CS 532: 3D Computer Vision
8^{th} Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

FRIDAY

- Make up class for last week
- 3:30 in Altorfer 501

Lecture Outline

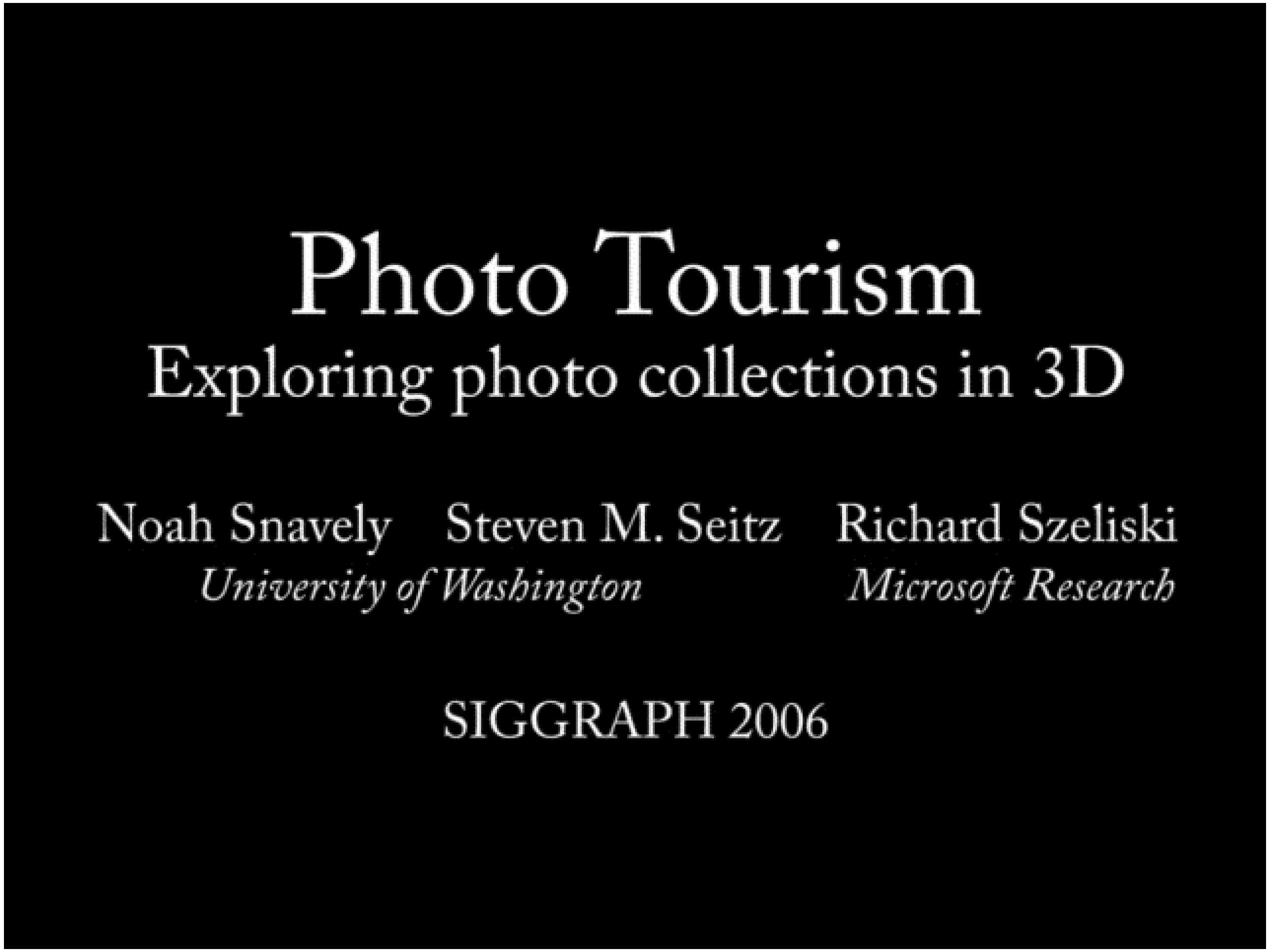
- Large-scale Structure from Motion
  - Image collections
- Multi-view Stereo part I

- Sources:
  - Slides by R. Szeliski, S. Seitz, N. Snavely, D. Gallup, C. Hernandez, G. Vogiatzis, Y. Furukawa, M. Bleyer

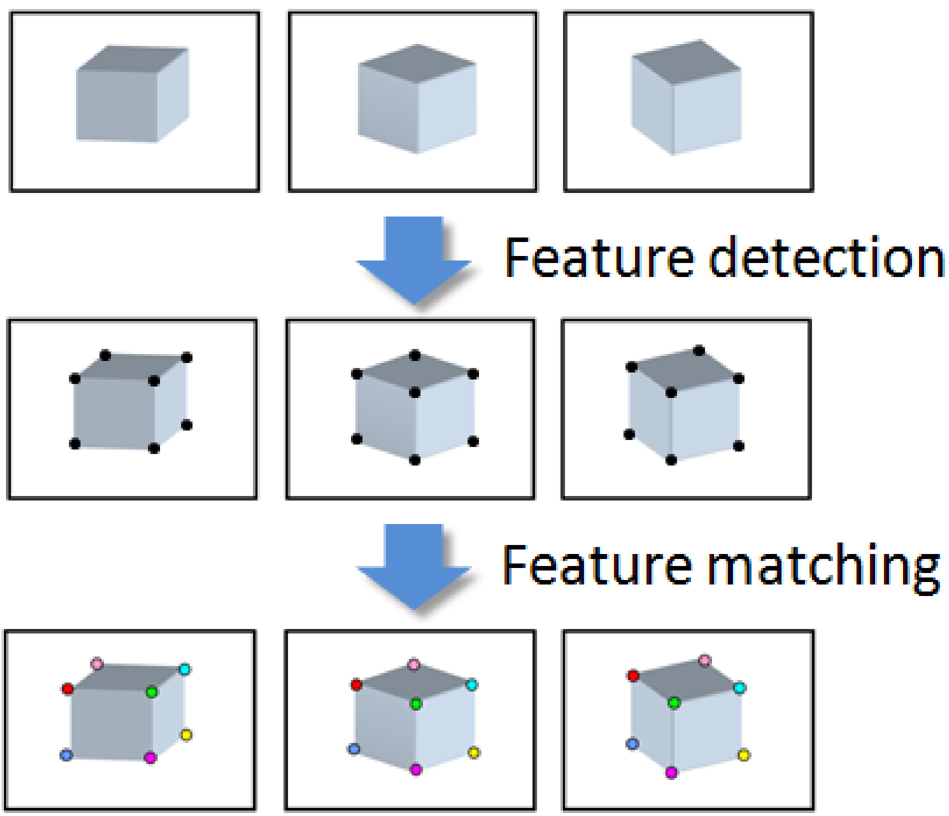
Structure from Motion from large Image Collections

- Given many images, how can we
  a) figure out where they were all taken from?
  b) build a 3D model of the scene?

Photo Tourism



Input

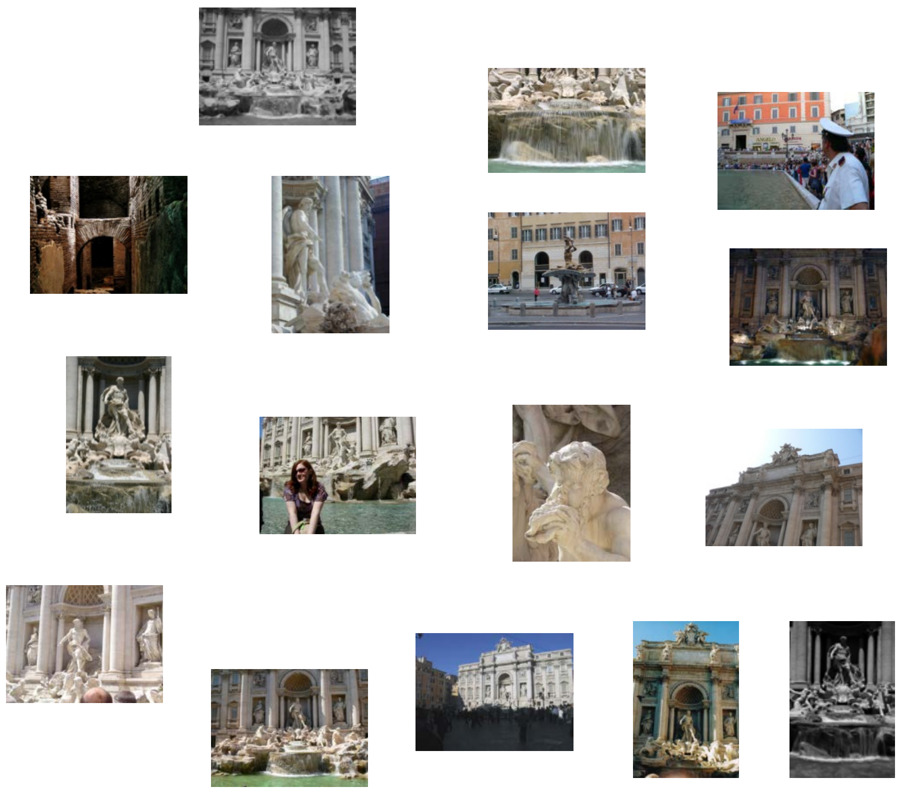


First Step: How to Get Correspondences?

