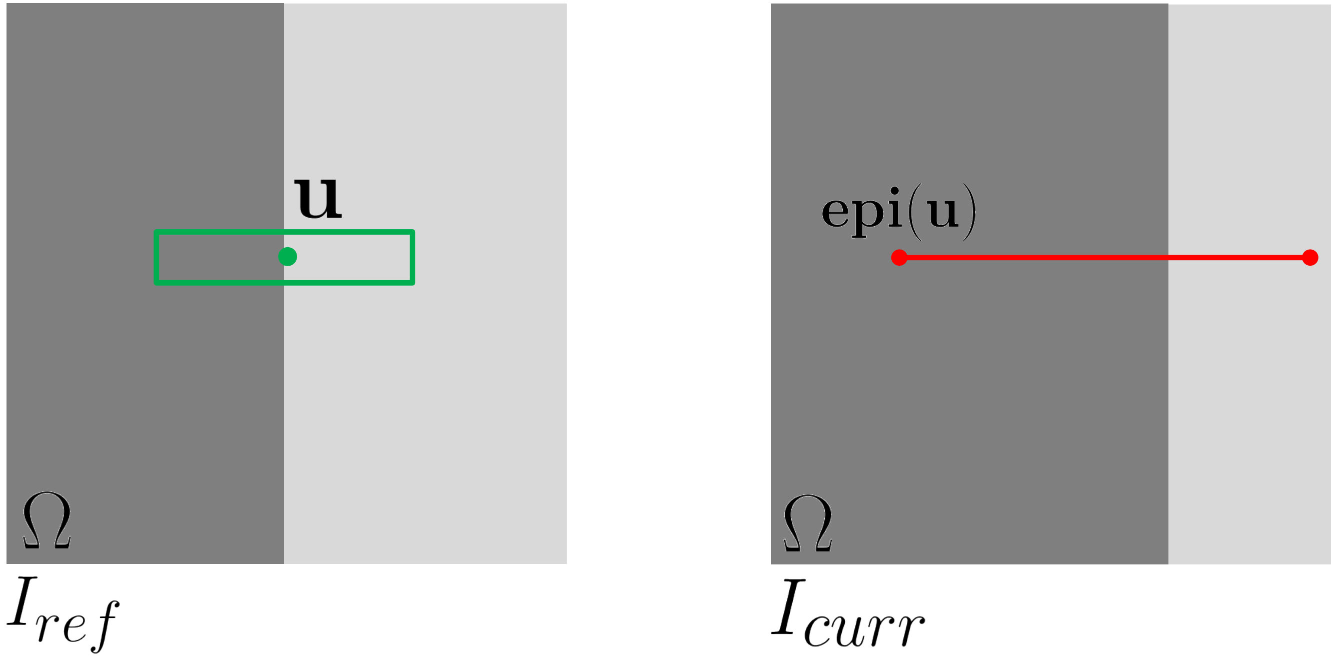
- We ***track*** features across new images using ***direct epipolar stereo comparisons***





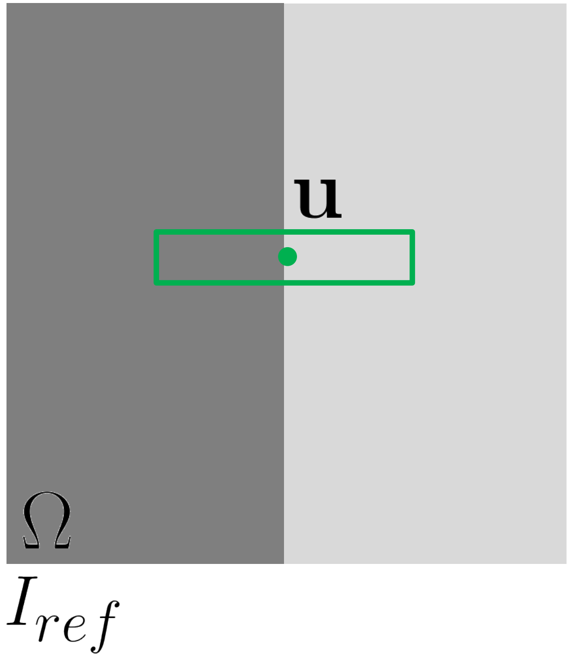
Estimate Inverse Depths for Features

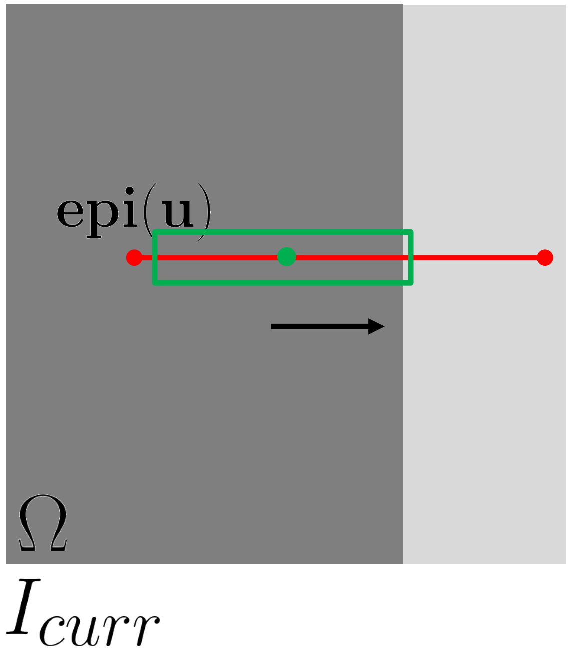
- We ***track*** features across new images using ***direct epipolar stereo comparisons***



Estimate Inverse Depths for Features

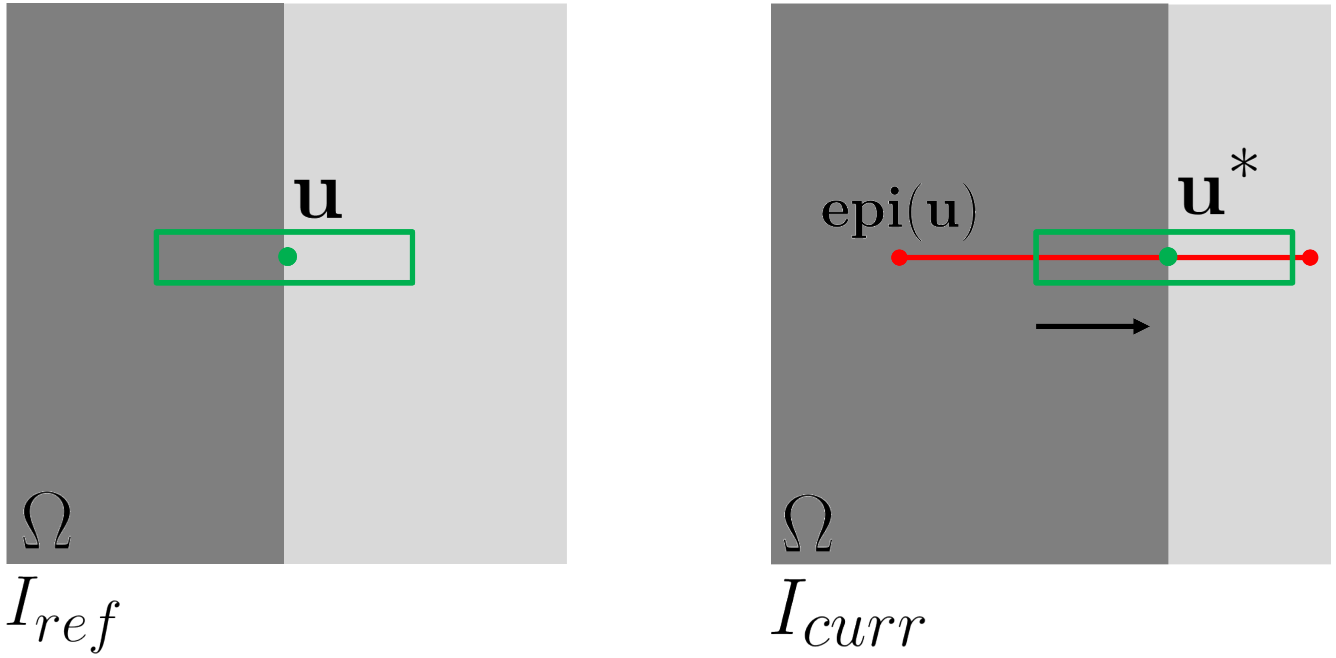
- We ***track*** features across new images using ***direct epipolar stereo comparisons***





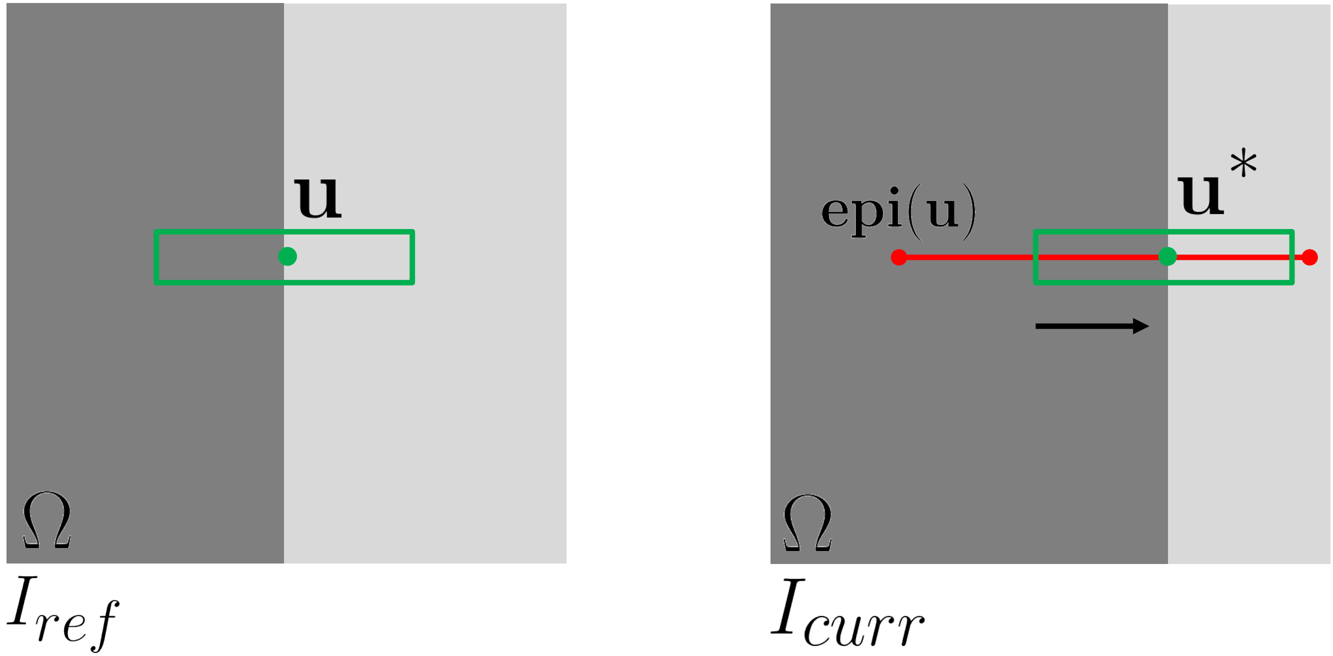
Estimate Inverse Depths for Features

- We ***track*** features across new images using ***direct epipolar stereo comparisons***



Estimate Inverse Depths for Features

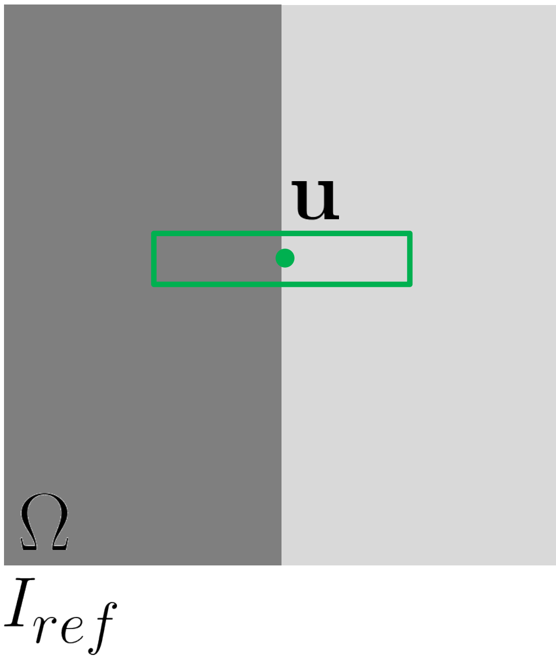
- We ***track*** features across new images using ***direct epipolar stereo comparisons***

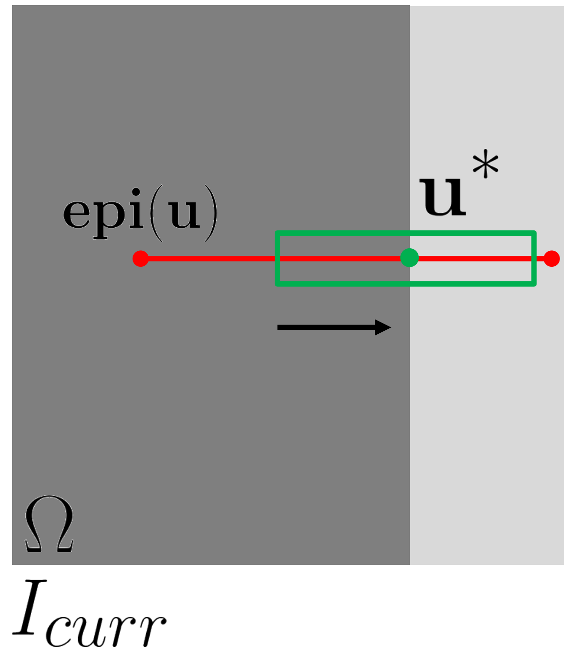


- Each successful match generates an inverse depth ***measurement*** (ξ_z, σ_z^2)

Estimate Inverse Depths for Features

- We ***track*** features across new images using ***direct epipolar stereo comparisons***

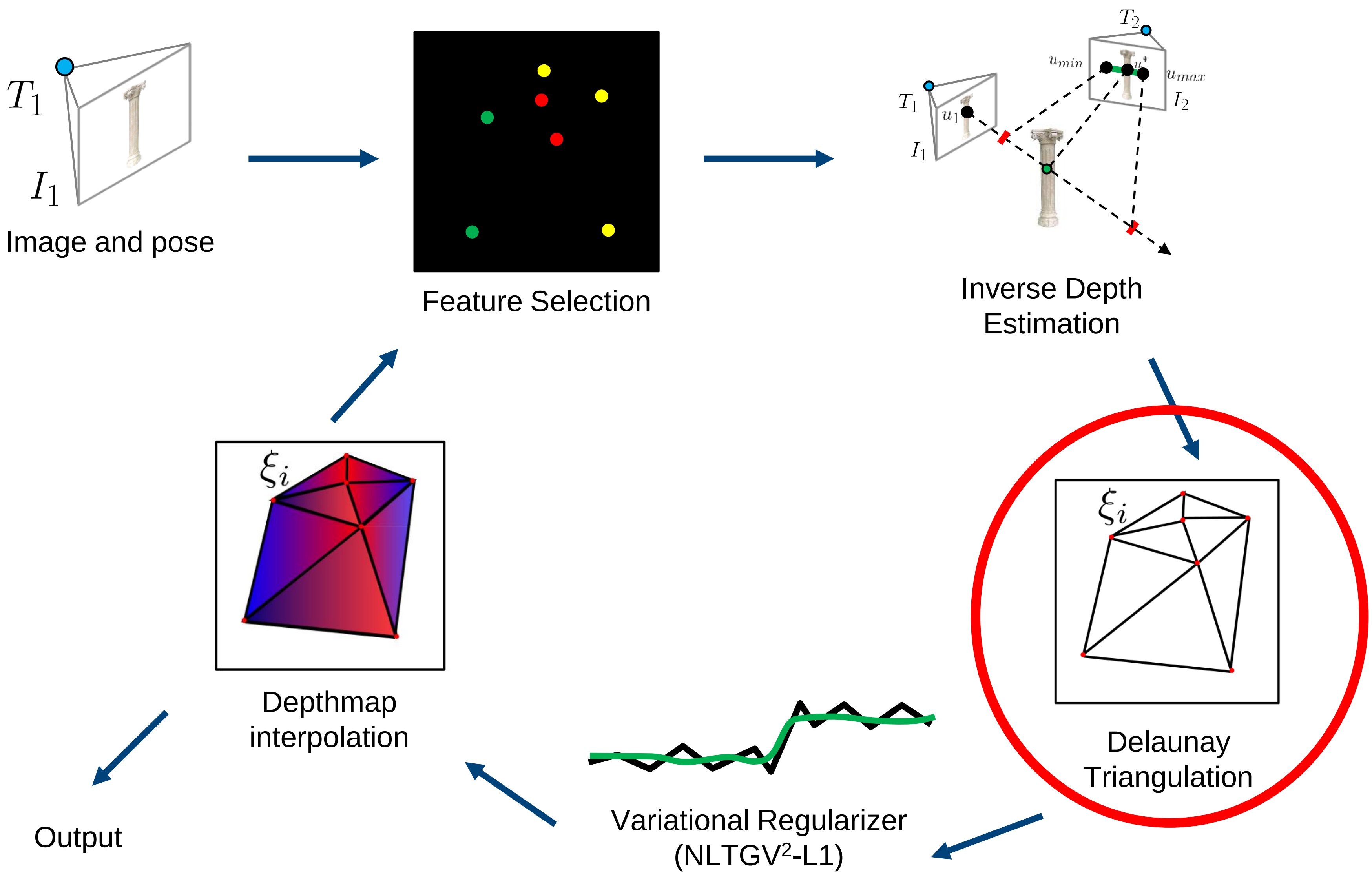




- Each successful match generates an inverse depth ***measurement*** (ξ_z, σ_z^2)
- Measurements are ***fused*** over time using Bayesian update