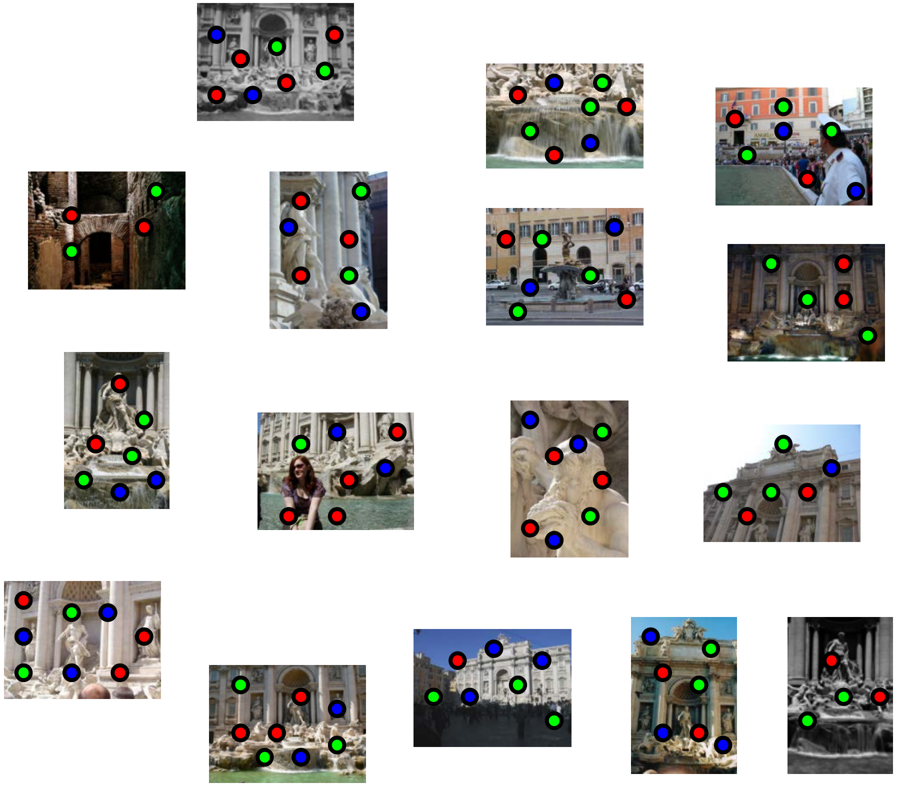
- Feature detection and matching

Feature detection

Detect features using SIFT [Lowe, IJCV 2004] in all images

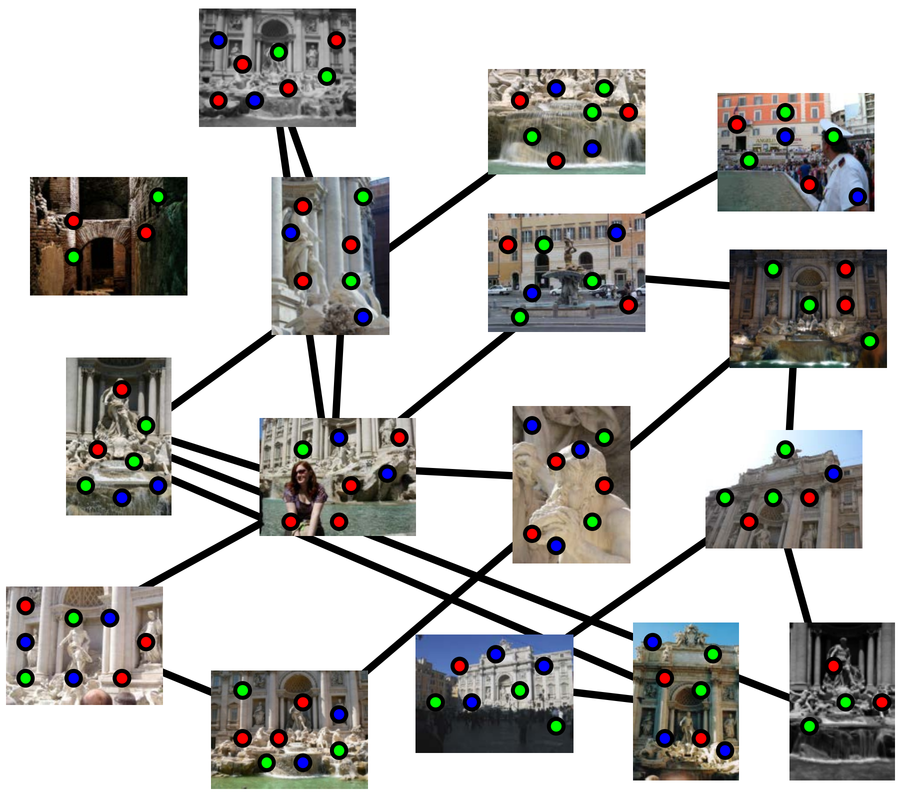


Feature Detection



Feature Matching

Match features between each pair of images



Feature Matching

Refine matching using RANSAC to estimate fundamental matrix between each pair

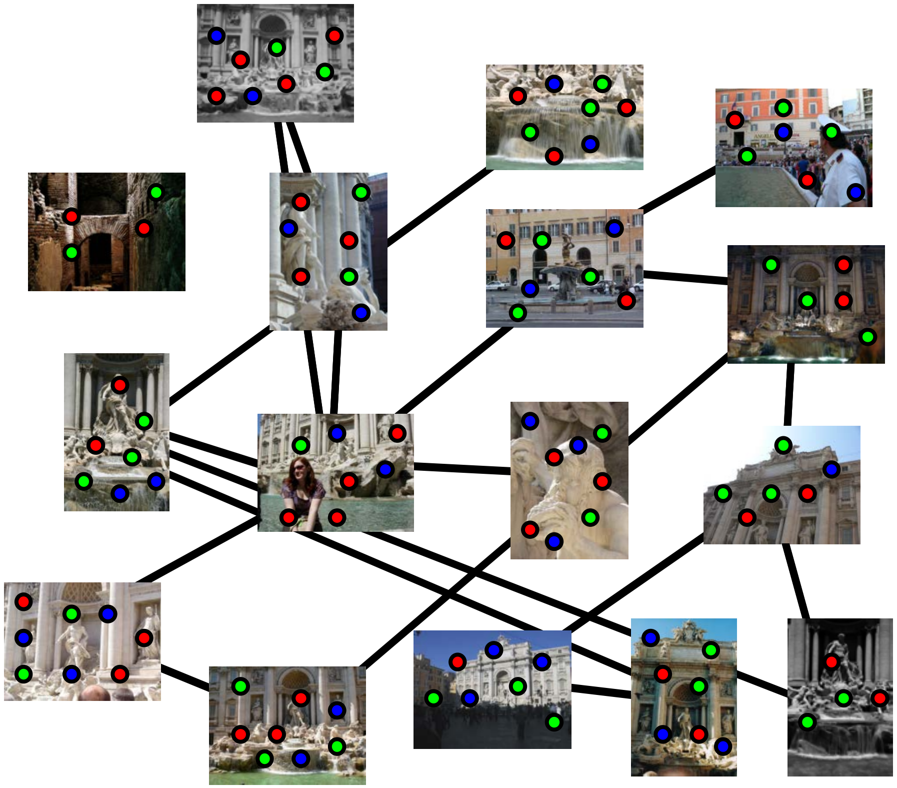
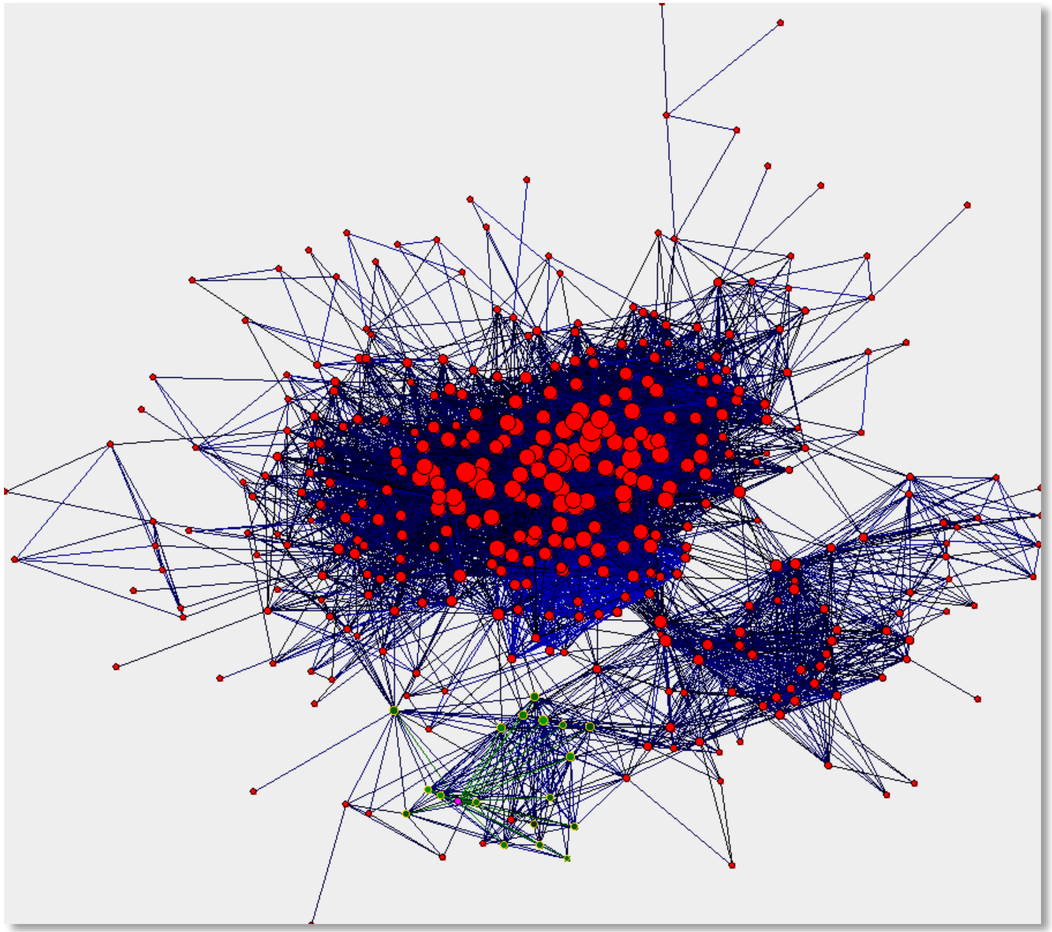
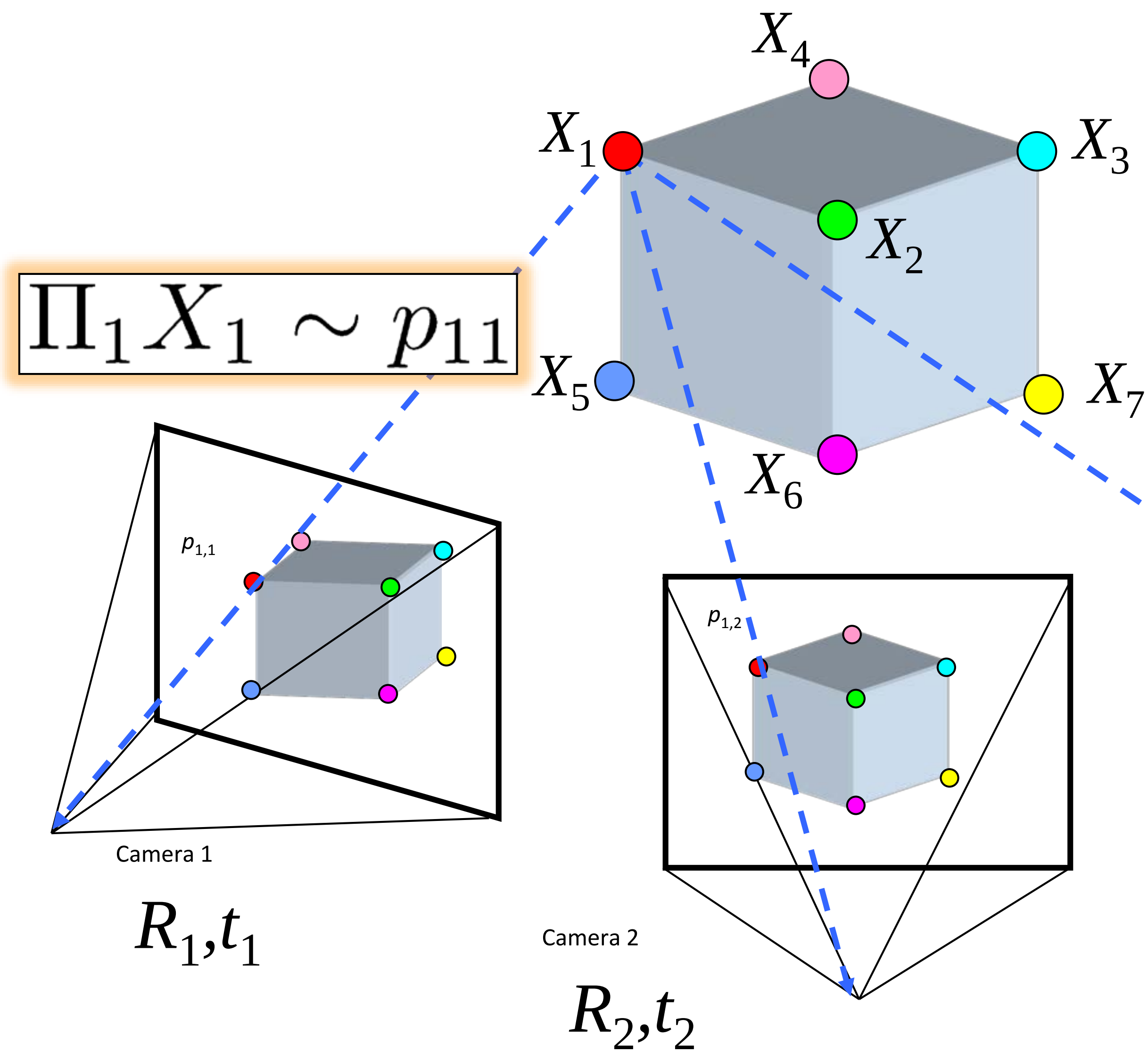


Image Connectivity Graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

Structure from motion



minimize
$$g(\mathbf{R}, \mathbf{T}, \mathbf{X})$$
non-linear least squares

Problem Size

- What are the variables?
- How many variables per camera?
- How many variables per point?

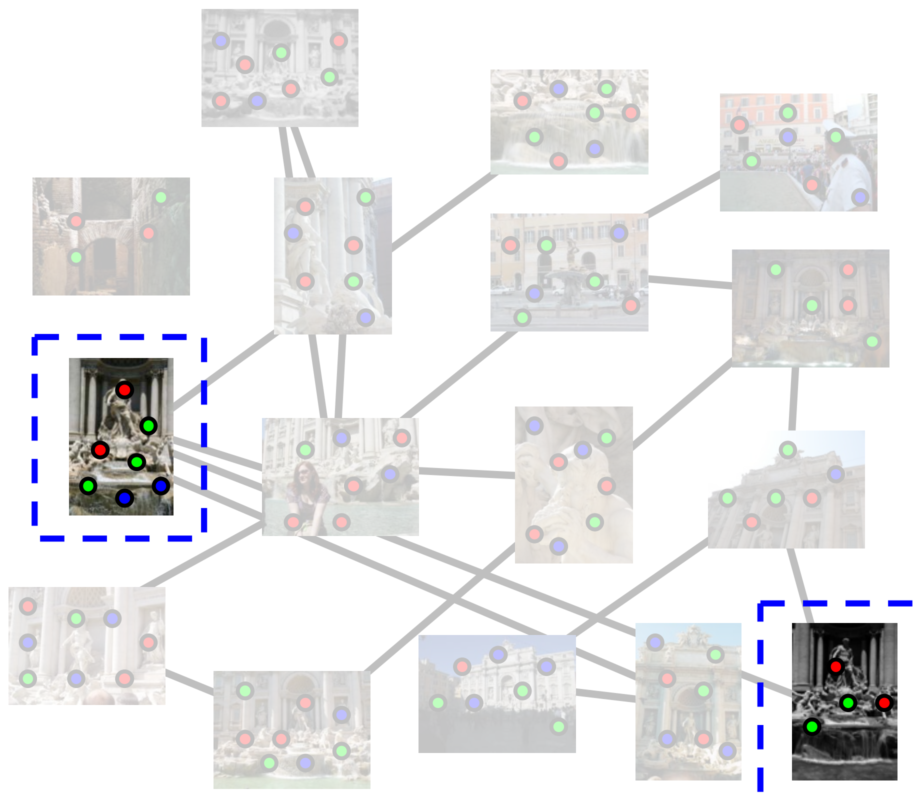
- Trevi Fountain collection

  466 input photos

  + > 100,000 3D points

  = very large optimization problem

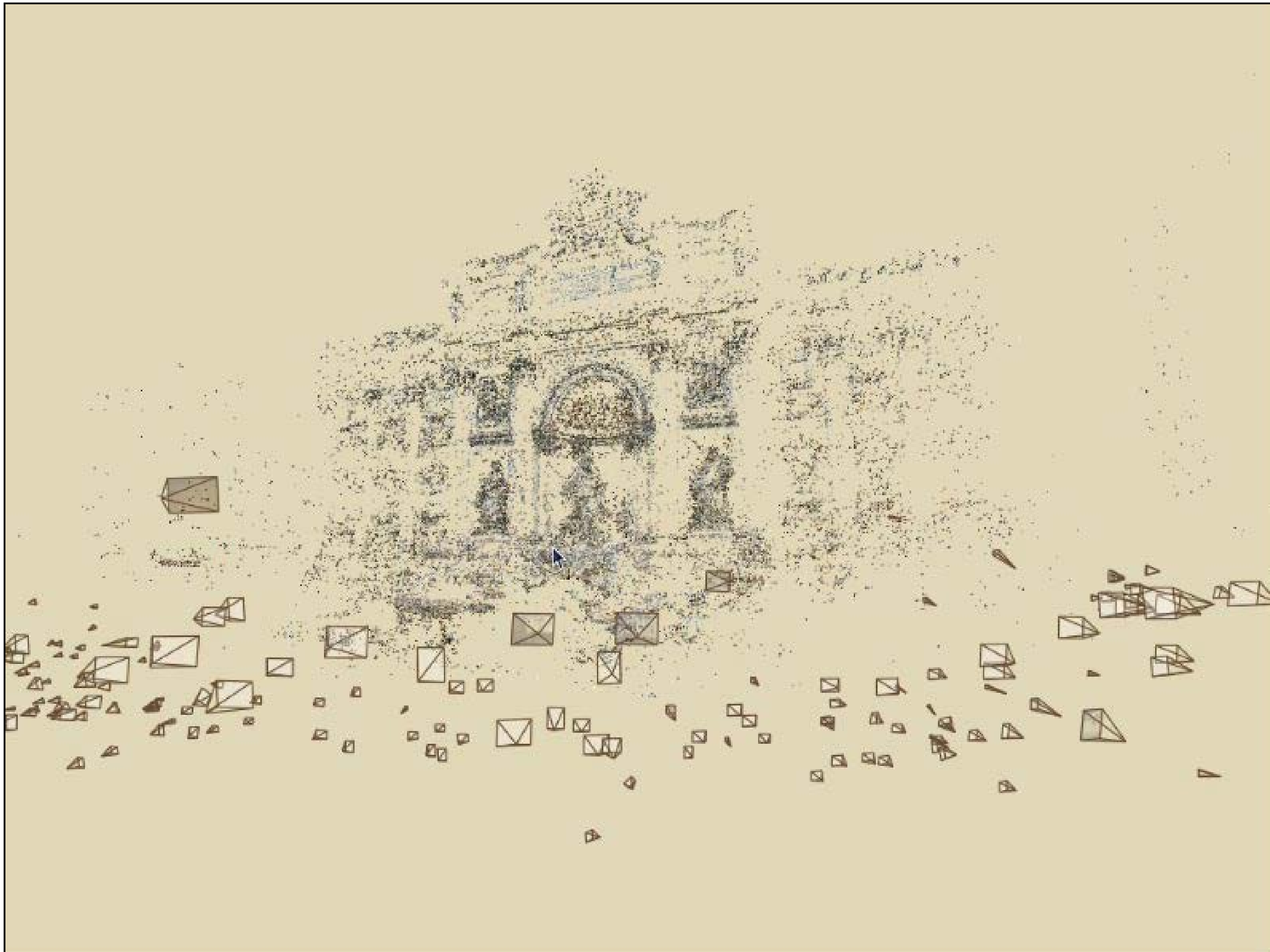
Incremental Structure from Motion



To help get good initializations for all of the parameters of the system, reconstruct the scene incrementally, starting from two photographs and the points they observe