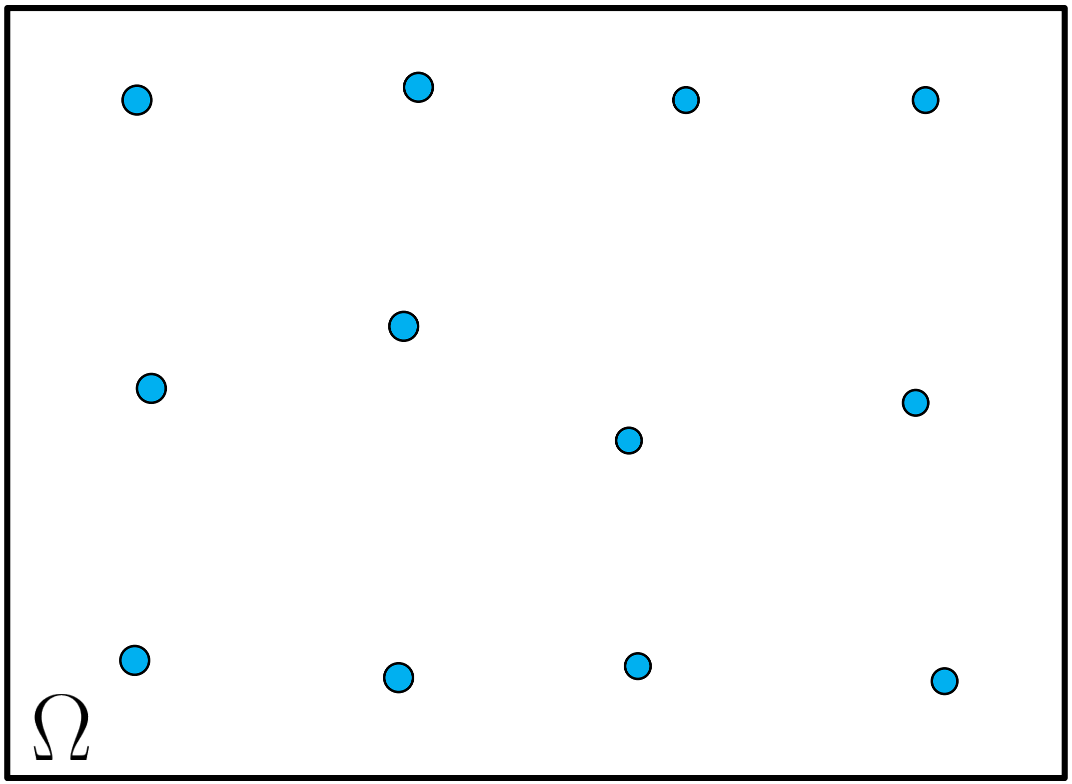
$$\xi_f \leftarrow \frac{\xi_f \sigma_z^2 + \xi_z \sigma_f^2}{\sigma_f^2 + \sigma_z^2}, \quad \sigma_f^2 \leftarrow \frac{\sigma_f^2 \sigma_z^2}{\sigma_f^2 + \sigma_x^2}$$

FLaME: Fast Lightweight Mesh Estimation



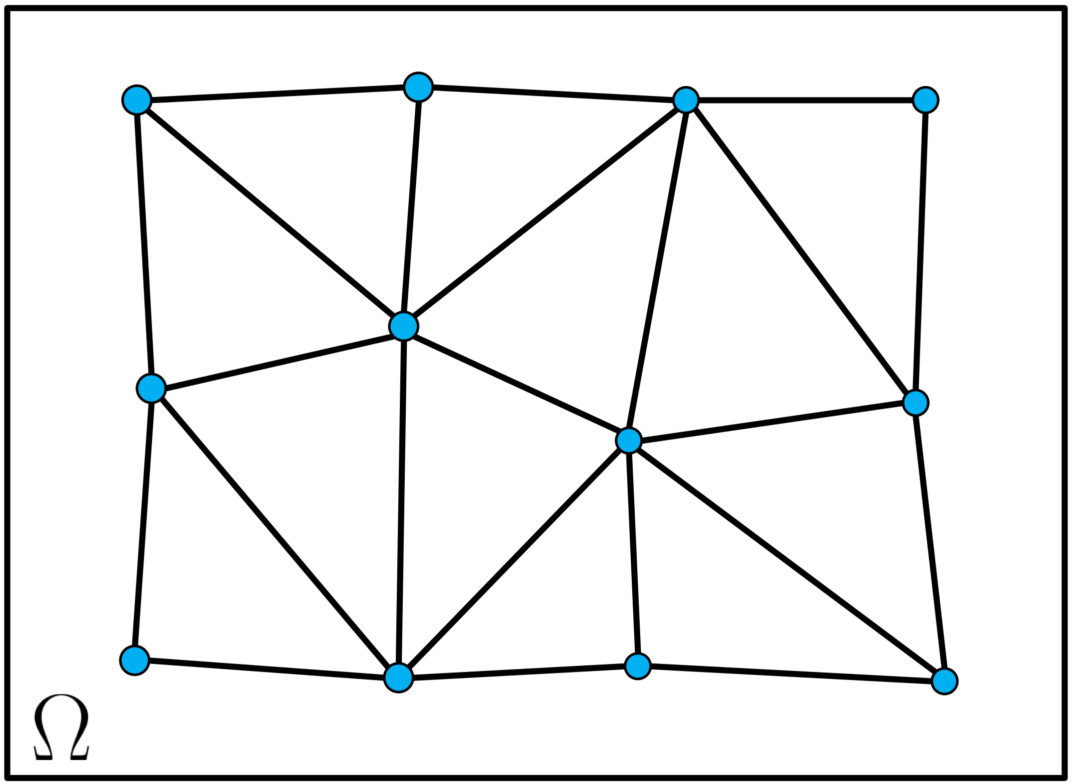
Triangulate Features into Mesh

- Once a features inverse depth variance drops below threshold, we insert it into the mesh using ***Delaunay triangulations***



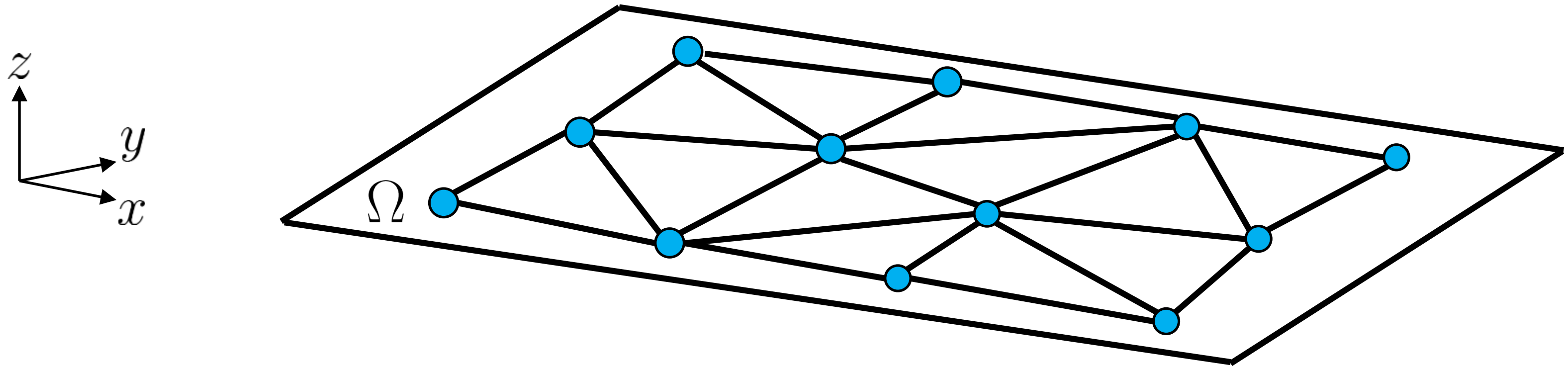
Triangulate Features into Mesh

- Once a features inverse depth variance drops below threshold, we insert it into the mesh using ***Delaunay triangulations***



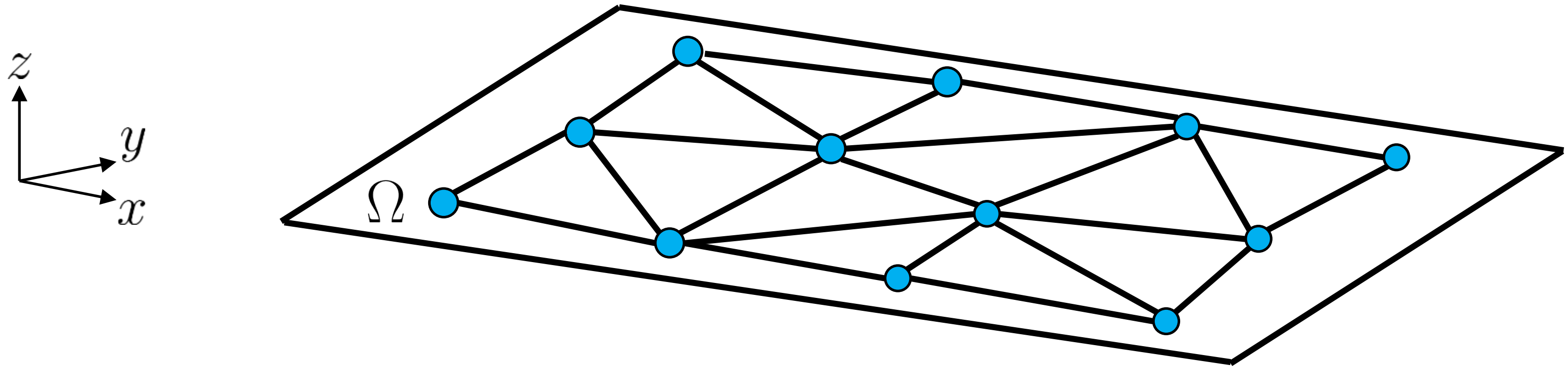
Triangulate Features into Mesh

- Once a features inverse depth variance drops below threshold, we insert it into the mesh using ***Delaunay triangulations***



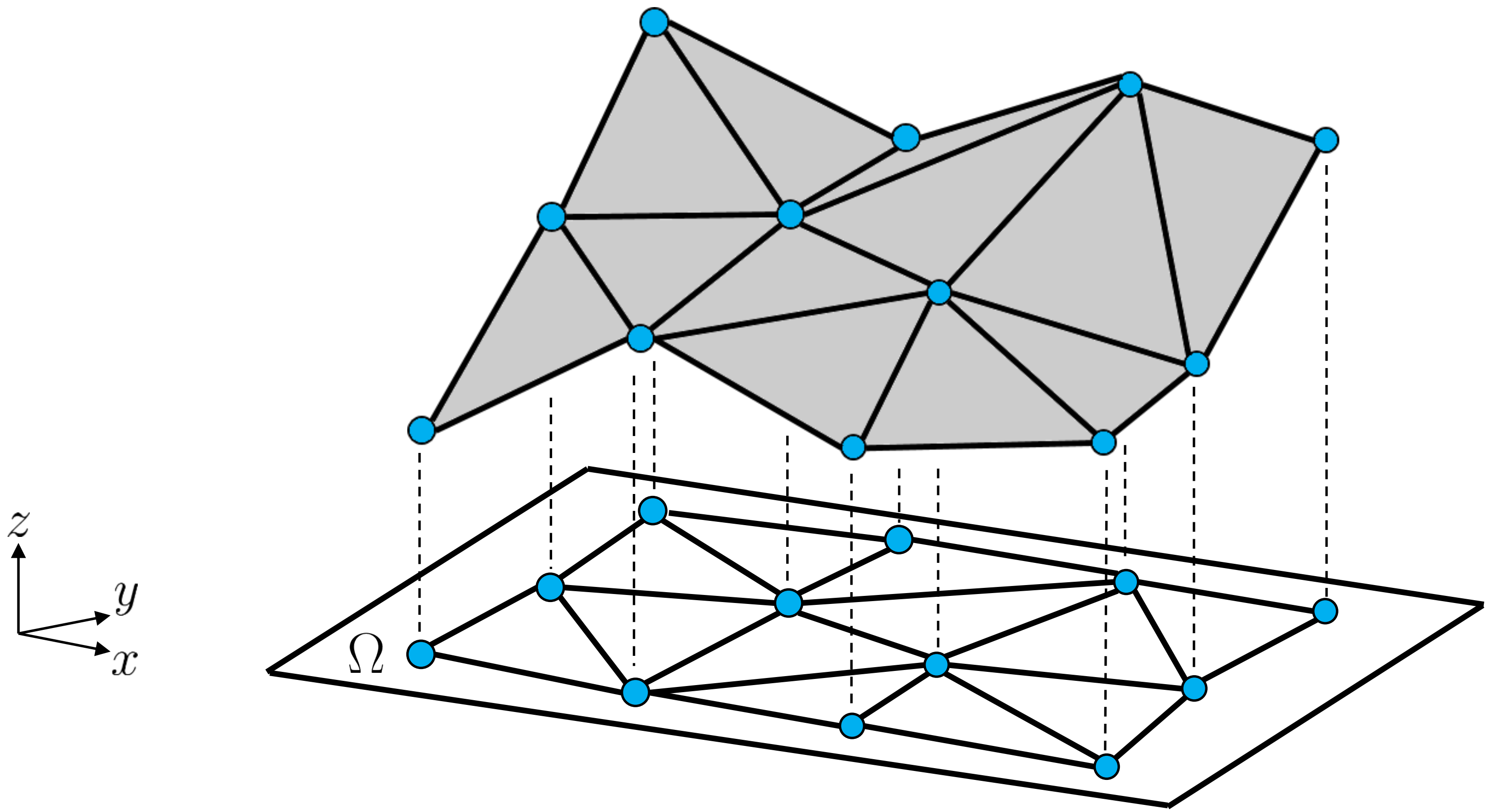
Triangulate Features into Mesh

- Once a features inverse depth variance drops below threshold, we insert it into the mesh using ***Delaunay triangulations***
- We can project 2D mesh into 3D using the inverse depth for each vertex

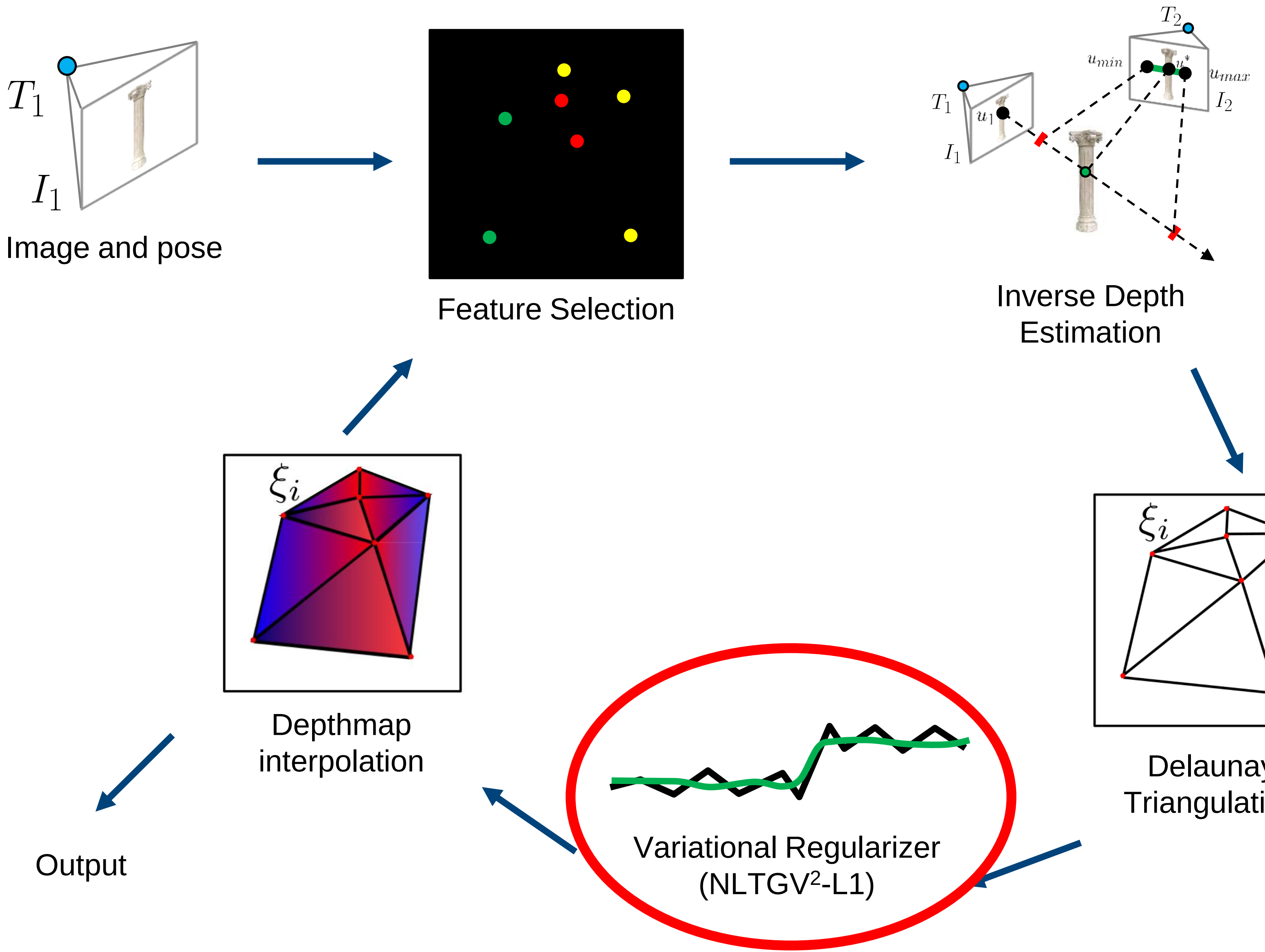


Triangulate Features into Mesh

- Once a features inverse depth variance drops below threshold, we insert it into the mesh using ***Delaunay triangulations***
- We can project 2D mesh into 3D using the inverse depth for each vertex

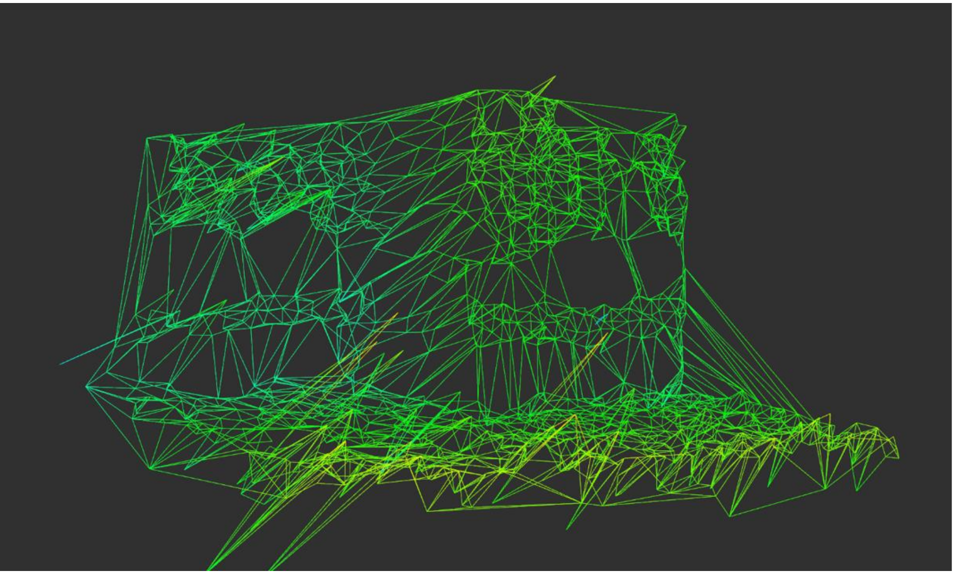


FLaME: Fast Lightweight Mesh Estimation



Smooth Mesh Using Variational Regularization

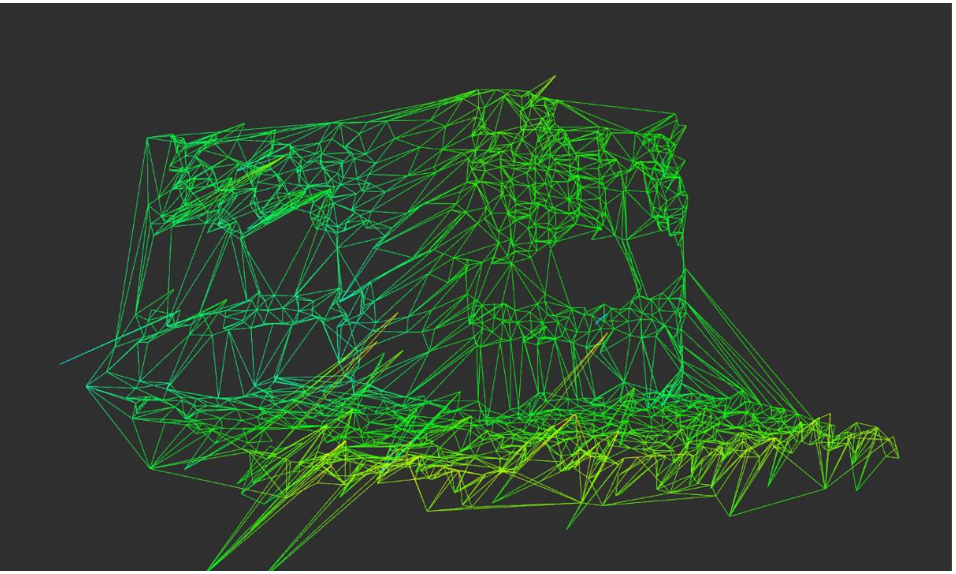
- Mesh inverse depths are noisy and prone to outliers
- Need to spatially regularize/smooth inverse depths