Incremental Structure from Motion

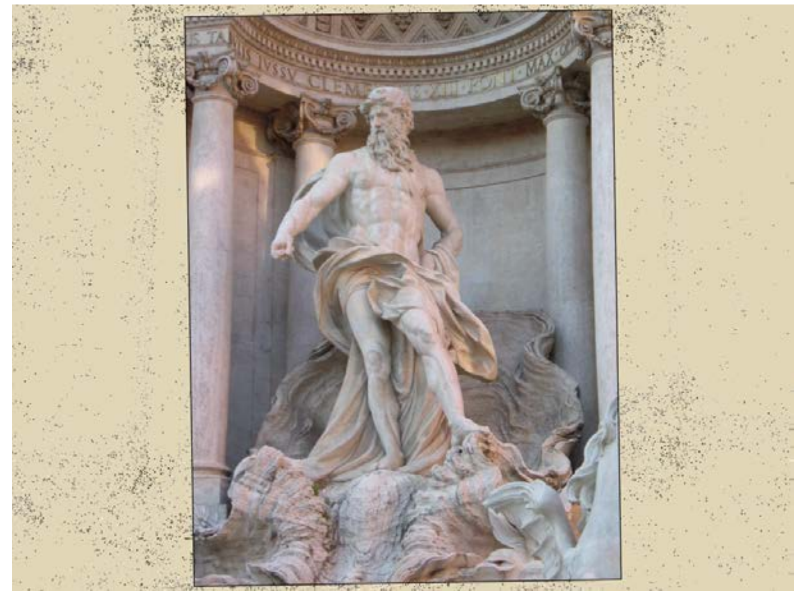
Photo Explorer



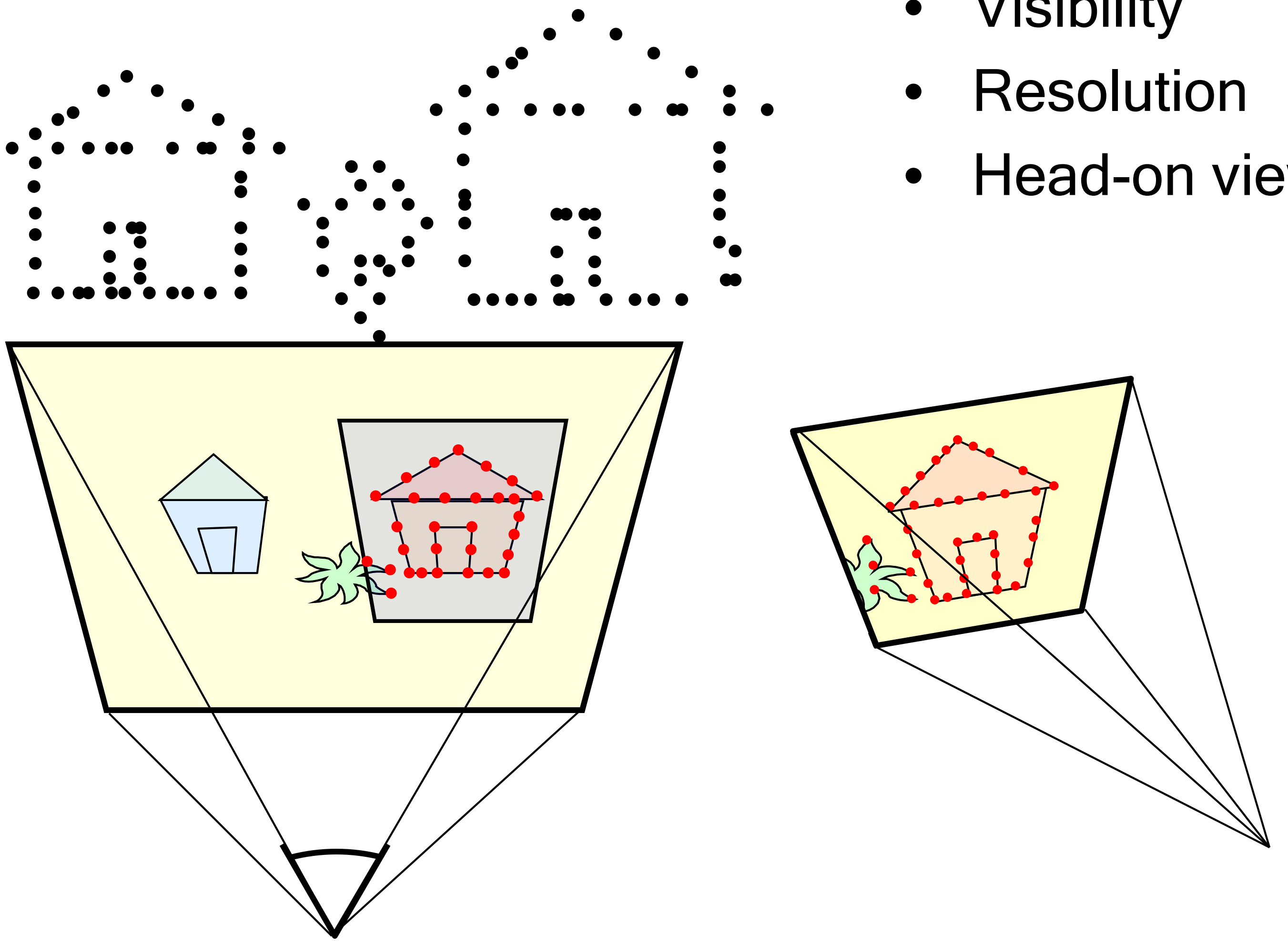
Navigation Controls

- Free-flight navigation
- Object-based browsing
- Relation-based browsing
- Overhead map

Object-based Browsing

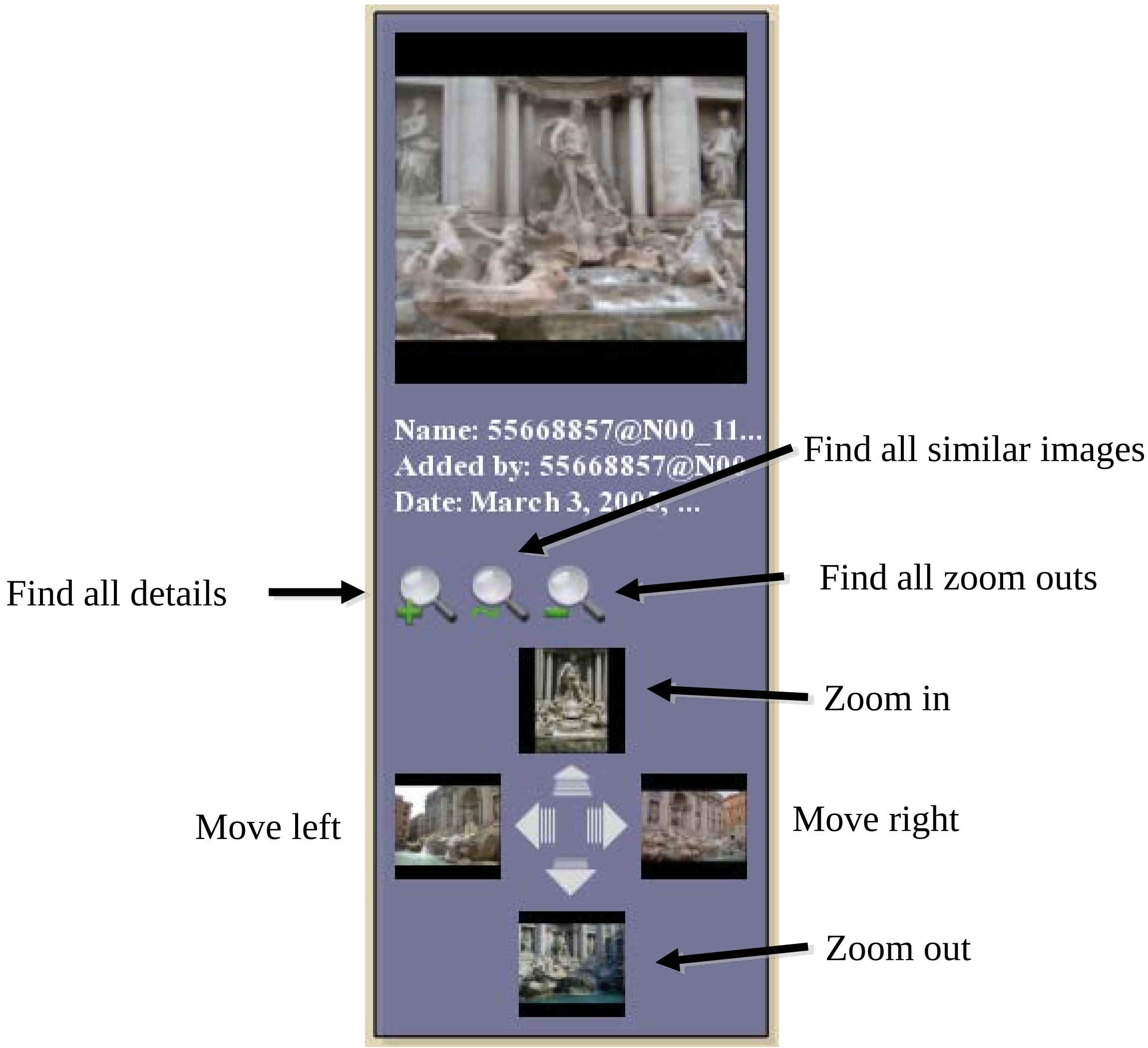


Object-based browsing

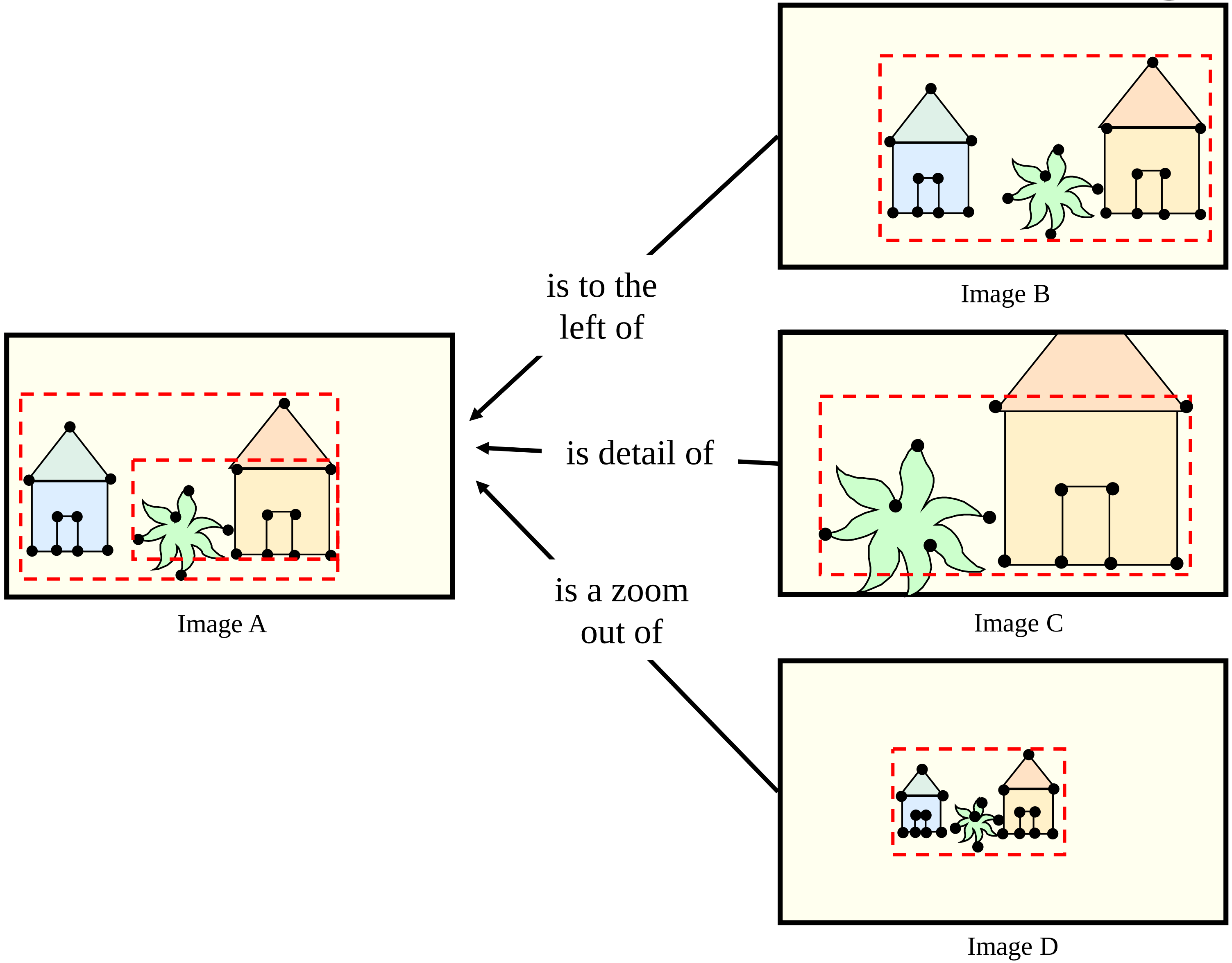


- Visibility
- Resolution
- Head-on view

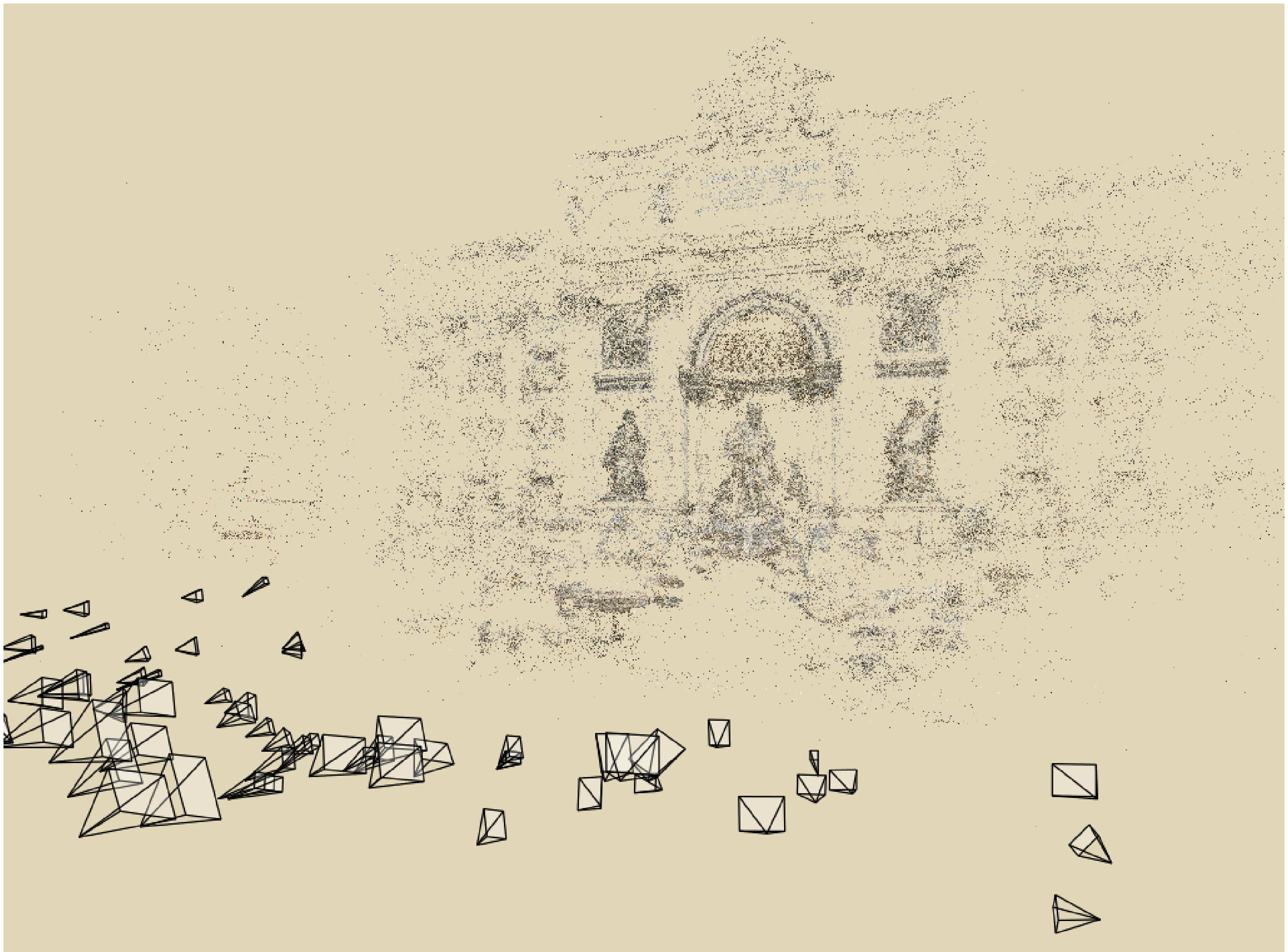
Relation-based Browsing



Relation-based browsing



Rendering

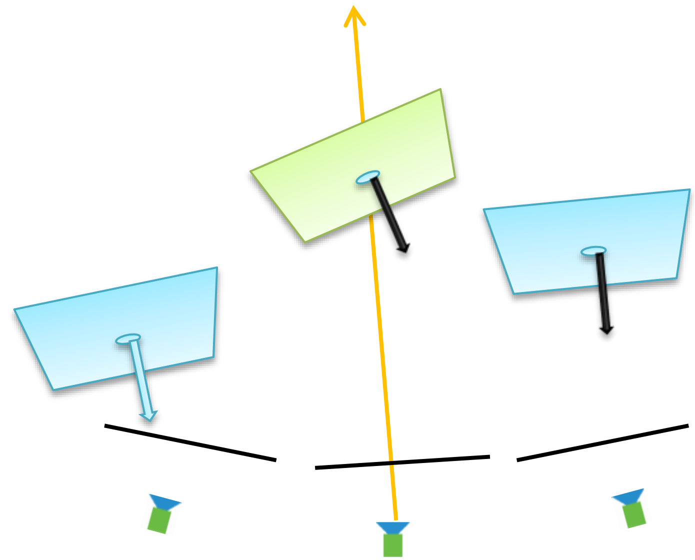


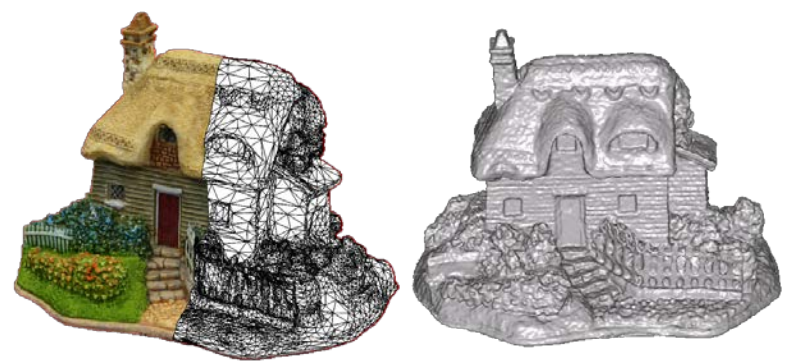
Rendering

Multi-View Stereo

Representations (coming soon)