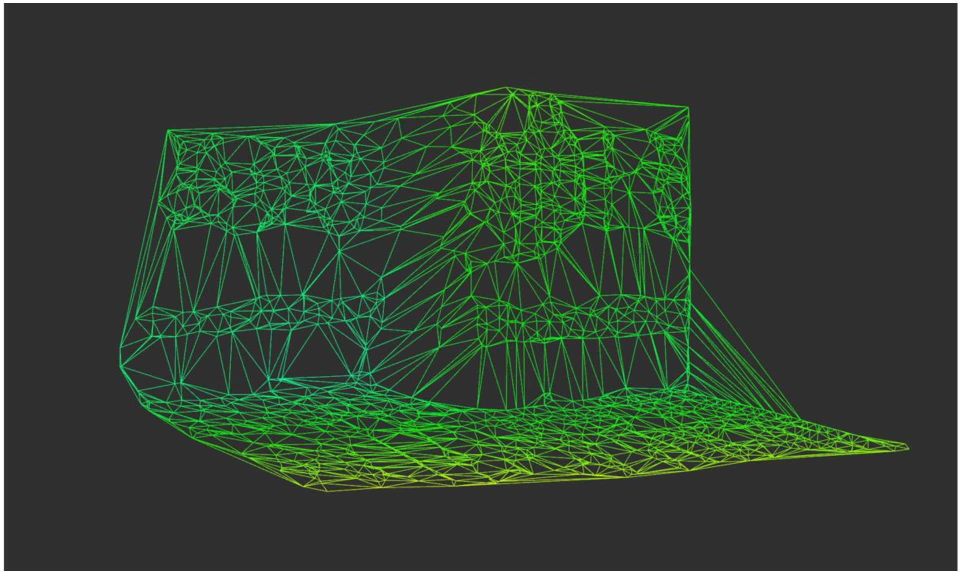
Raw

Smooth Mesh Using Variational Regularization

- Mesh inverse depths are noisy and prone to outliers
- Need to spatially regularize/smooth inverse depths
- Will exploit graph structure to make optimization fast and incremental

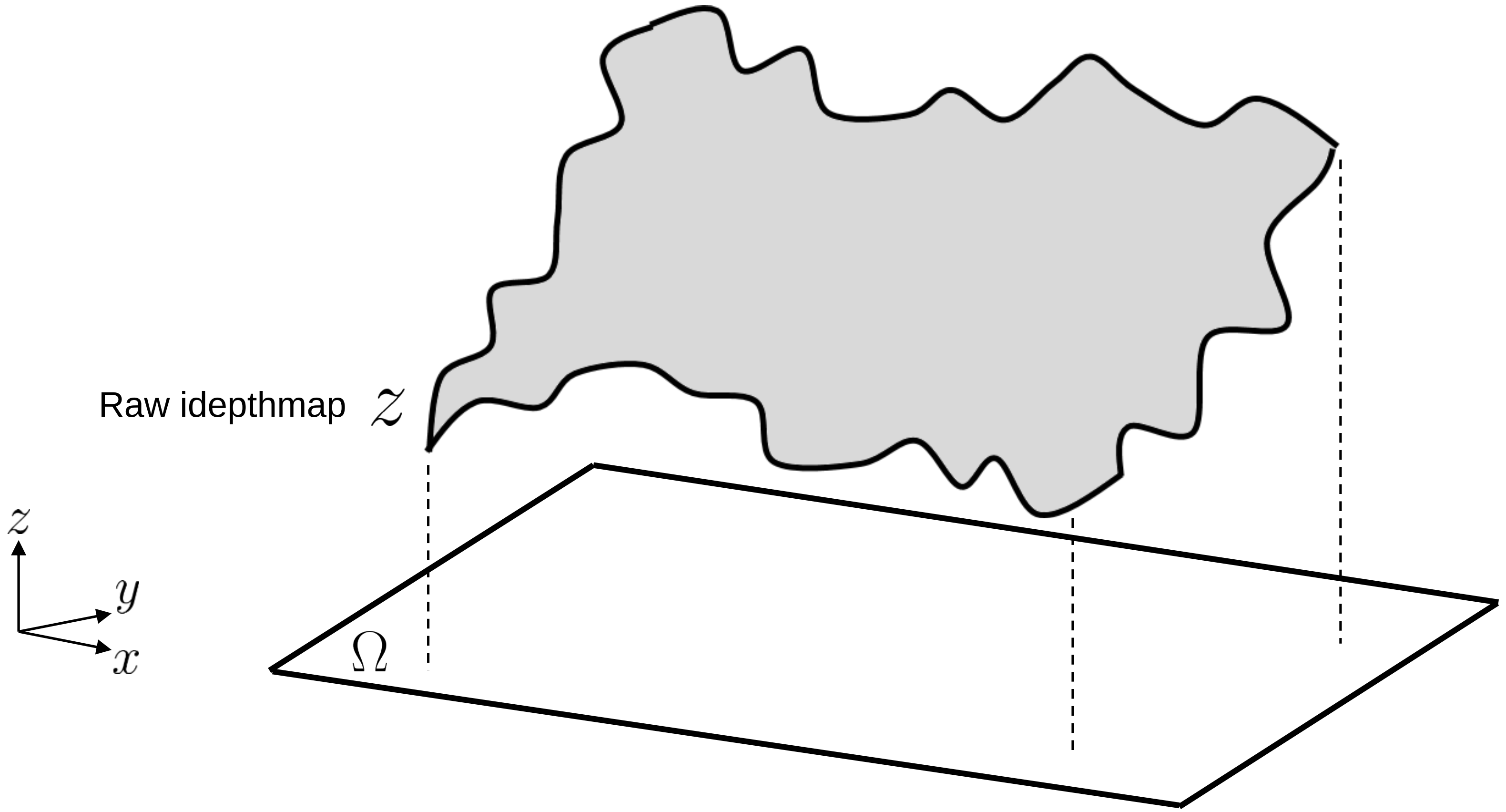


Raw

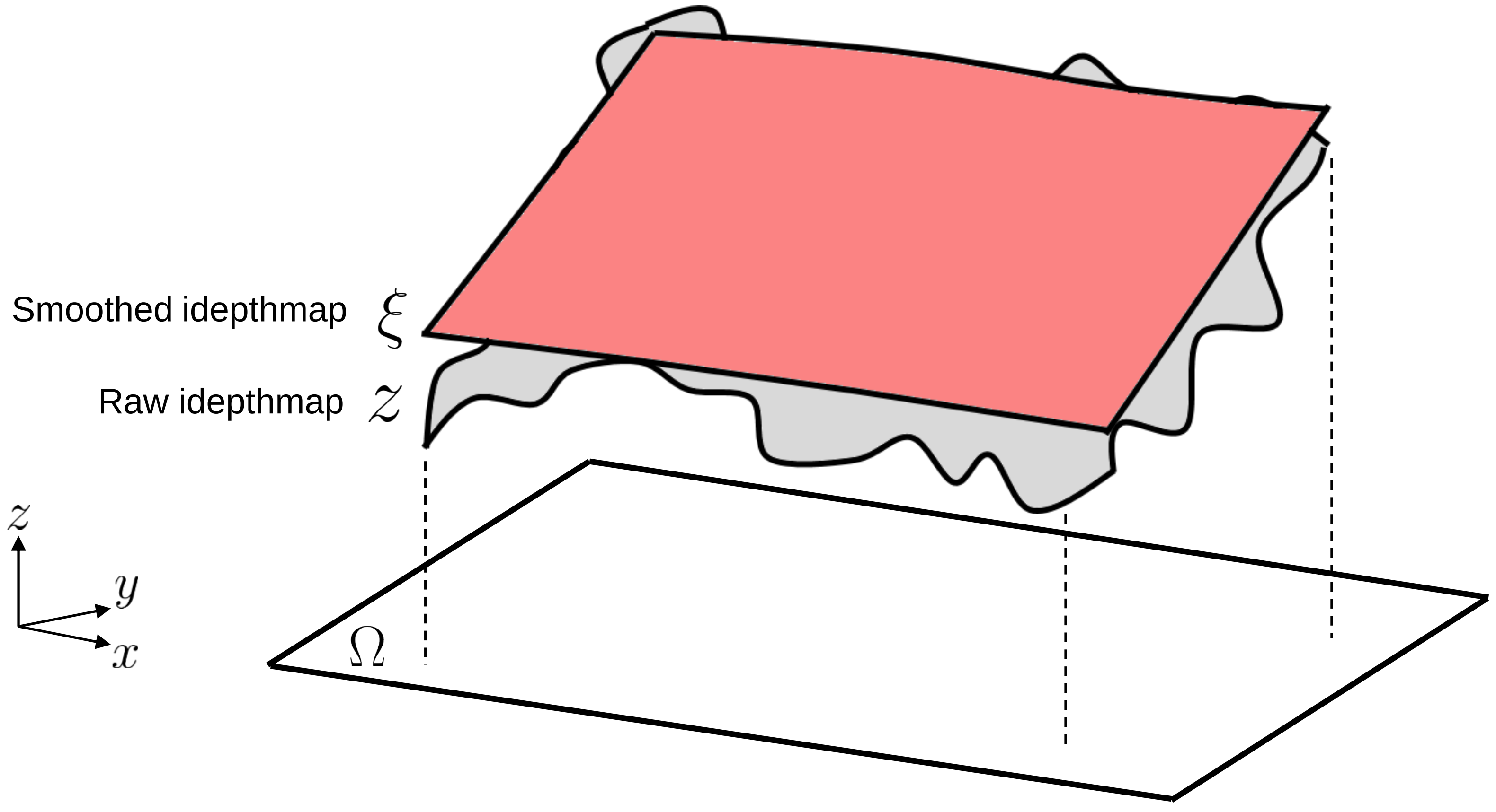


Smoothed

Define Variational Cost over Inverse Depthmap

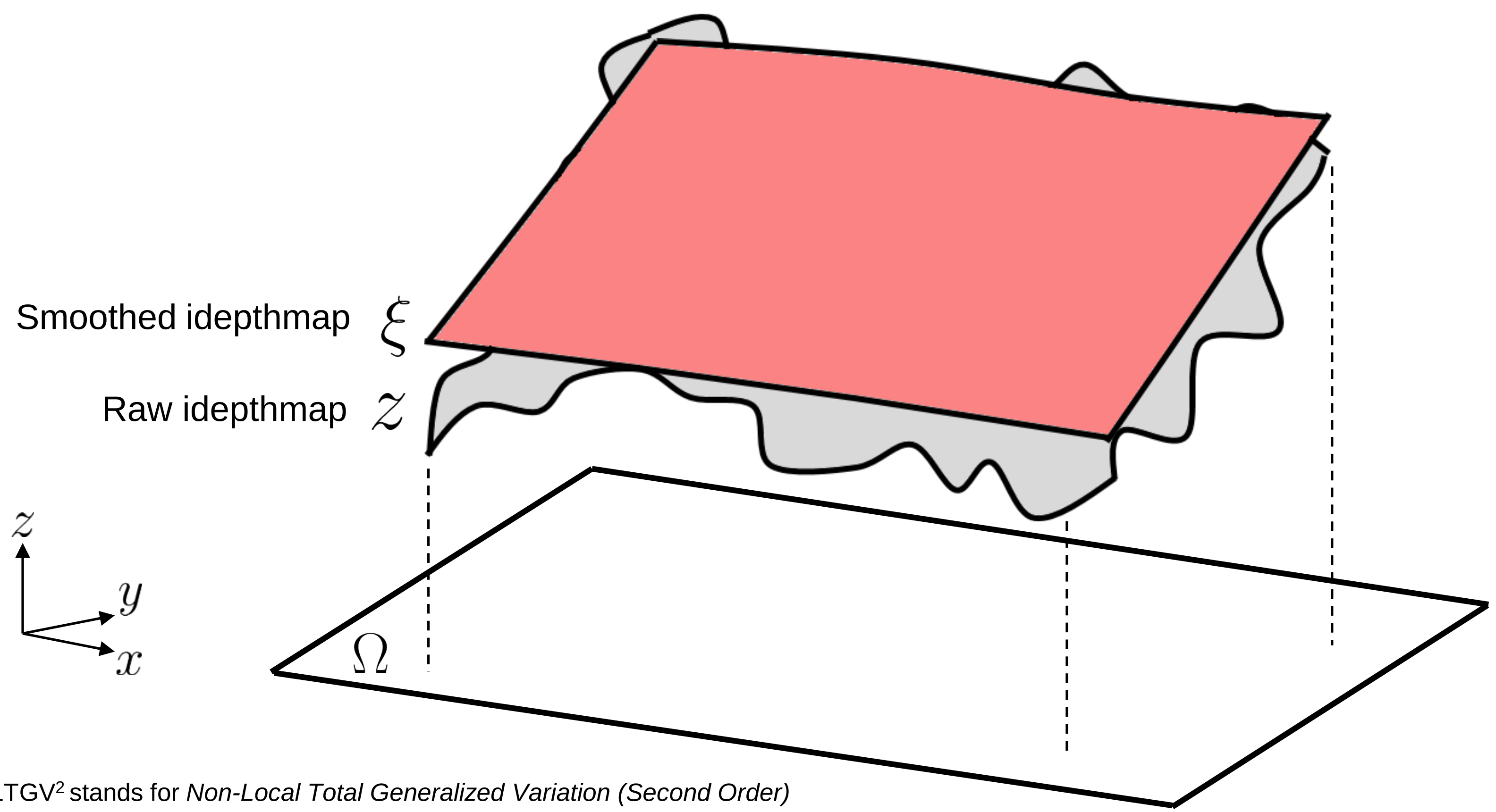


Define Variational Cost over Inverse Depthmap



Define Variational Cost over Inverse Depthmap

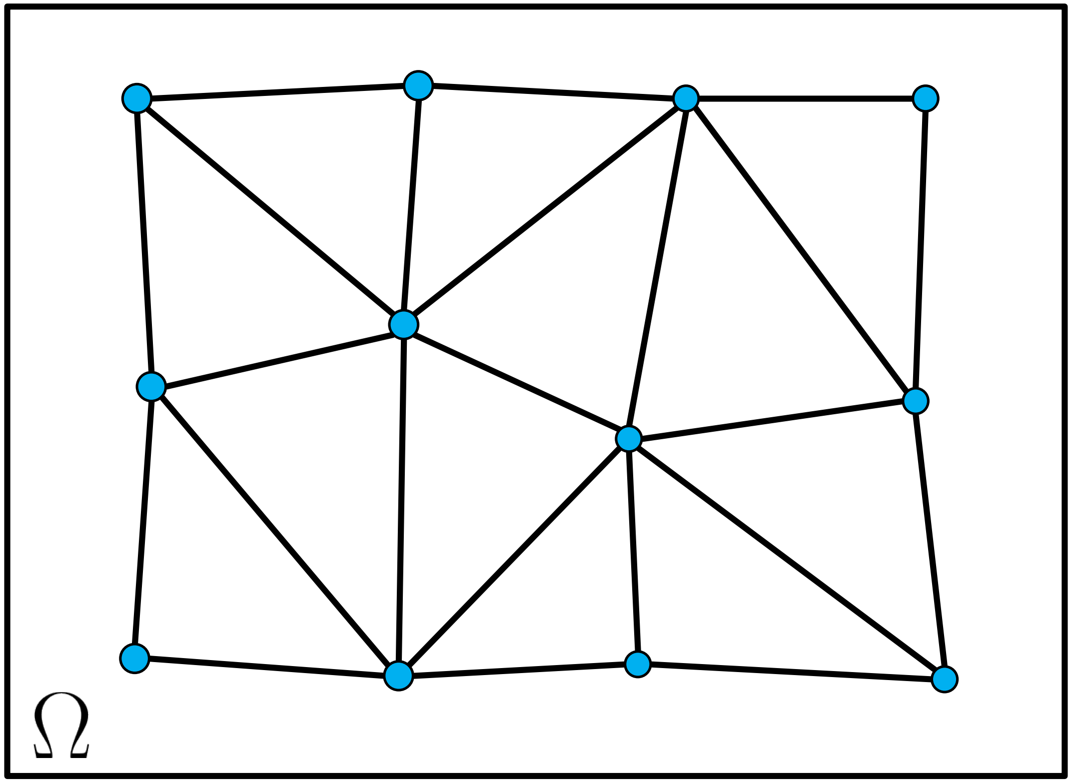
$$E(\xi) = \mathrm{NLTGV}^2(\xi) + \lambda \int_\Omega |\xi(\mathbf{u}) - z(\mathbf{u})| \; d\mathbf{u}$$



NLTGV[2] stands for *Non-Local Total Generalized Variation (Second Order)*
Ranftl et al. 2014, Pinies et al. 2015