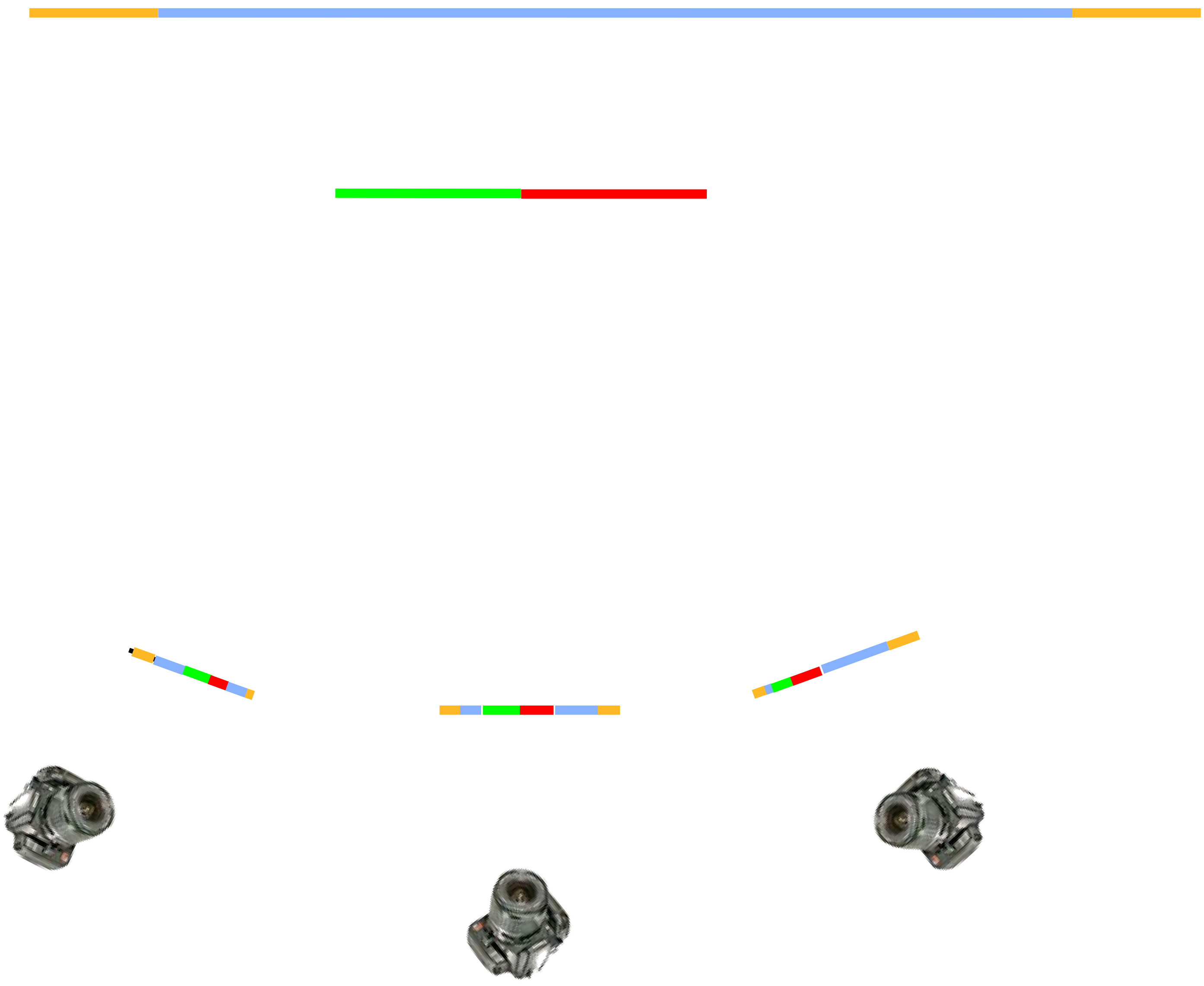
- Depth maps
- Point clouds
- Surface patches
- Level sets
- Voxels grids
- Meshes
- …