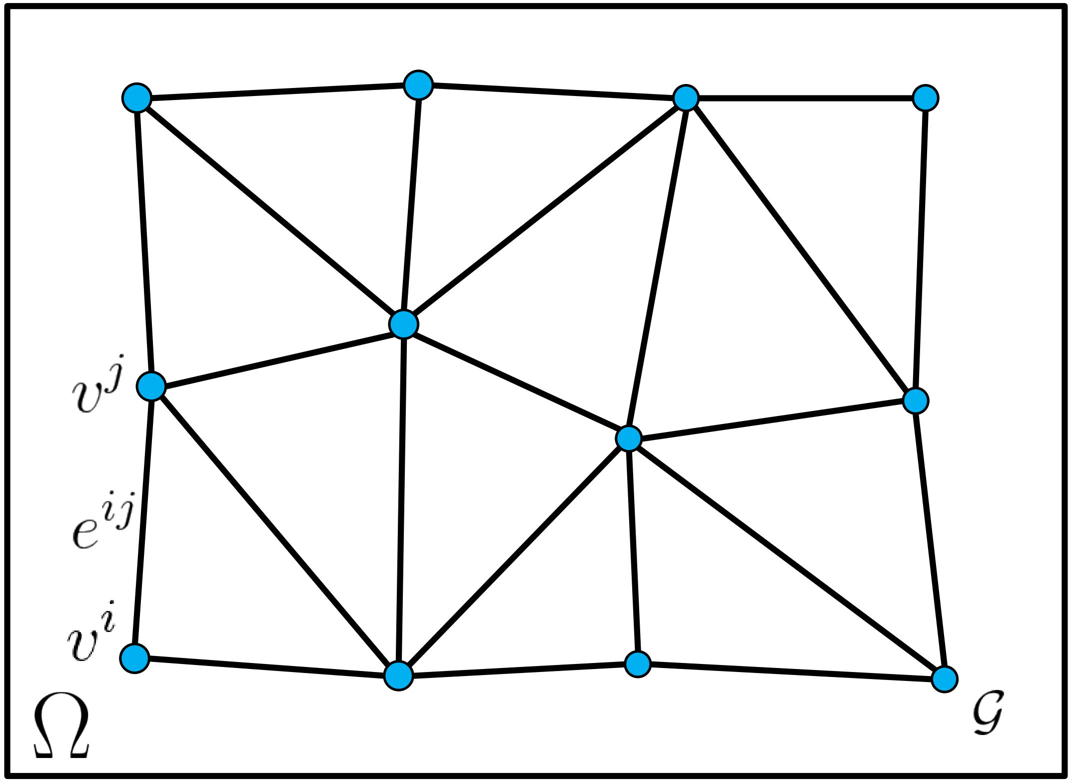
Approximate Cost over Mesh

- Reinterpret mesh as graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

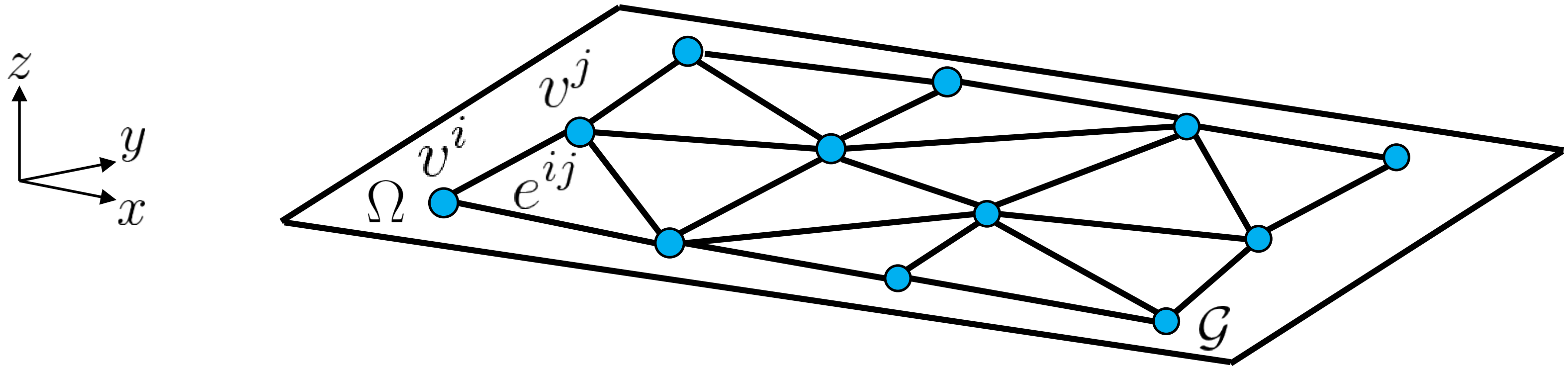


Approximate Cost over Mesh

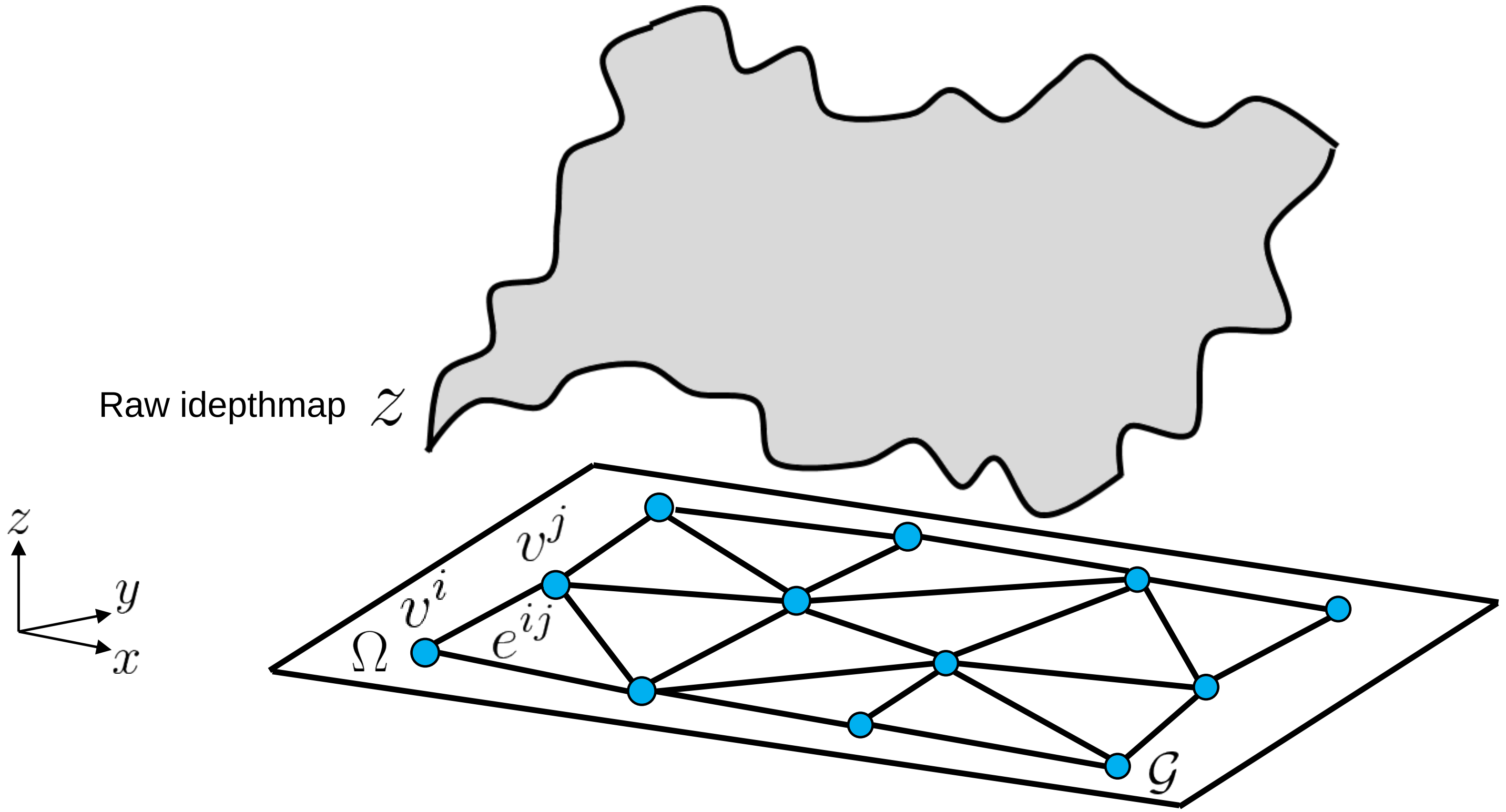
- Reinterpret mesh as graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$



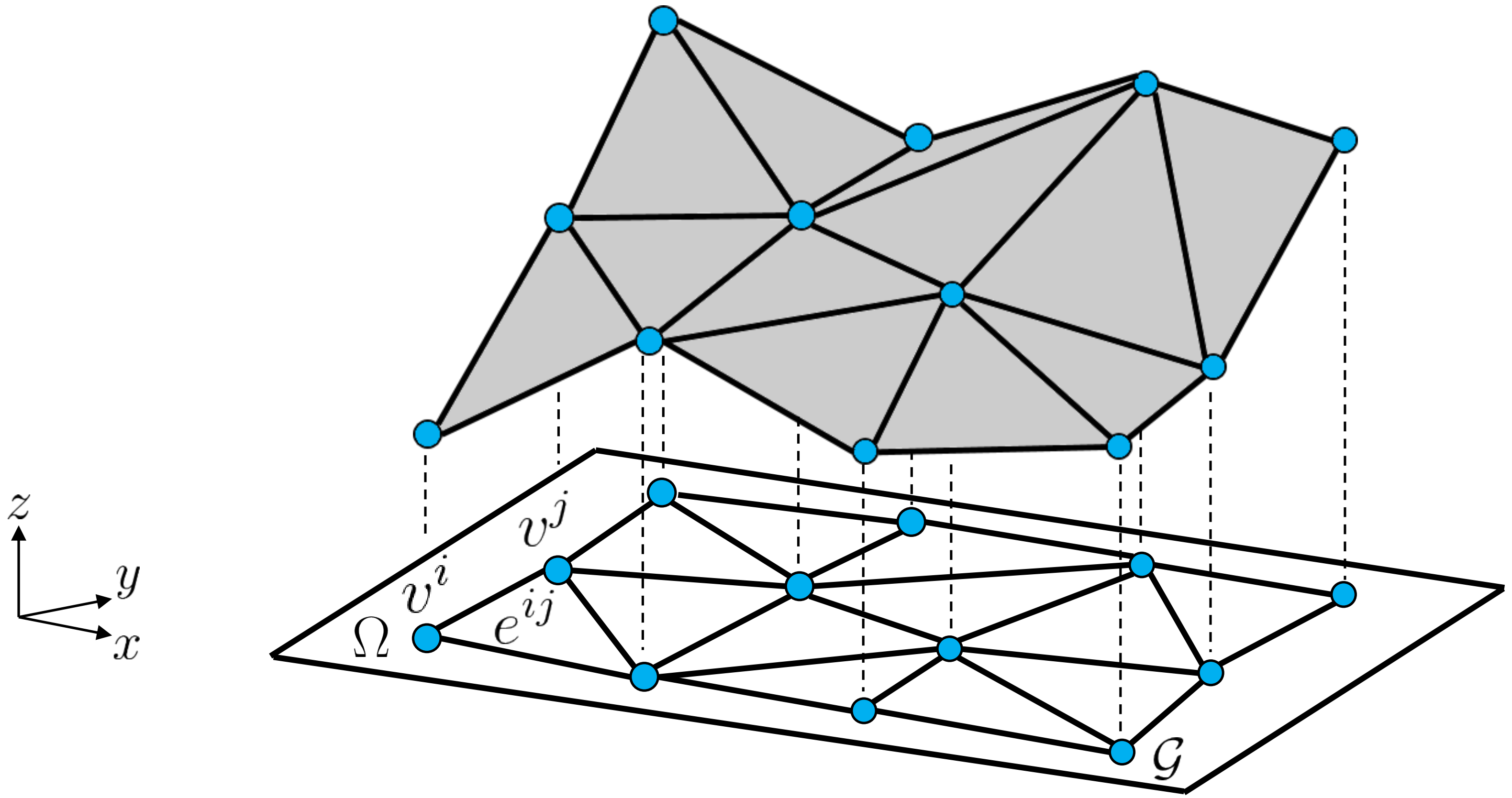
Approximate Cost over Mesh



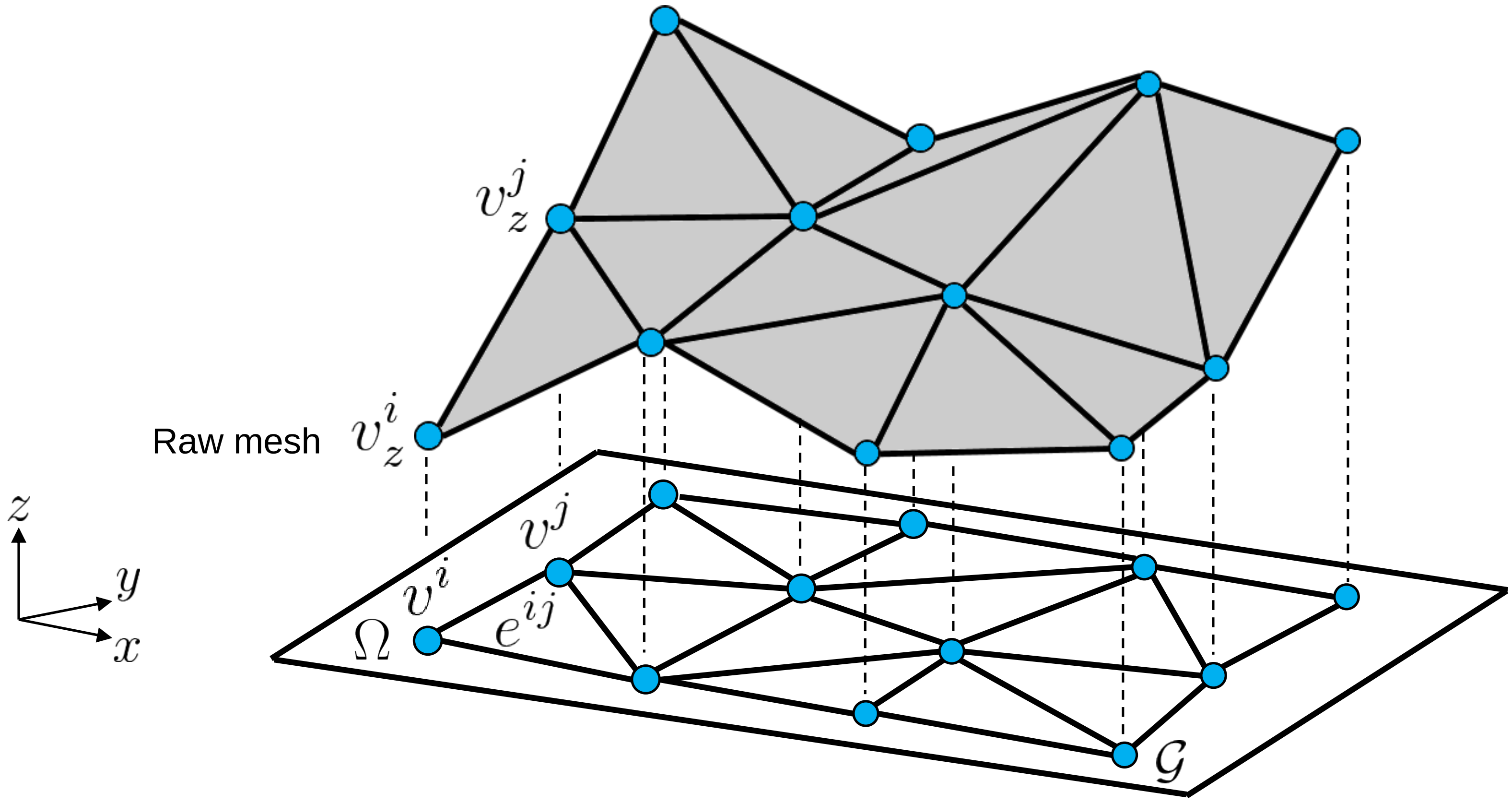
Approximate Cost over Mesh



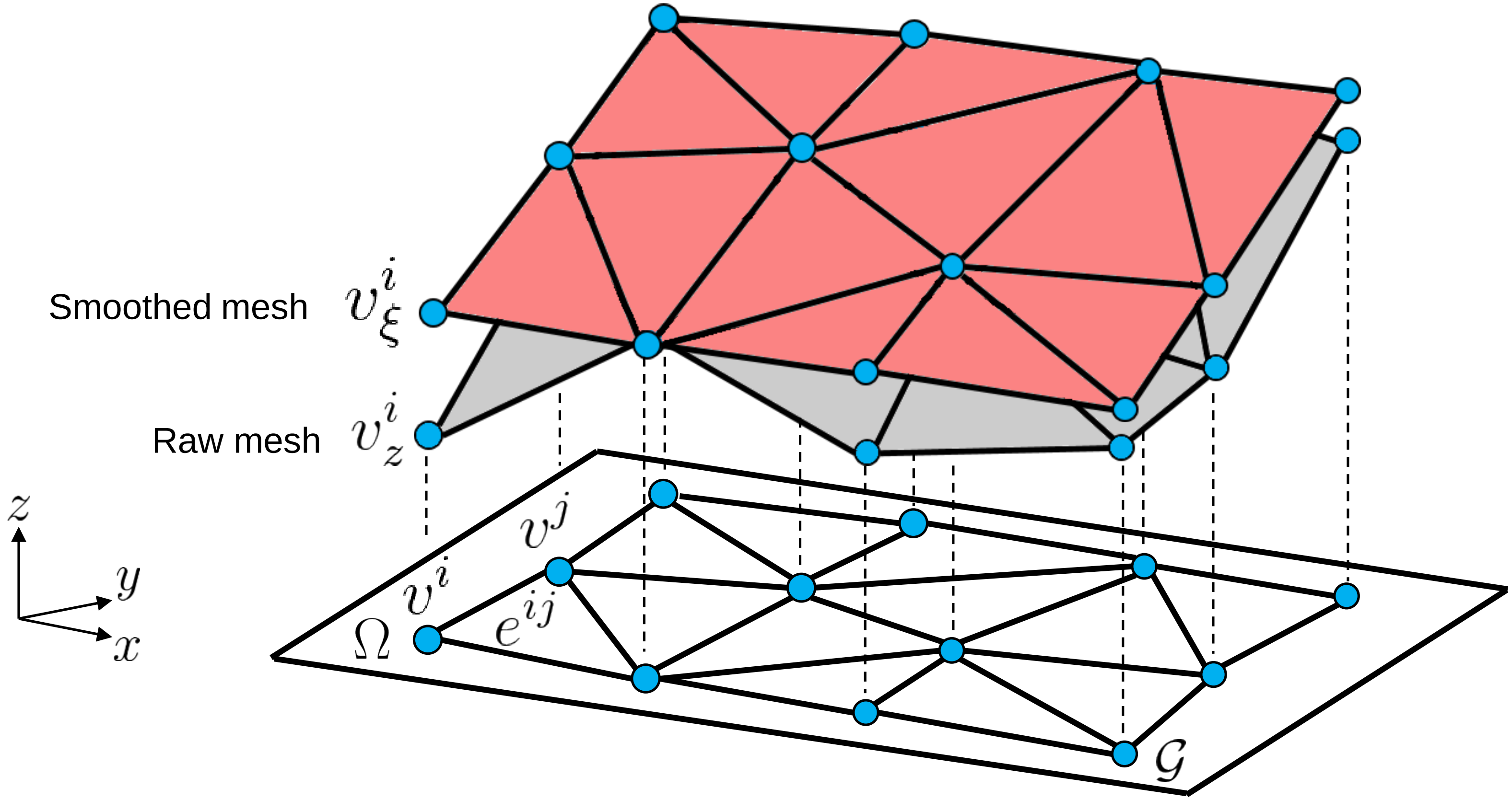
Approximate Cost over Mesh



Approximate Cost over Mesh

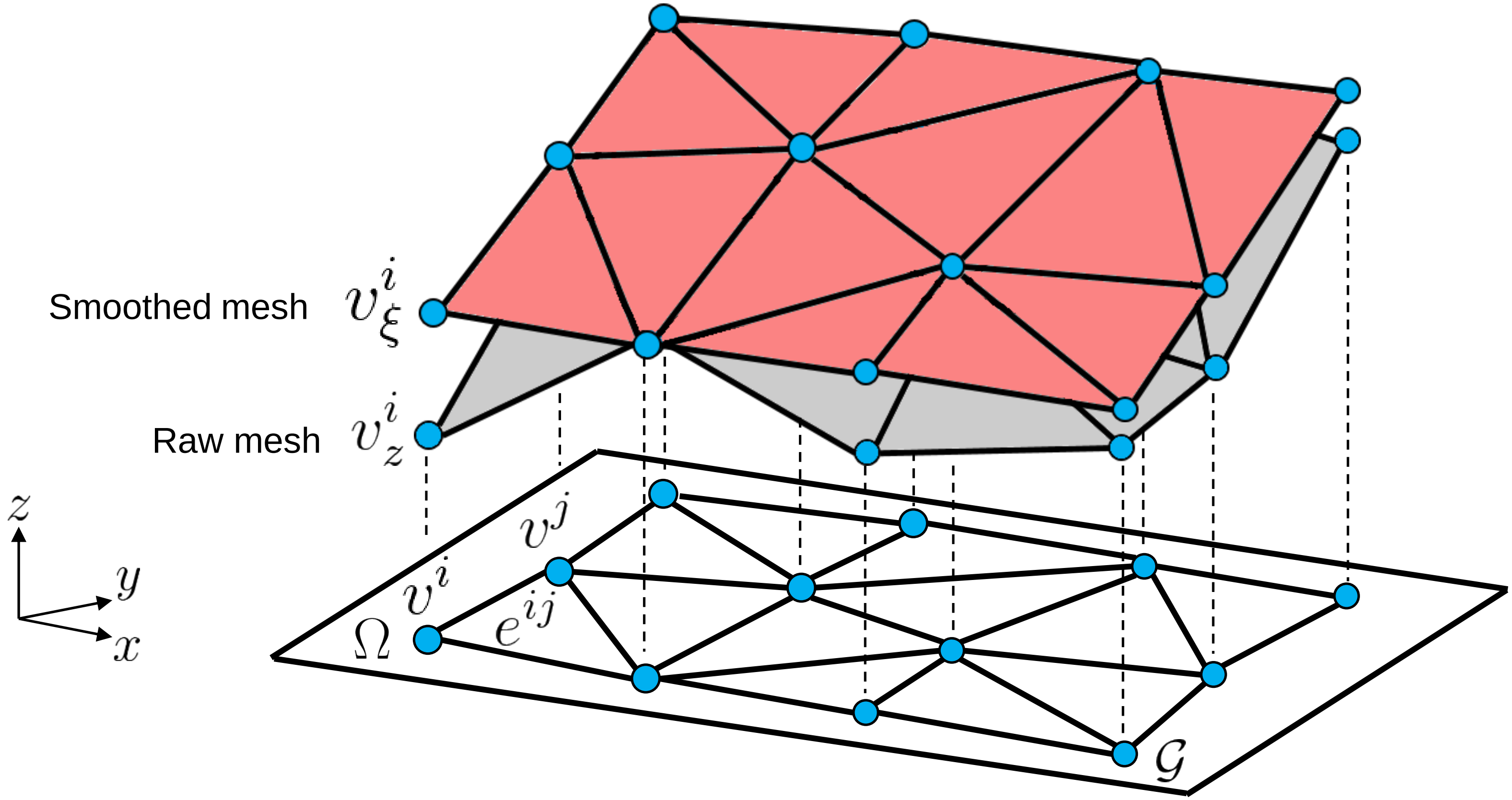


Approximate Cost over Mesh



Approximate Cost over Mesh

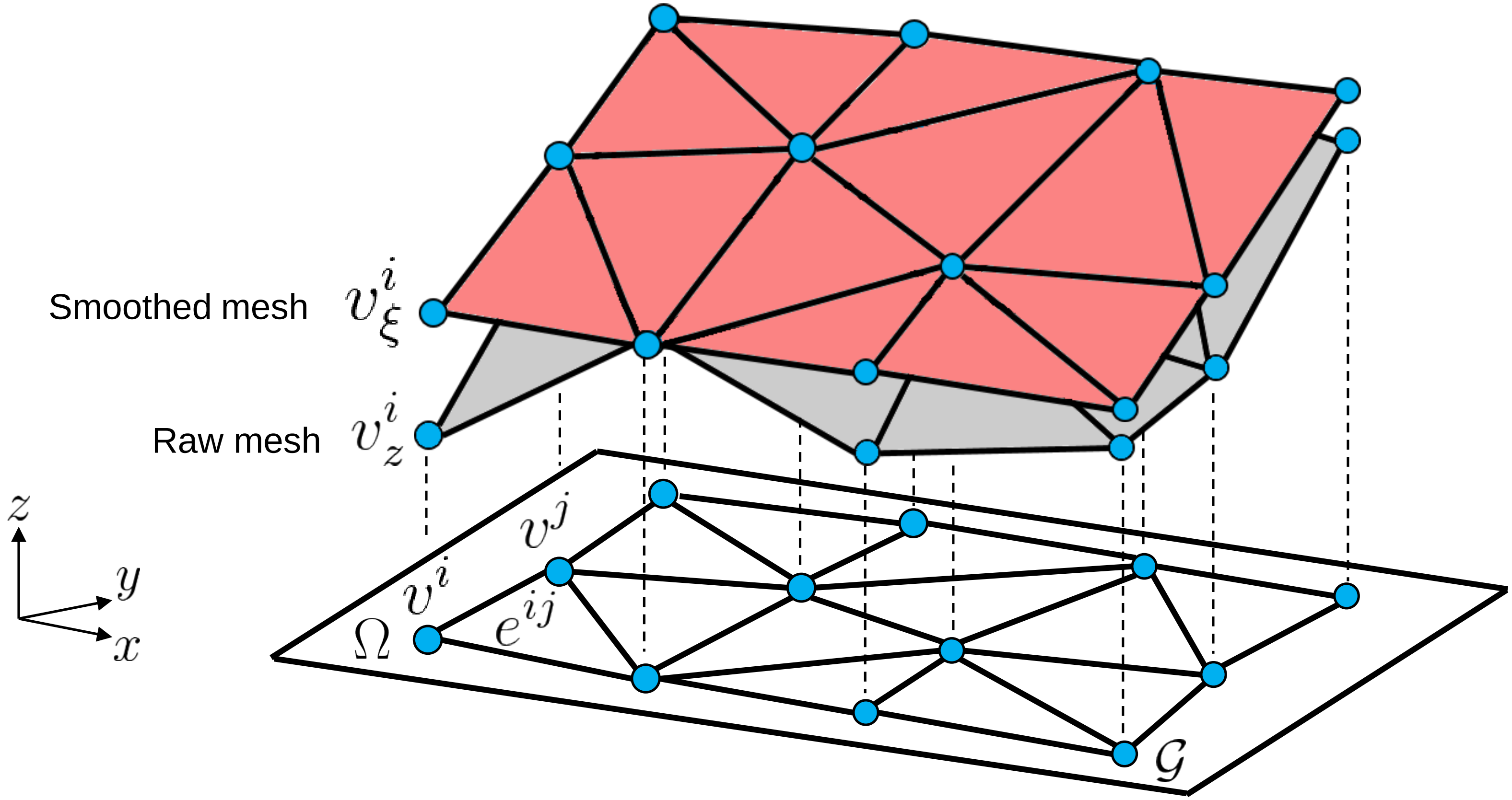
$$E(\xi) = \mathrm{NLTGV}^2(\xi) + \lambda \int_\Omega |\xi(\mathbf{u}) - z(\mathbf{u})| \, d\mathbf{u}$$



Approximate Cost over Mesh

$$E(\xi) = \mathrm{NLTGV}^2(\xi) + \lambda \int_\Omega |\xi(\mathbf{u}) - z(\mathbf{u})| \; d\mathbf{u}$$

Math…

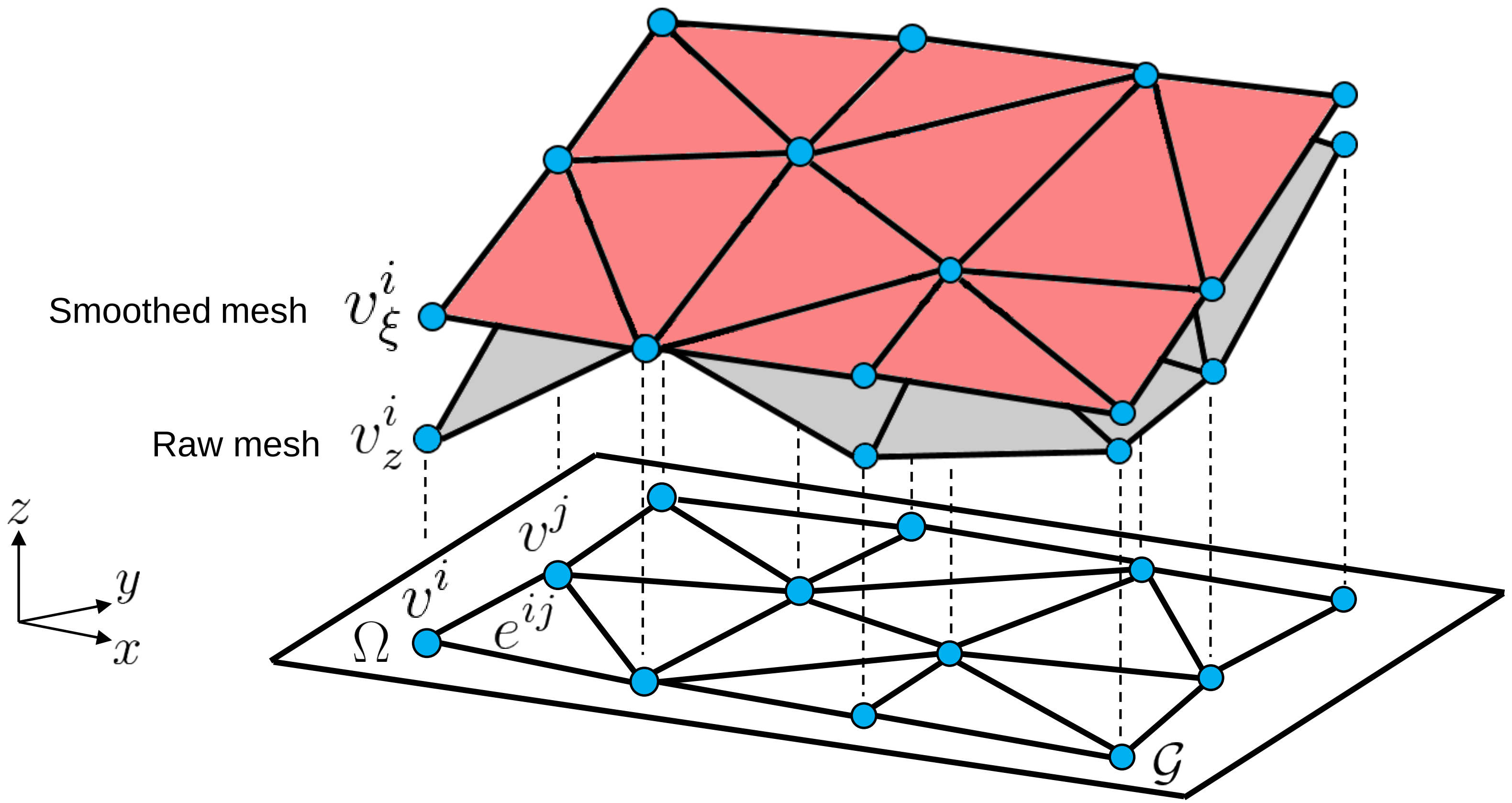


Approximate Cost over Mesh

$$E(\xi) = \mathrm{NLTGV}^2(\xi) + \lambda \int_\Omega |\xi(\mathbf{u}) - z(\mathbf{u})| \; d\mathbf{u}$$

Math…

$$E(\mathcal{G}) = \sum_{e \in \mathcal{E}} ||\mathbf{D}_e(v^i, v^j)||_1 + \lambda \sum_{v \in \mathcal{V}} |v_\xi - v_z|$$



Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G}) \xrightarrow{\text{Math...}}$$

Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

$\xrightarrow{\text{Math...}}$

Algorithm 2 NLTGV$^2 - L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0,1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \text{prox}_{F^*}\left(e_{\mathbf{q}}^n + \sigma \mathbf{D}_e(v_{\bar{\mathbf{x}}}^i, v_{\bar{\mathbf{x}}}^j)\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \text{prox}_G\left(v_{\mathbf{x}}^n - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^*(e_{\mathbf{q}}^{n+1}) - \tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^*(e_{\mathbf{q}}^{n+1})\right)$
    $v_{\bar{\mathbf{x}}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^n\right)$

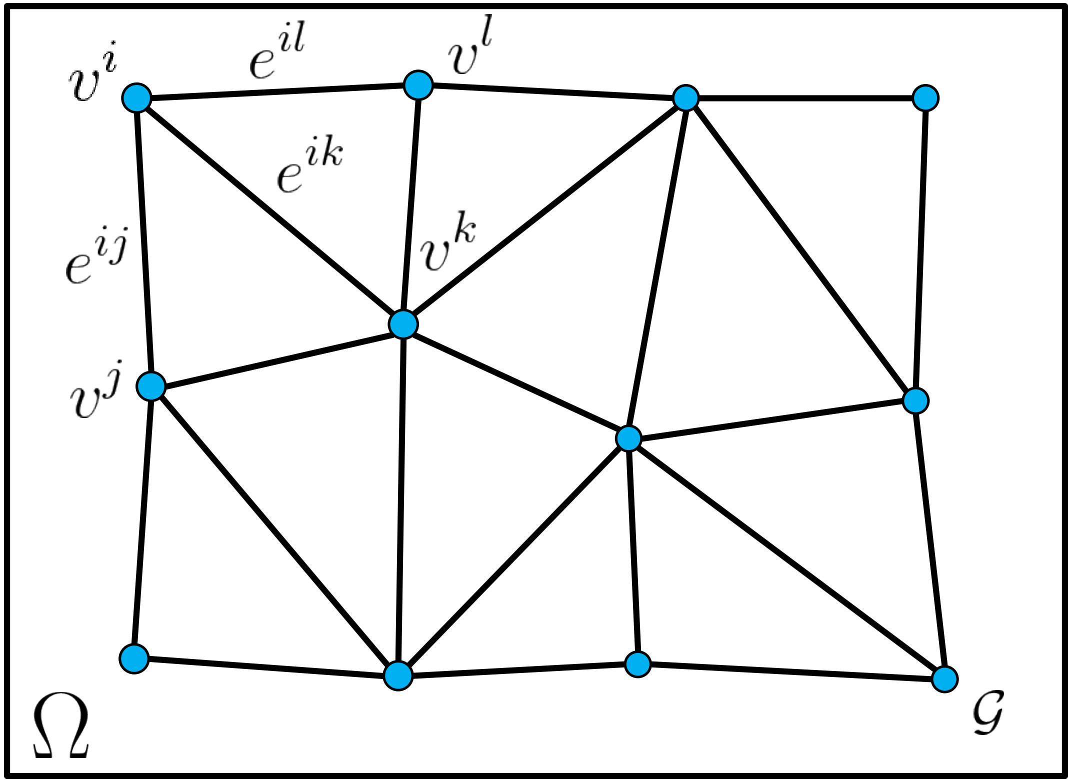
Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Math…

Algorithm 2 NLTGV$^2 - L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0, 1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \text{prox}_{F^*}\left(e_{\mathbf{q}}^n + \sigma \mathbf{D}_e(v_{\bar{\mathbf{x}}}^i, v_{\bar{\mathbf{x}}}^j)\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \text{prox}_G\left(v_{\mathbf{x}}^n - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^*(e_{\mathbf{q}}^{n+1}) - \tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^*(e_{\mathbf{q}}^{n+1})\right)$
    $v_{\bar{\mathbf{x}}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^n\right)$



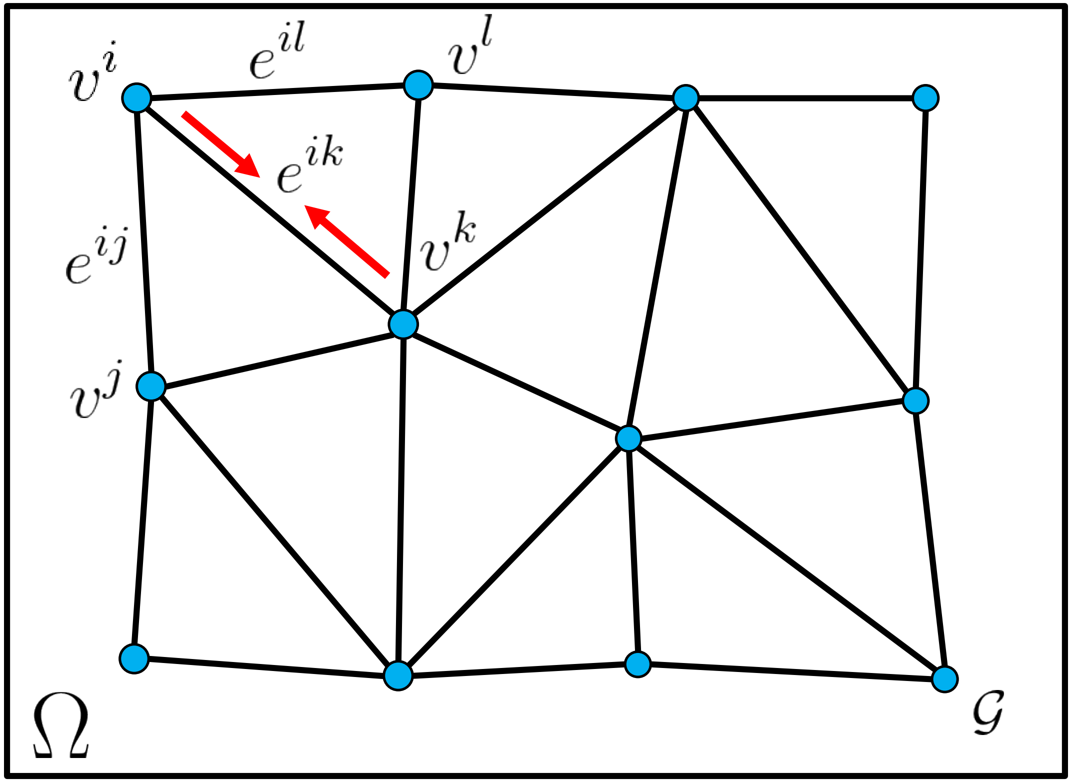
Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Math…