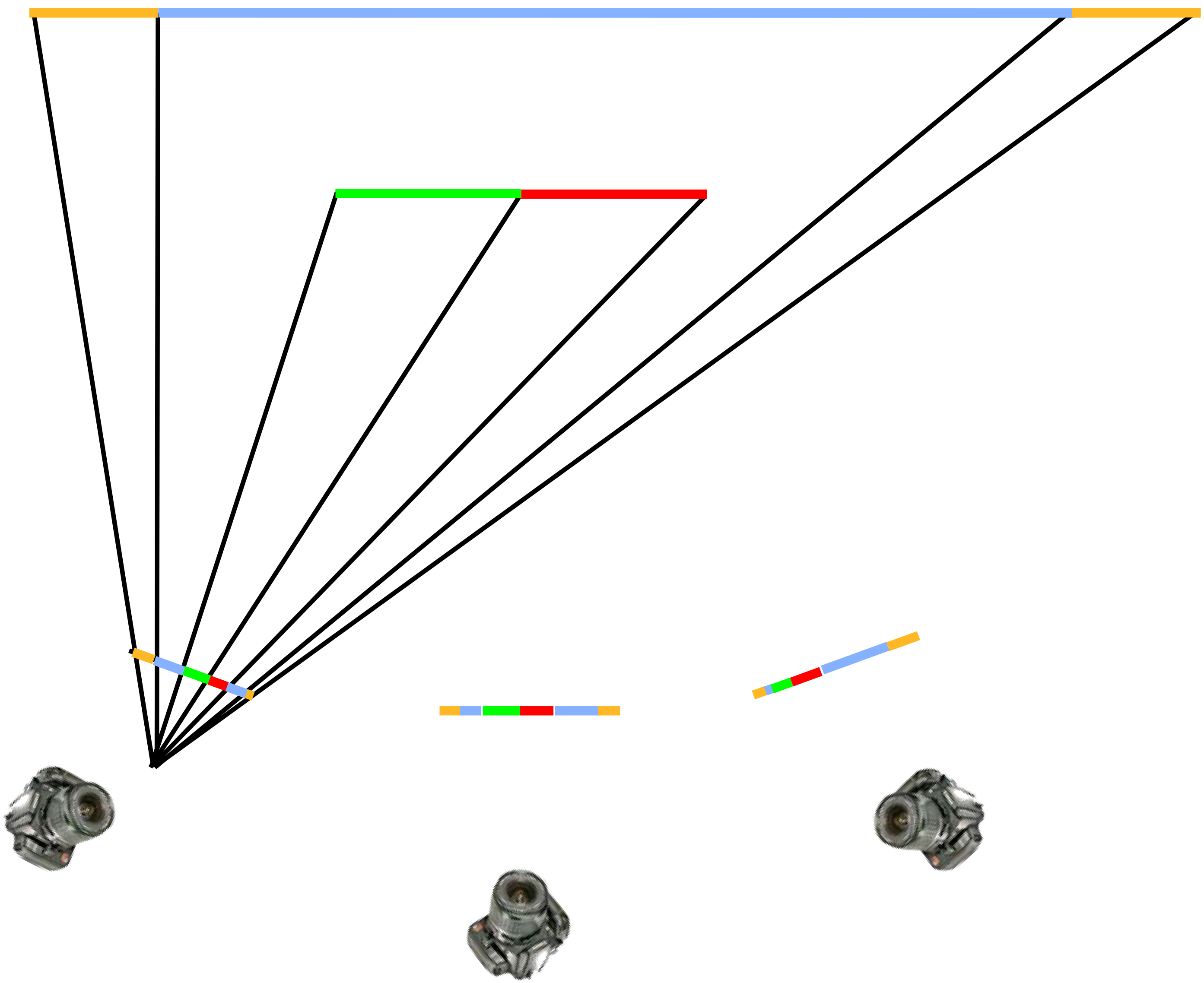


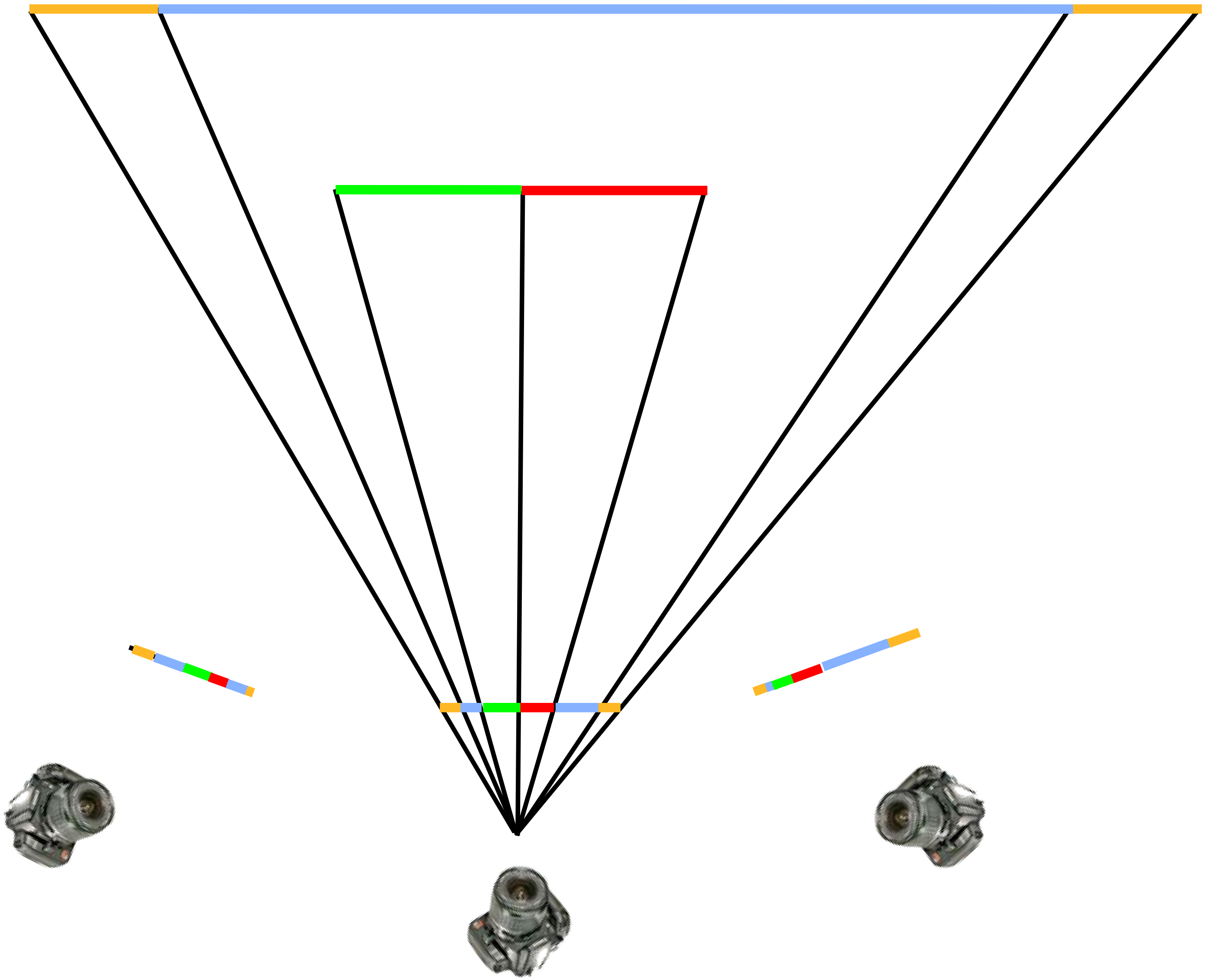


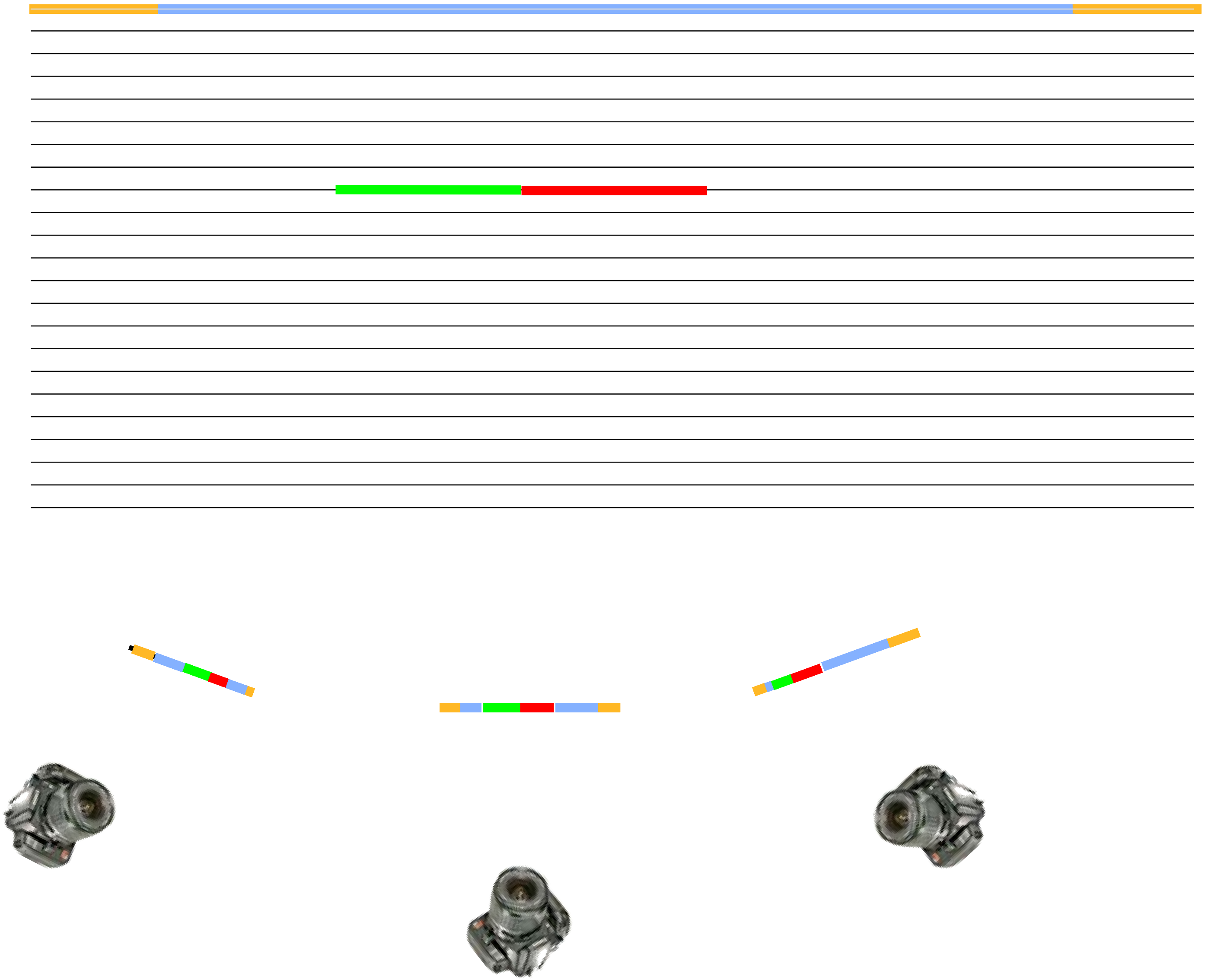
Plane Sweeping Stereo

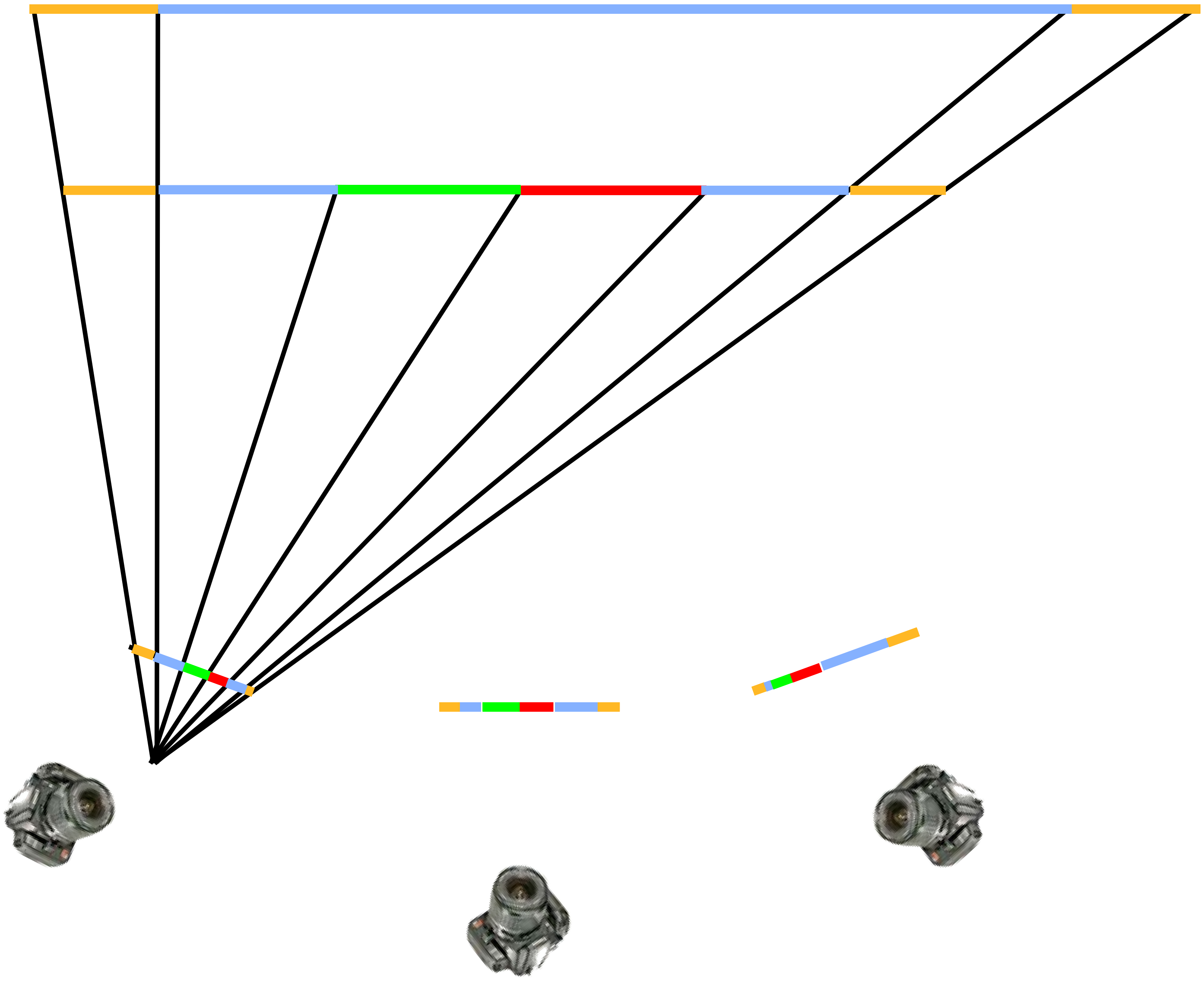
- True multi-view stereo
- Define family of planes that sweep depth range of interest
- For each pixel, generate and evaluate depth hypotheses by intersecting its ray with planes
  - Compute cost/score function to measure "photo-consistency"