Algorithm 2 NLTGV$^2 - L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0,1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \mathrm{prox}_{F^*}\left(e_{\mathbf{q}}^{n} + \sigma \mathbf{D}_e(v_{\bar{\mathbf{x}}}^{i}, v_{\bar{\mathbf{x}}}^{j})\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \mathrm{prox}_G\left(v_{\mathbf{x}}^{n} - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^{*}(e_{\mathbf{q}}^{n+1}) - \tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^{*}(e_{\mathbf{q}}^{n+1})\right)$
    $v_{\bar{\mathbf{x}}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^{n}\right)$



Edges update using vertices

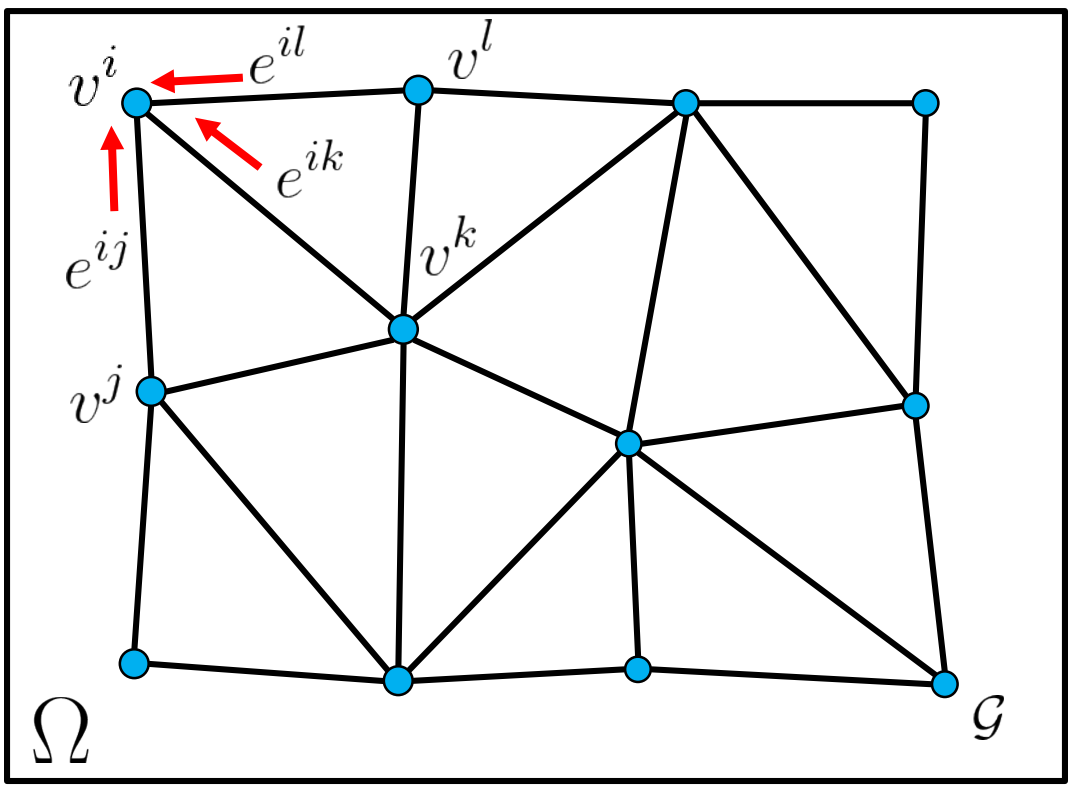
Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Math…

Algorithm 2 NLTGV$^2 - L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0, 1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \text{prox}_{F^*}\left(e_{\mathbf{q}}^{n} + \sigma \mathbf{D}_e(v_{\bar{\mathbf{x}}}^{i}, v_{\bar{\mathbf{x}}}^{j})\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \text{prox}_G\Big(v_{\mathbf{x}}^{n} - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^{*}(e_{\mathbf{q}}^{n+1})$
      $-\tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^{*}(e_{\mathbf{q}}^{n+1})\Big)$
    $v_{\bar{\mathbf{x}}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^{n}\right)$



Vertices update using edges

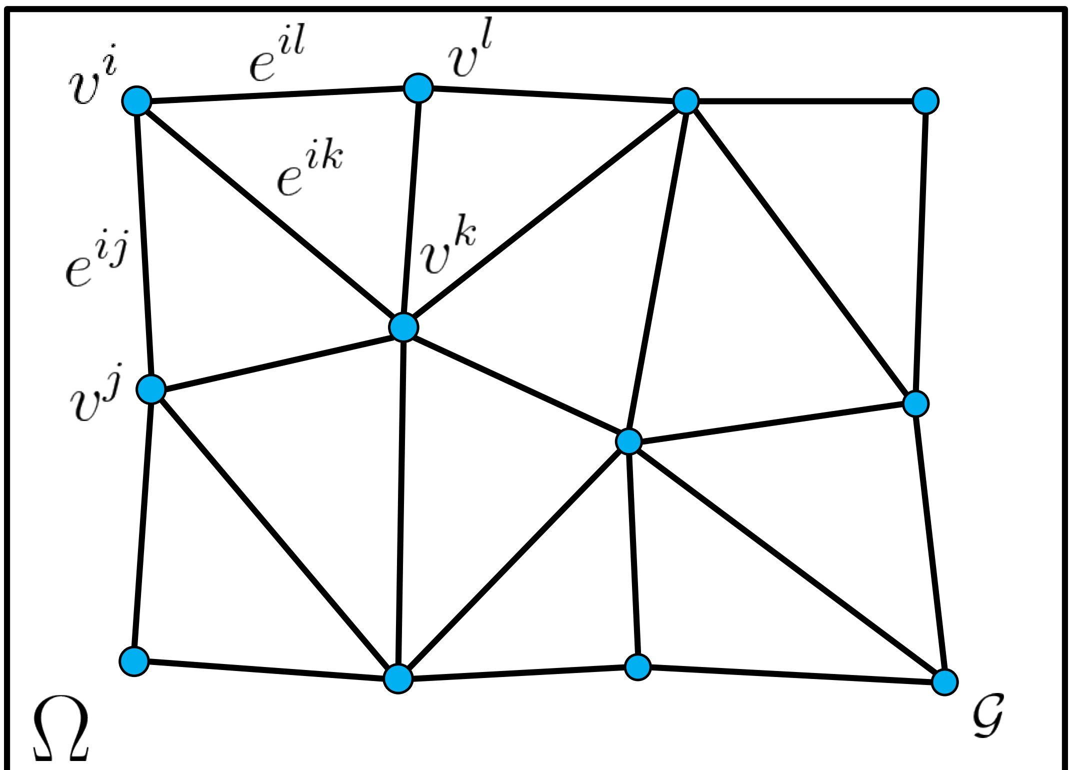
Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Math…

Algorithm 2 NLTGV$^2 - L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0,1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \text{prox}_{F^*}\left(e_{\mathbf{q}}^n + \sigma \mathbf{D}_e(v_{\mathbf{x}}^i, v_{\mathbf{x}}^j)\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \text{prox}_G\left(v_{\mathbf{x}}^n - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^*(e_{\mathbf{q}}^{n+1}) - \tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^*(e_{\mathbf{q}}^{n+1})\right)$
    $v_{\mathbf{x}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^n\right)$



Optimization steps are fast even without GPU (<< frame rate)

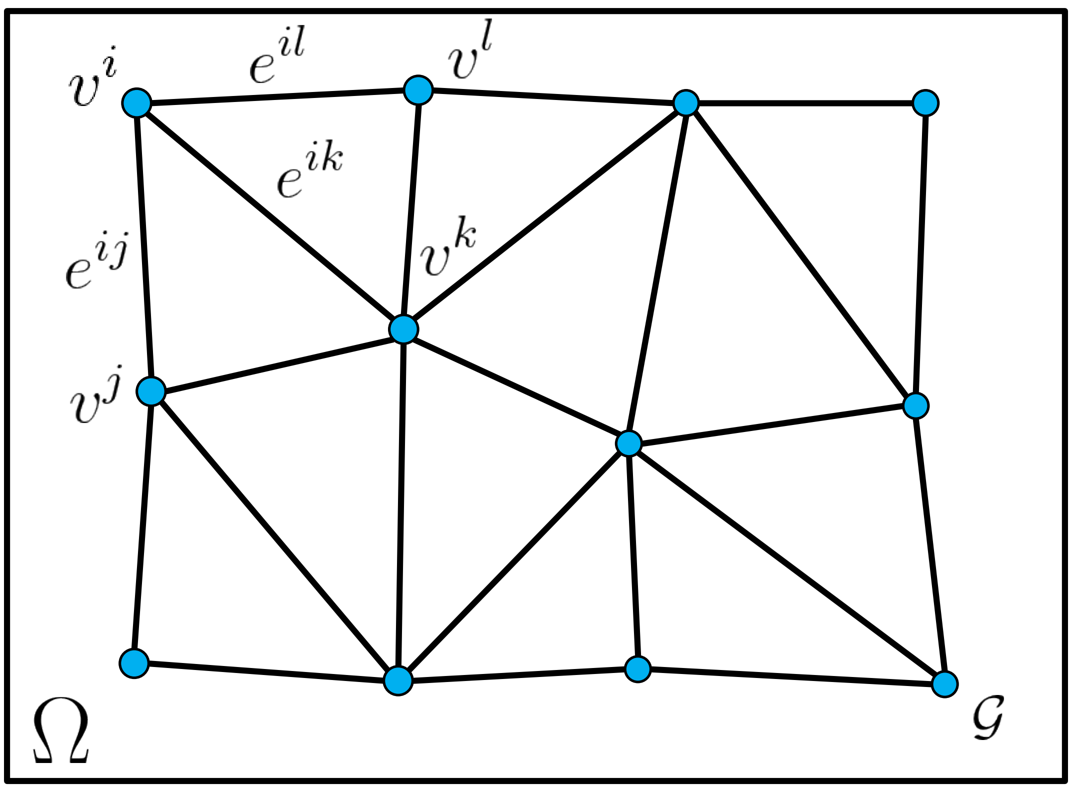
Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Math…

Algorithm 2 NLTGV^2 $- L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0,1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \text{prox}_{F^*}\left(e_{\mathbf{q}}^n + \sigma \mathbf{D}_e(v_{\mathbf{x}}^i, v_{\mathbf{x}}^j)\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \text{prox}_G\Big(v_{\mathbf{x}}^n - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^*(e_{\mathbf{q}}^{n+1})$
      $- \tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^*(e_{\mathbf{q}}^{n+1})\Big)$
    $v_{\bar{\mathbf{x}}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^n\right)$



Optimization steps are fast even without GPU (<< frame rate)

Optimization convergence is fast (~ frame rate)

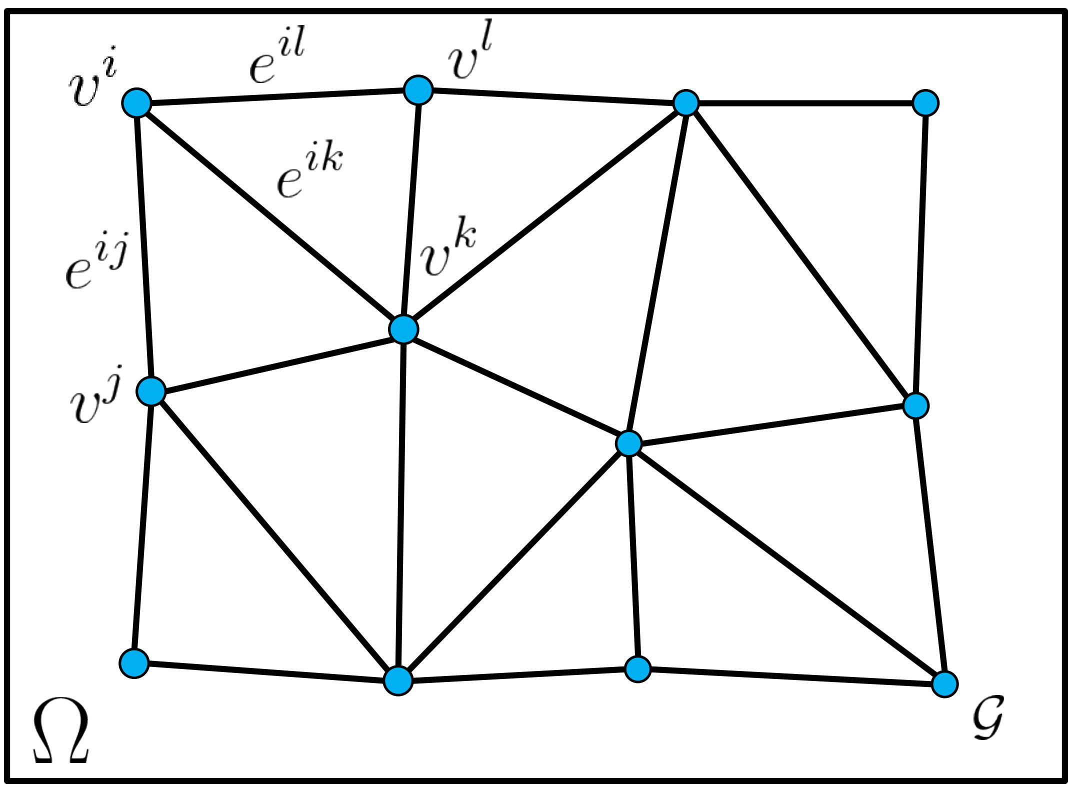
Optimize using Primal-Dual on Graph

$$\min E(\mathcal{G})$$

Math…

Algorithm 2 NLTGV$^2 - L_1$ Graph Optimization

// Choose $\sigma, \tau > 0, \theta \in [0,1]$.
while not converged **do**
  for each $e \in \mathcal{E}_k$ **do**
    $e_{\mathbf{q}}^{n+1} = \mathrm{prox}_{F^*}\left(e_{\mathbf{q}}^{n} + \sigma \mathbf{D}_e(v_{\bar{\mathbf{x}}}^{i}, v_{\bar{\mathbf{x}}}^{j})\right)$
  for each $v \in \mathcal{V}_k$ **do**
    $v_{\mathbf{x}}^{n+1} = \mathrm{prox}_{G}\left(v_{\mathbf{x}}^{n} - \tau \sum_{e \in \mathcal{N}_{in}(v)} \mathbf{D}_{in}^{*}(e_{\mathbf{q}}^{n+1}) - \tau \sum_{e \in \mathcal{N}_{out}(v)} \mathbf{D}_{out}^{*}(e_{\mathbf{q}}^{n+1})\right)$
    $v_{\bar{\mathbf{x}}}^{n+1} = v_{\mathbf{x}}^{n+1} + \theta\left(v_{\mathbf{x}}^{n+1} - v_{\mathbf{x}}^{n}\right)$

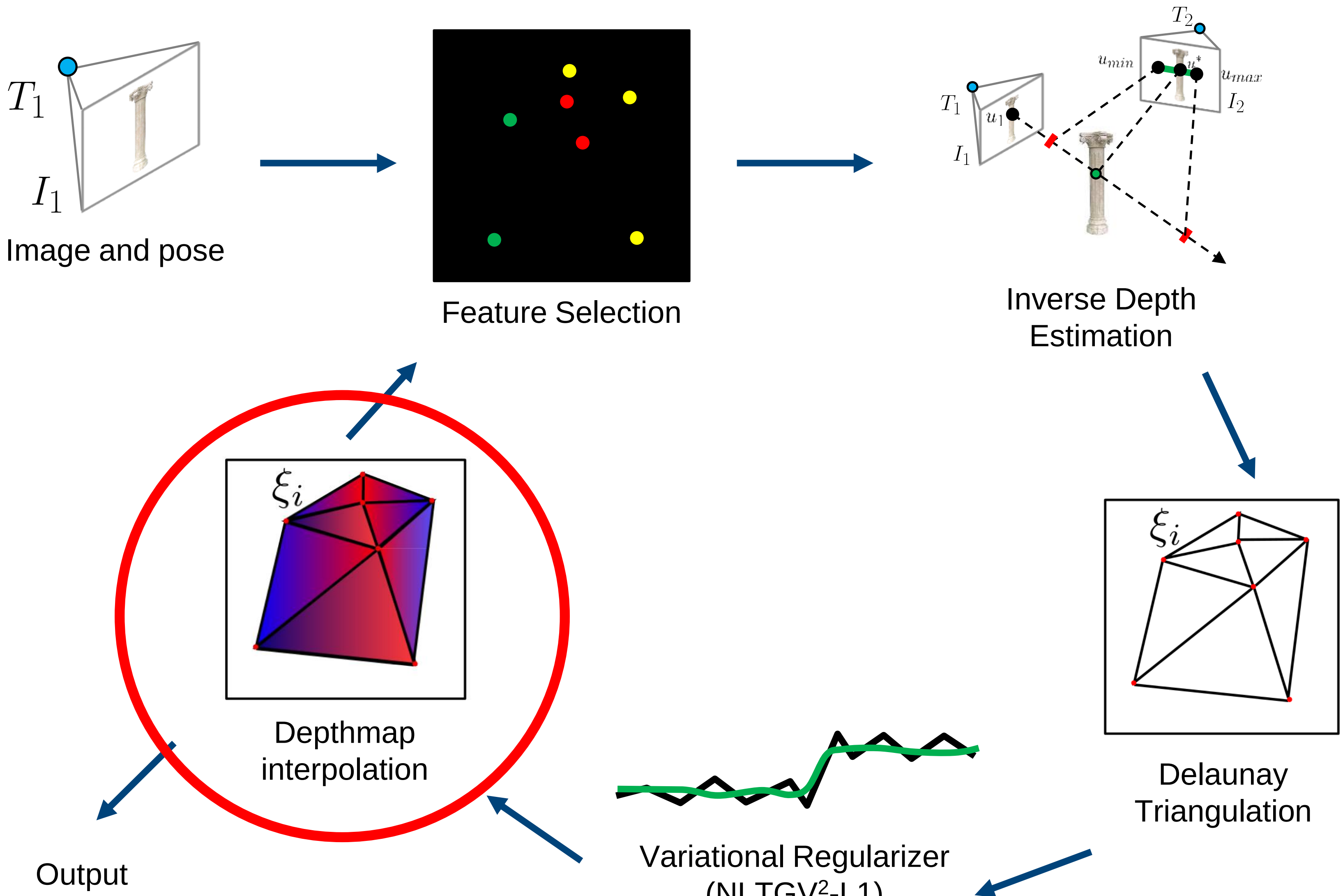


Optimization steps are fast even without GPU (<< frame rate)

Optimization convergence is fast (~ frame rate)

Mesh can be augmented without restarting optimization

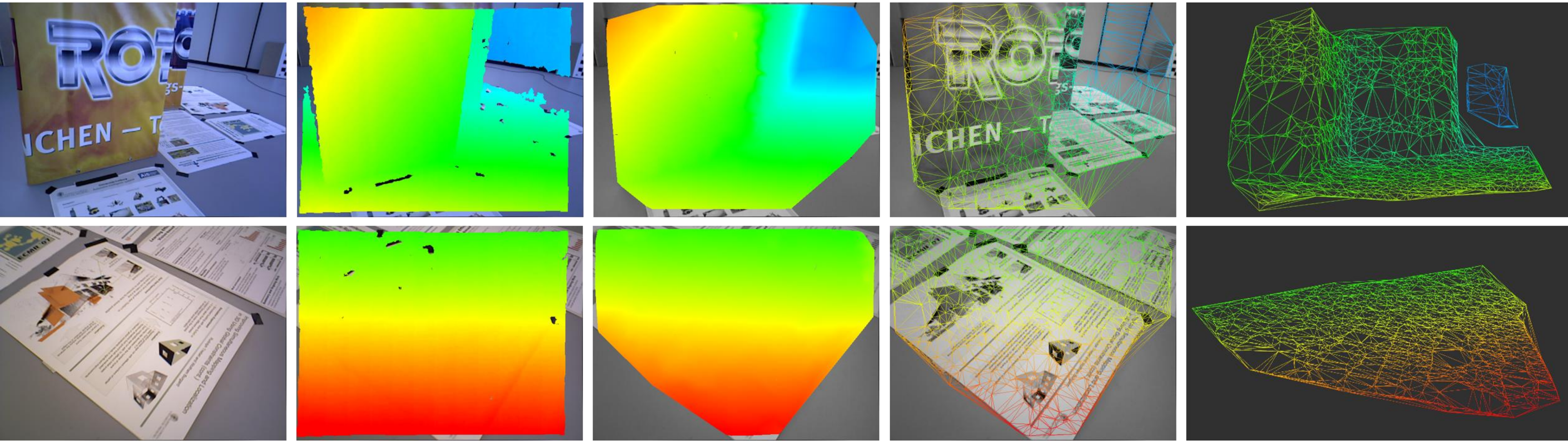
FLaME: Fast Lightweight Mesh Estimation



Benchmark Results

Compared FLaME to two existing CPU-only approaches:

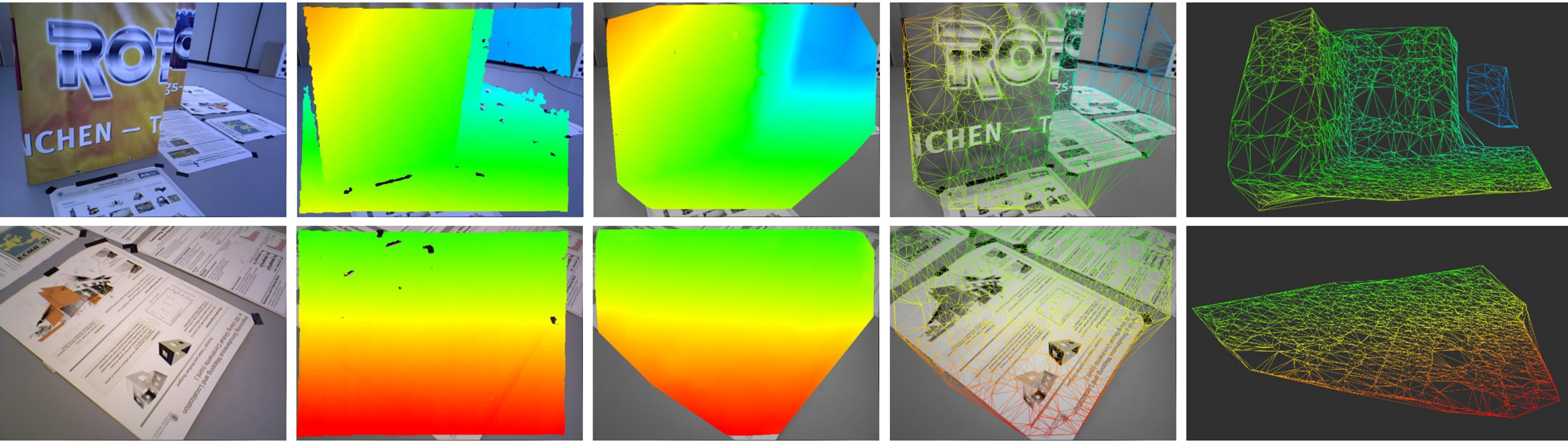
- LSD-SLAM
- MLM



Benchmark Results

Compared FLaME to two existing CPU-only approaches:
- LSD-SLAM
- MLM

Evaluated on two benchmark datasets (ground truth poses):
- TUM RGBD SLAM (640x480 @ 30 Hz)
- EuRoC MAV (752x480 @ 20 Hz)

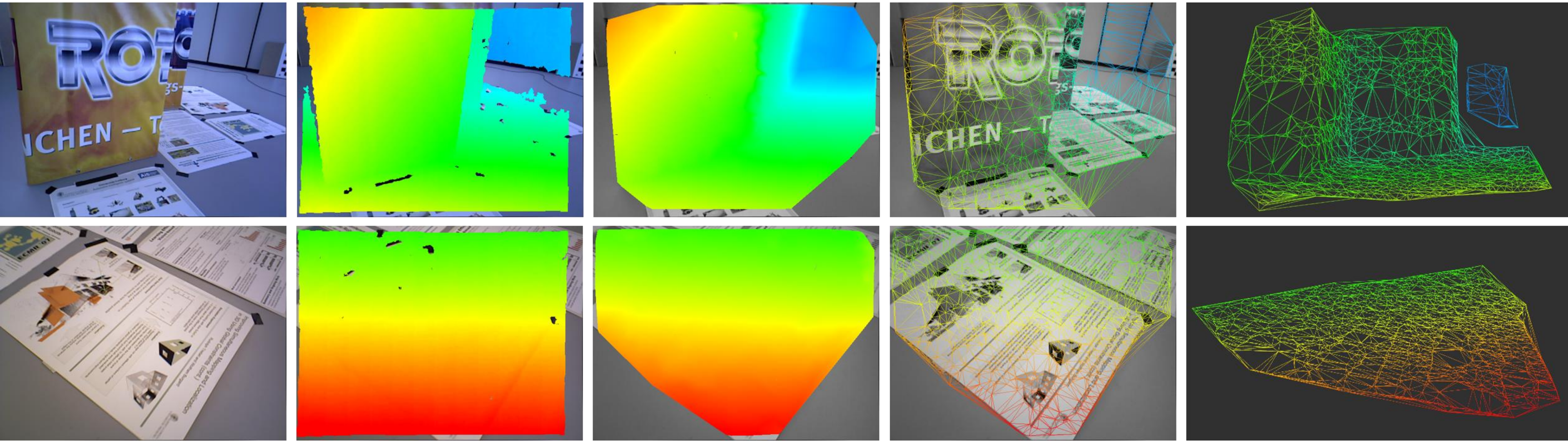


Benchmark Results

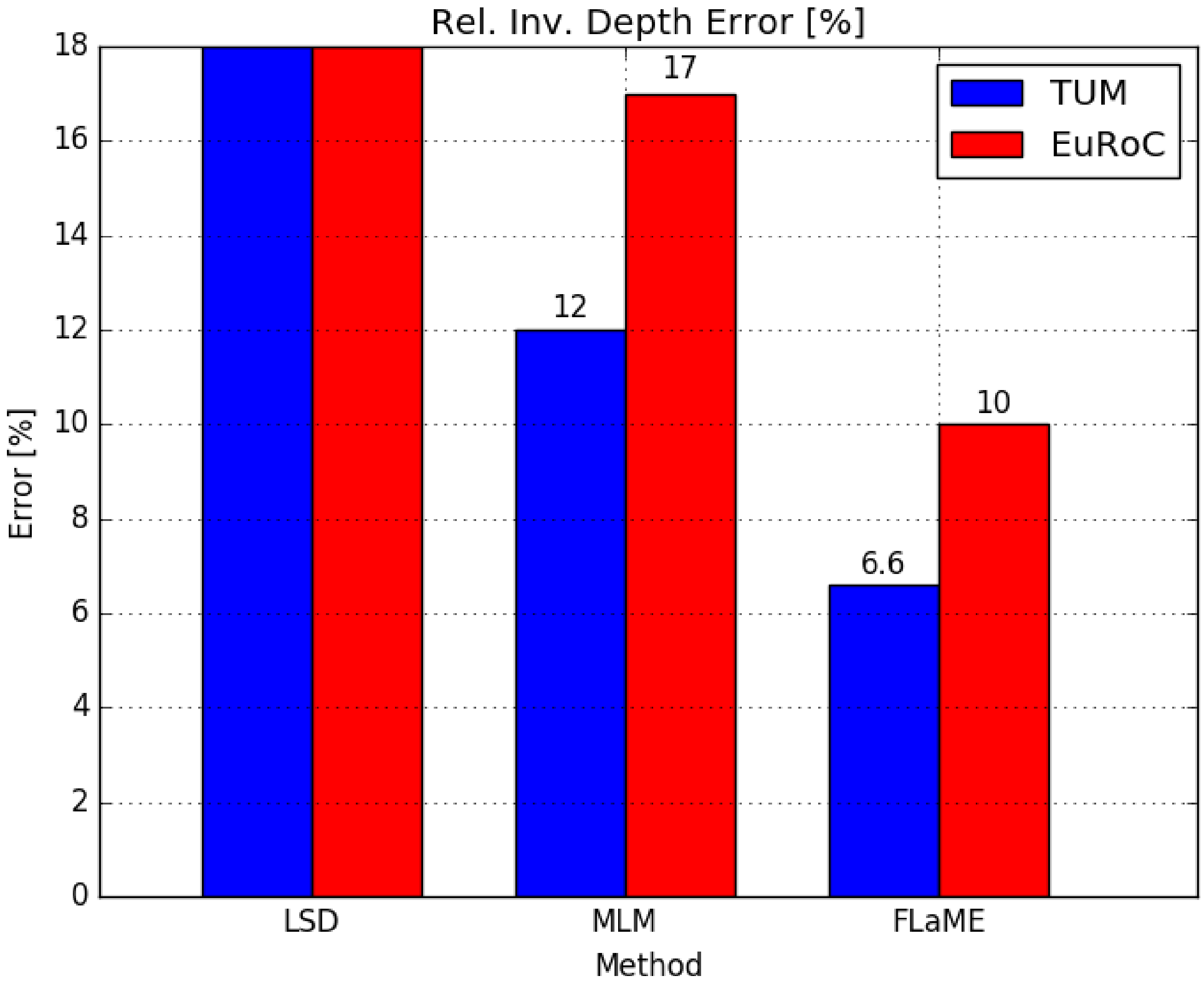
Compared FLaME to two existing CPU-only approaches:
- LSD-SLAM
- MLM

Evaluated on two benchmark datasets (ground truth poses):
- TUM RGBD SLAM (640x480 @ 30 Hz)
- EuRoC MAV (752x480 @ 20 Hz)

Desktop Intel i7 CPU only

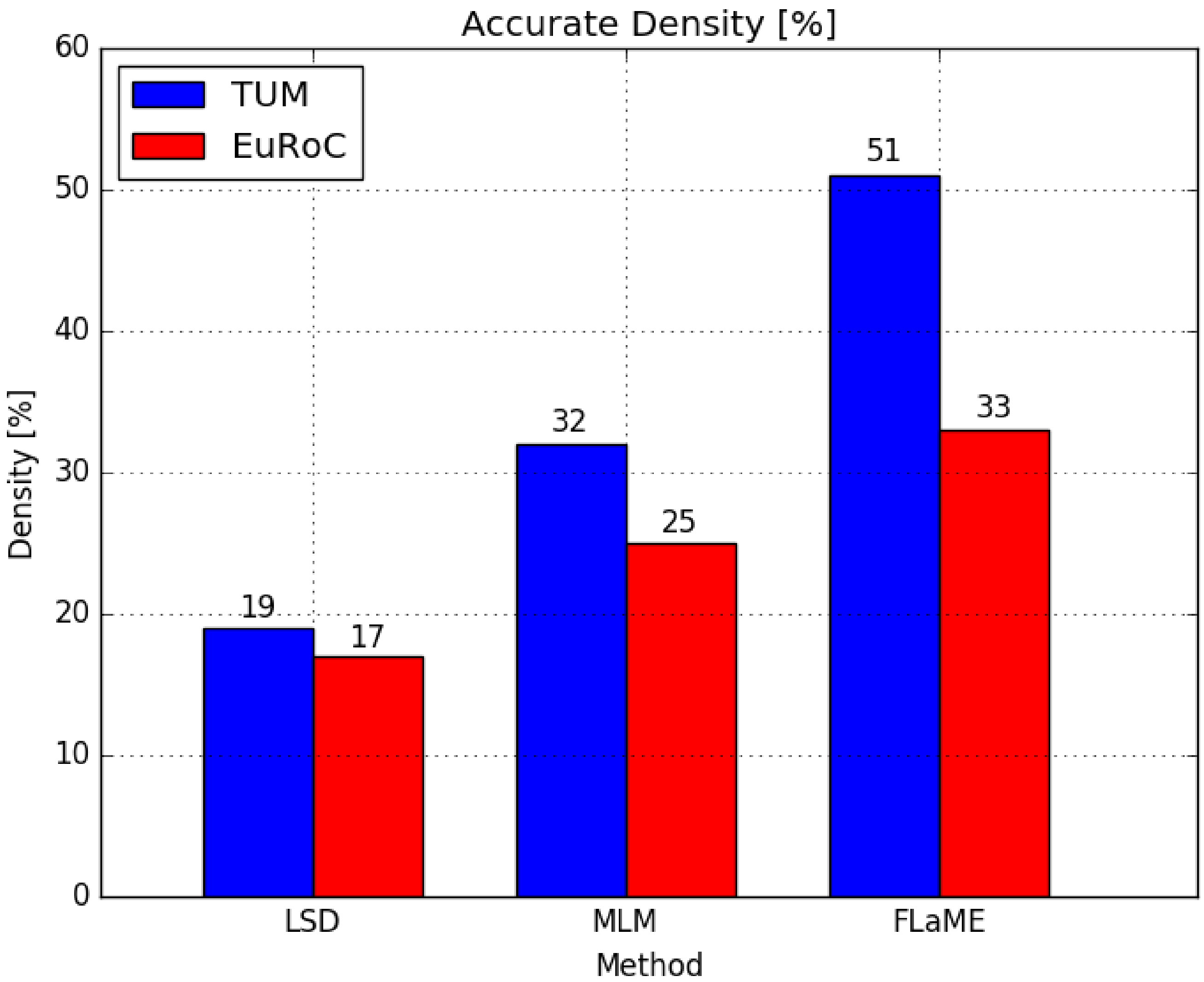


Benchmark Results



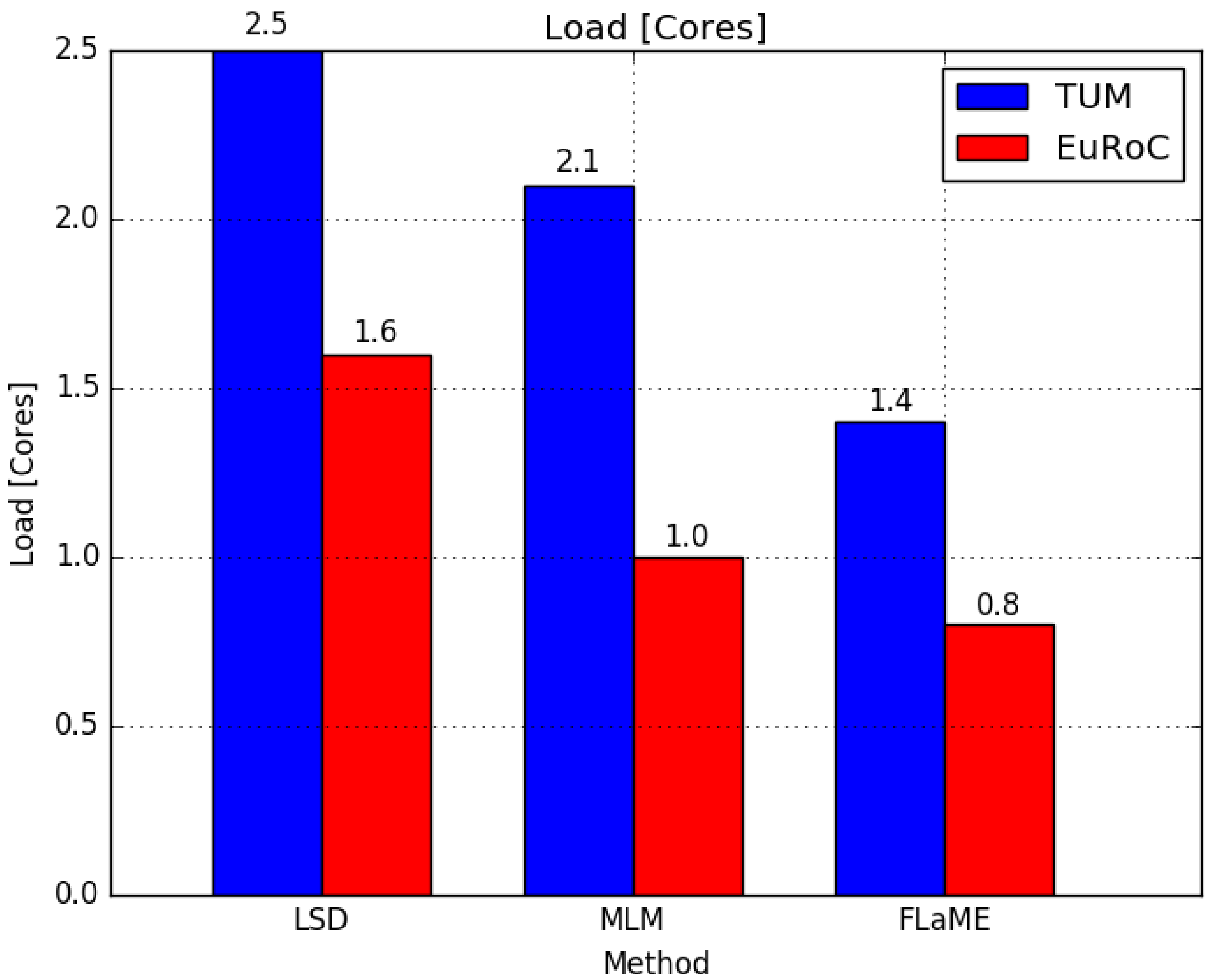
Lower Error

Benchmark Results



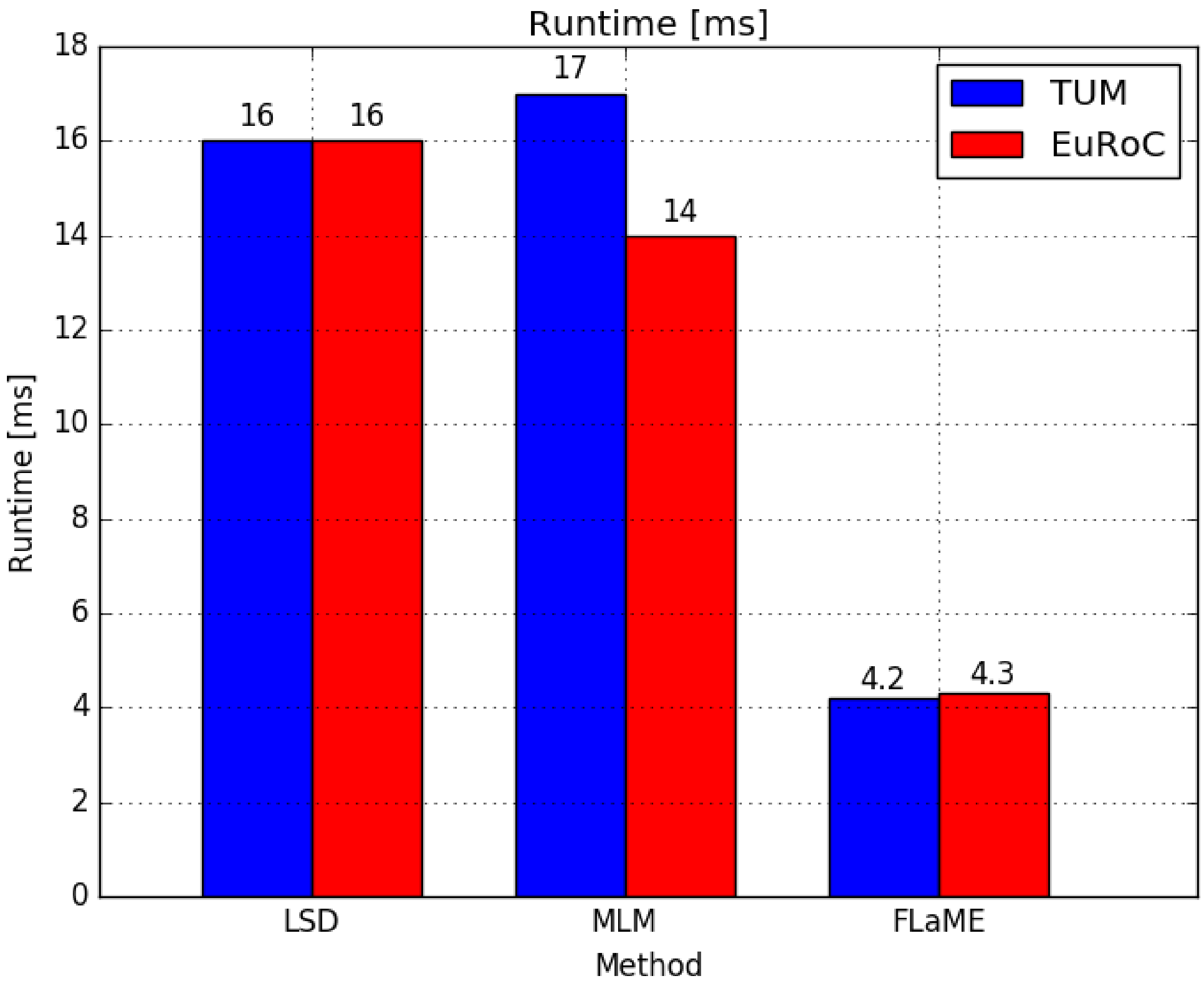
More Dense

Benchmark Results



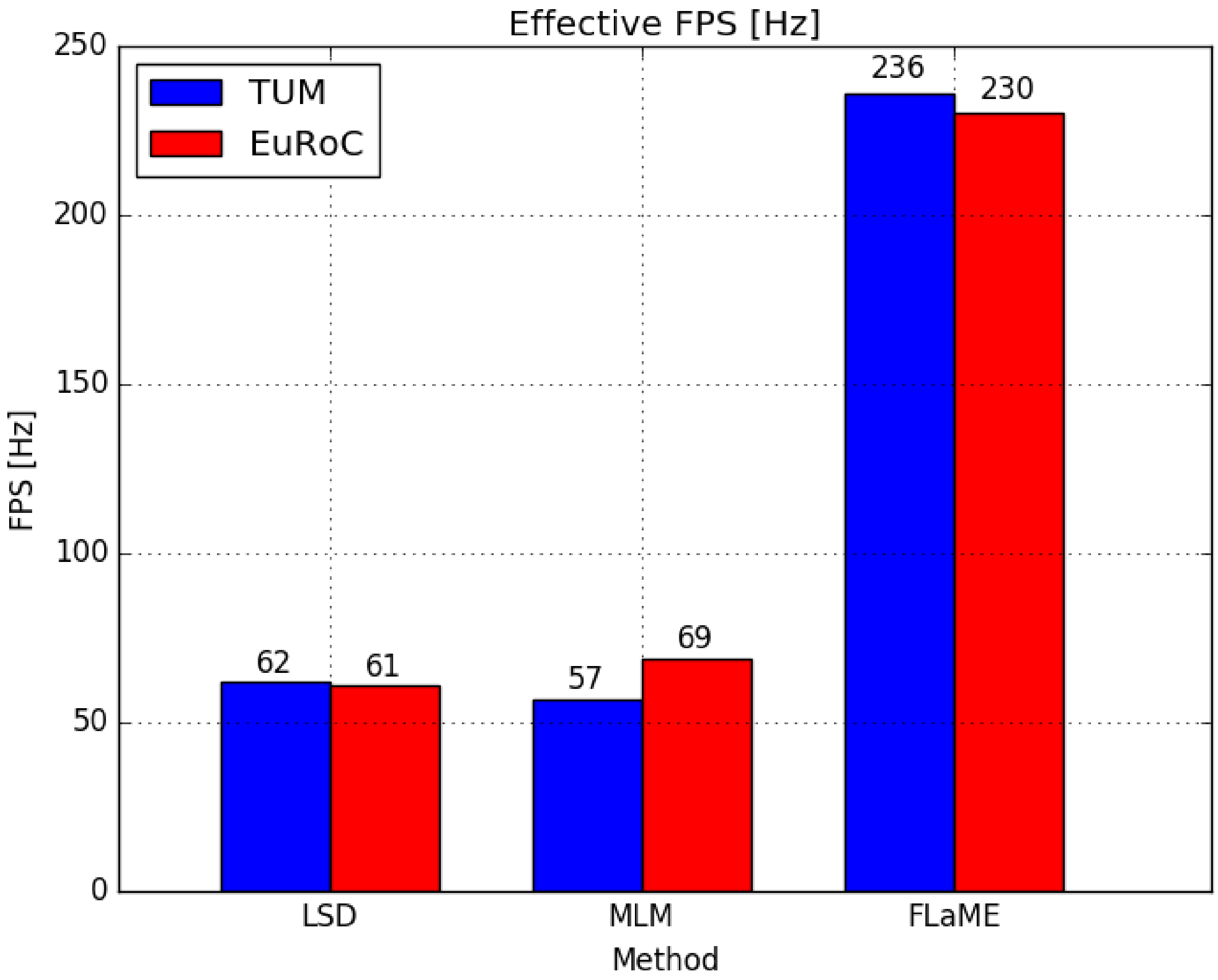
Less Load

Benchmark Results



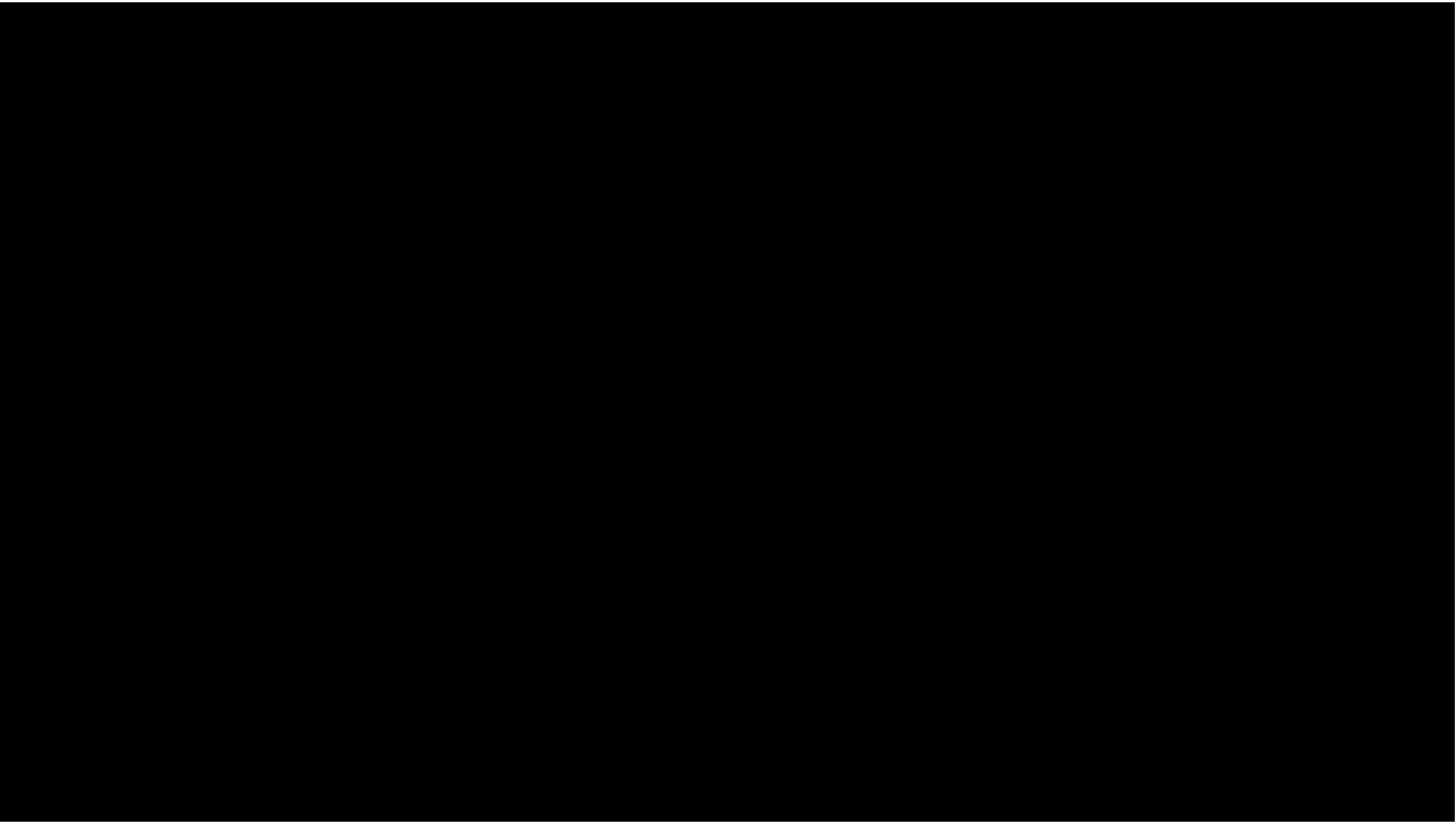
Faster

Benchmark Results

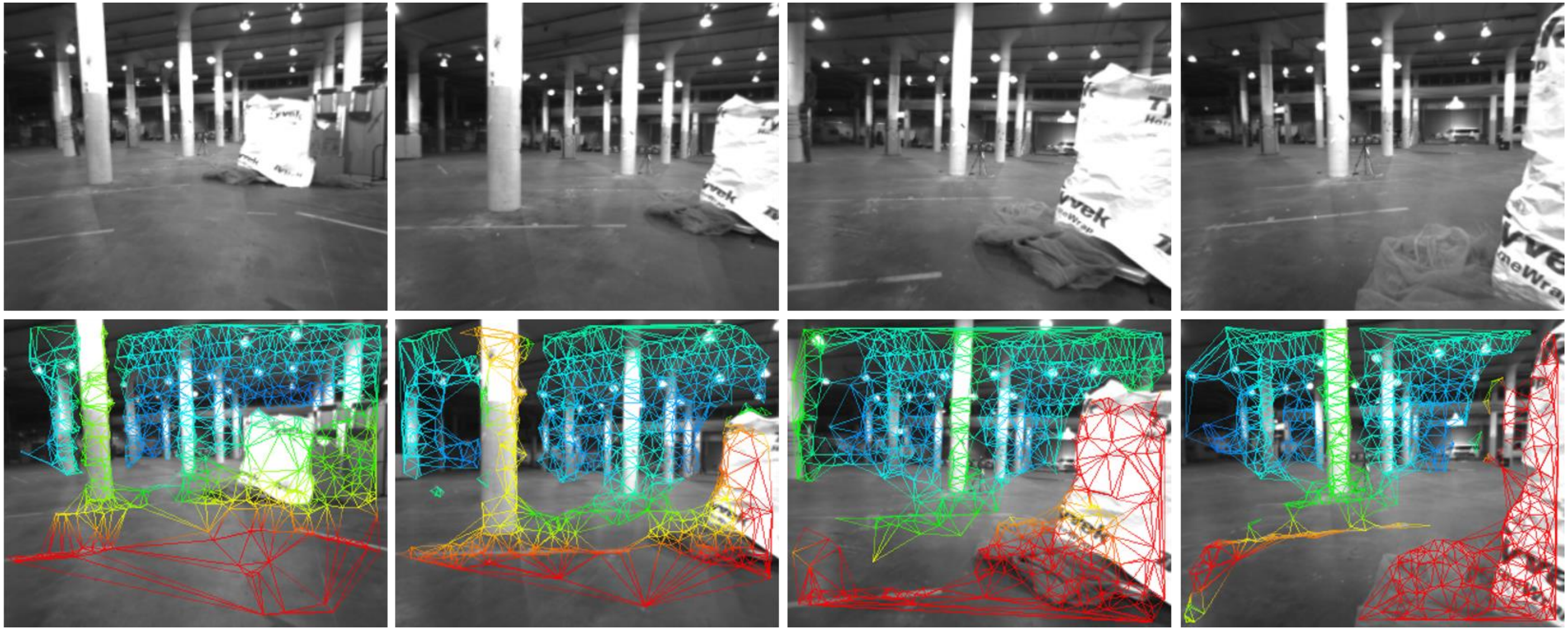


Faster

Benchmark Results

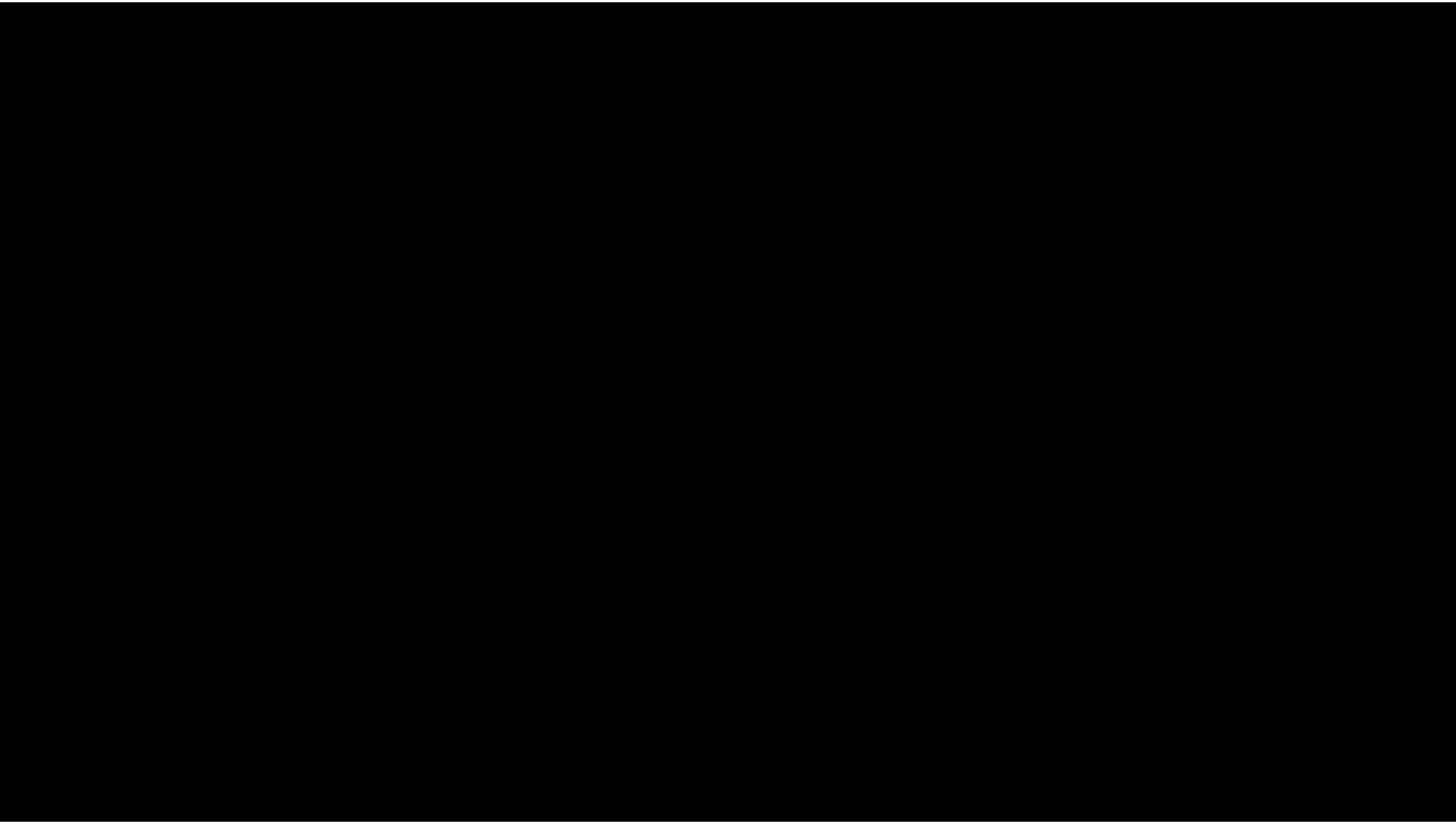


Flight Experiments



DJI F450 frame
Intel NUC flight computer
10 ms/frame runtime
1.5 core load
Indoor Flight at 2.5 m/s
Outdoor Flight at 3.5 m/s

Flight Experiments



FLaME Contributions

- Proposed a lightweight method for dense online monocular depth estimation

Paper

Code*

W. Nicholas Greene (wng@csail.mit.edu)

* Coming soon!

FLaME Contributions

- Proposed a lightweight method for dense online monocular depth estimation
- Formulated the reconstruction problem as a non-local variational optimization over a Delaunay graph, which allows for a fast, efficient approach to depth estimation

Paper

Code*

* Coming soon!

W. Nicholas Greene (wng@csail.mit.edu)

FLaME Contributions

- Proposed a lightweight method for dense online monocular depth estimation
- Formulated the reconstruction problem as a non-local variational optimization over a Delaunay graph, which allows for a fast, efficient approach to depth estimation
- Demonstrated improved depth accuracy and density on benchmark datasets using less than 1 Intel i7 CPU core

Paper

Code*

W. Nicholas Greene (wng@csail.mit.edu)

* Coming soon!

FLaME Contributions

- Proposed a lightweight method for dense online monocular depth estimation
- Formulated the reconstruction problem as a non-local variational optimization over a Delaunay graph, which allows for a fast, efficient approach to depth estimation
- Demonstrated improved depth accuracy and density on benchmark datasets using less than 1 Intel i7 CPU core
- Demonstrated real-world applicability with indoor and outdoor flight experiments running FLaME onboard, in-the-loop

Paper

Code*

W. Nicholas Greene (wng@csail.mit.edu)

* Coming soon!

Acknowledgments

This material is based upon work supported by DARPA under Contract No. HR0011-15-C-0110 and by the NSF Graduate Research Fellowship under Grant No. 1122374. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

We also thank John Ware, John Carter, and the rest of the MIT-Draper FLA Team for continued development and testing support.

Paper

Code*

* Coming soon!

W. Nicholas Greene (wng@csail.mit.edu)