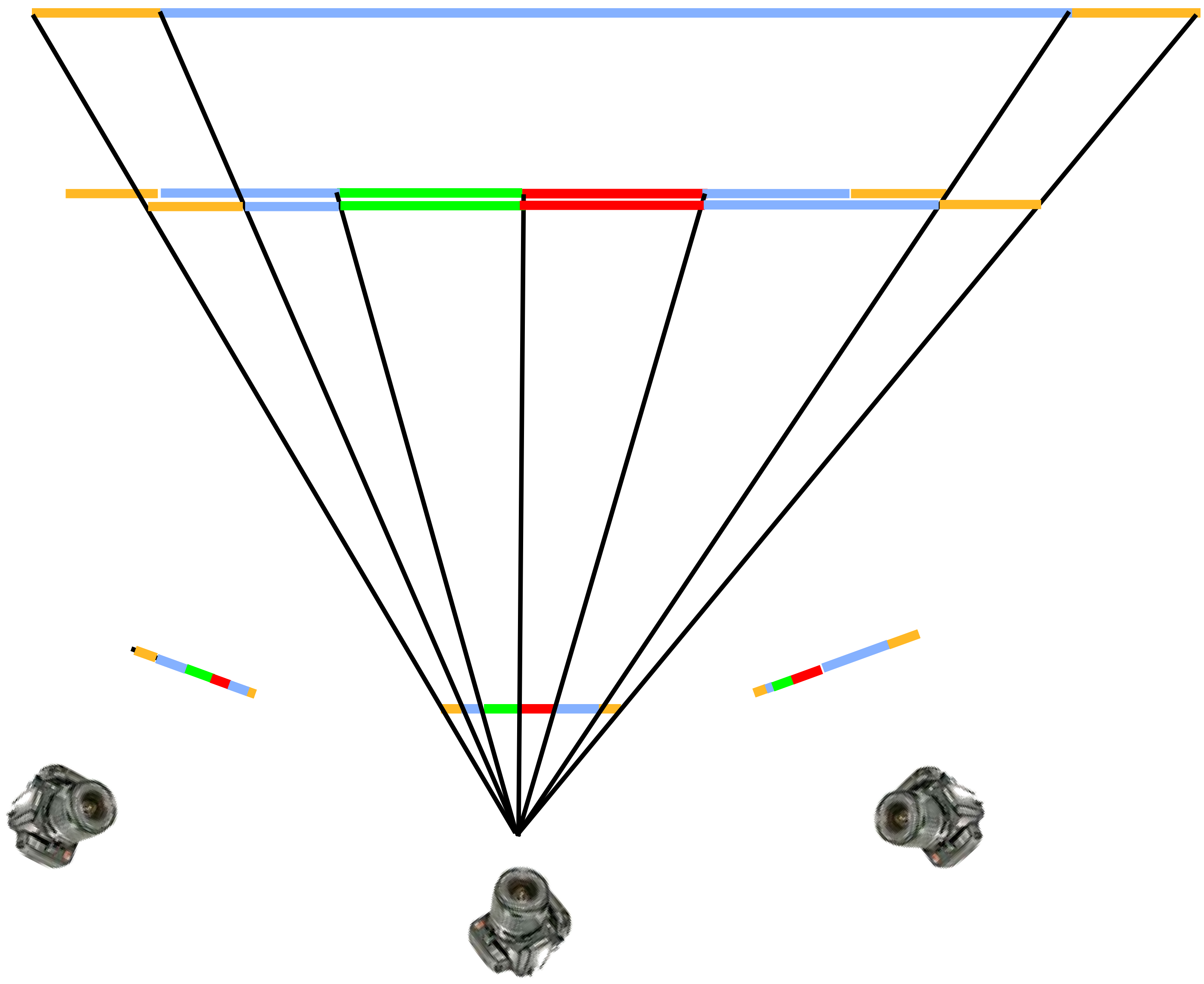


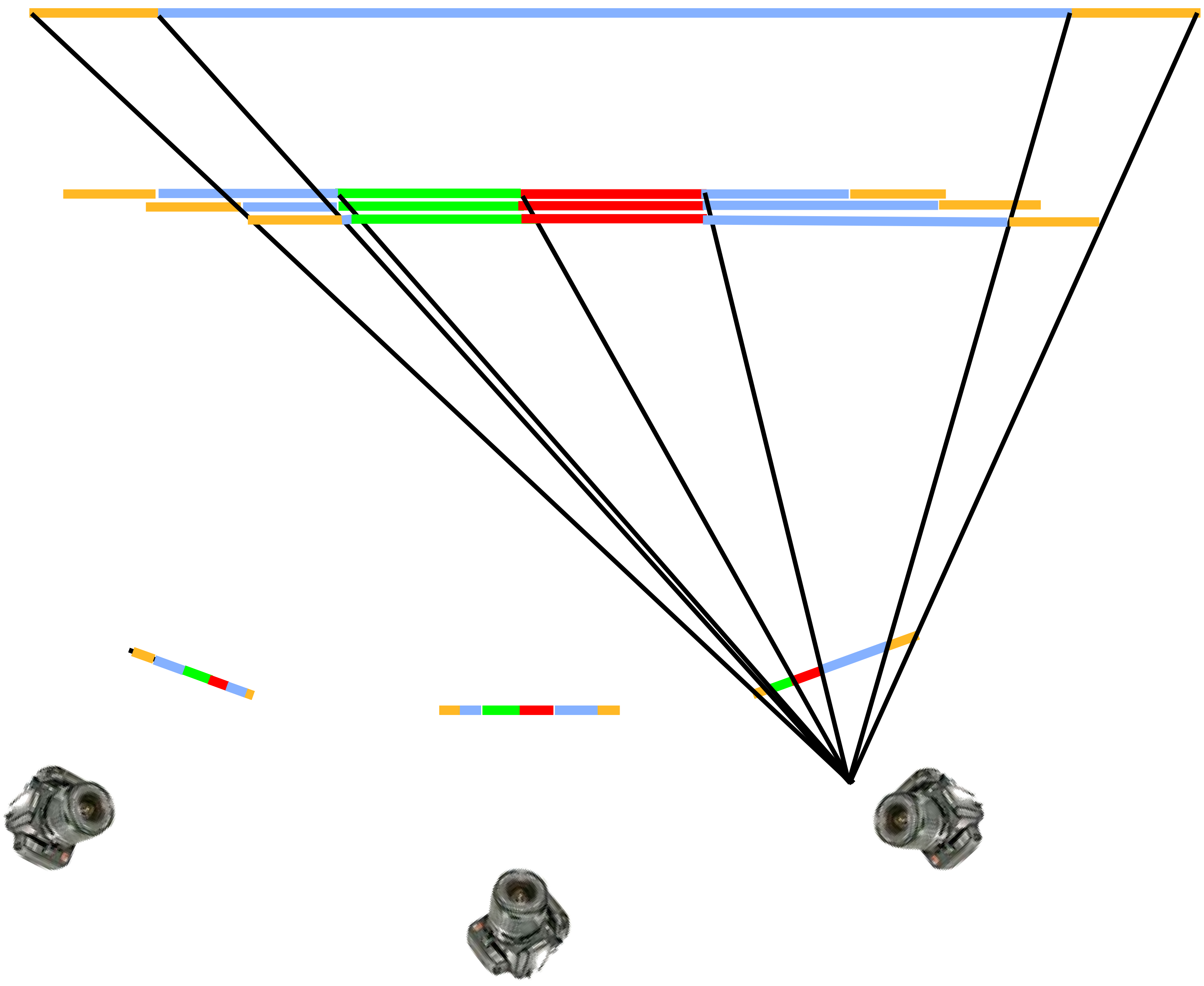


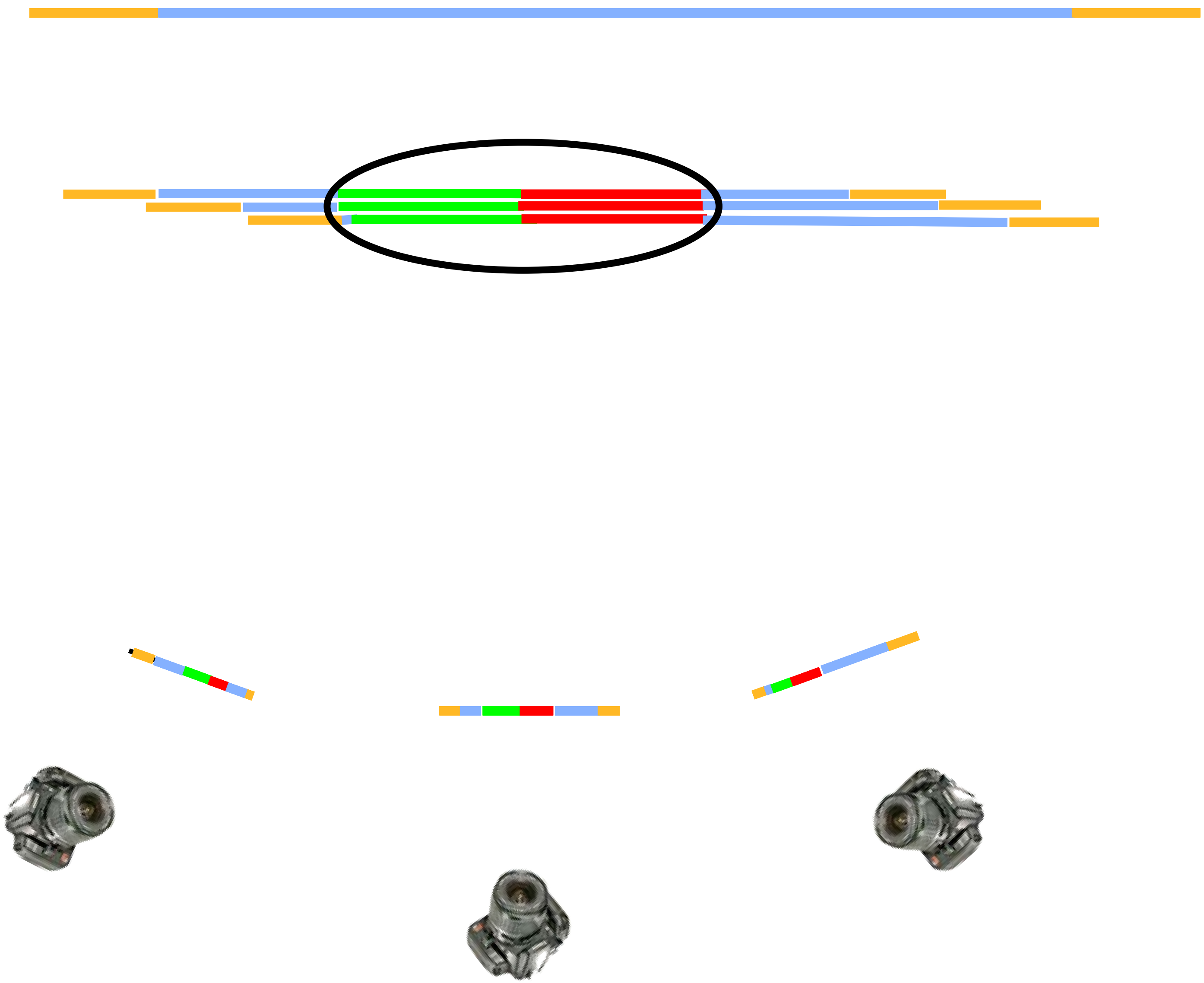


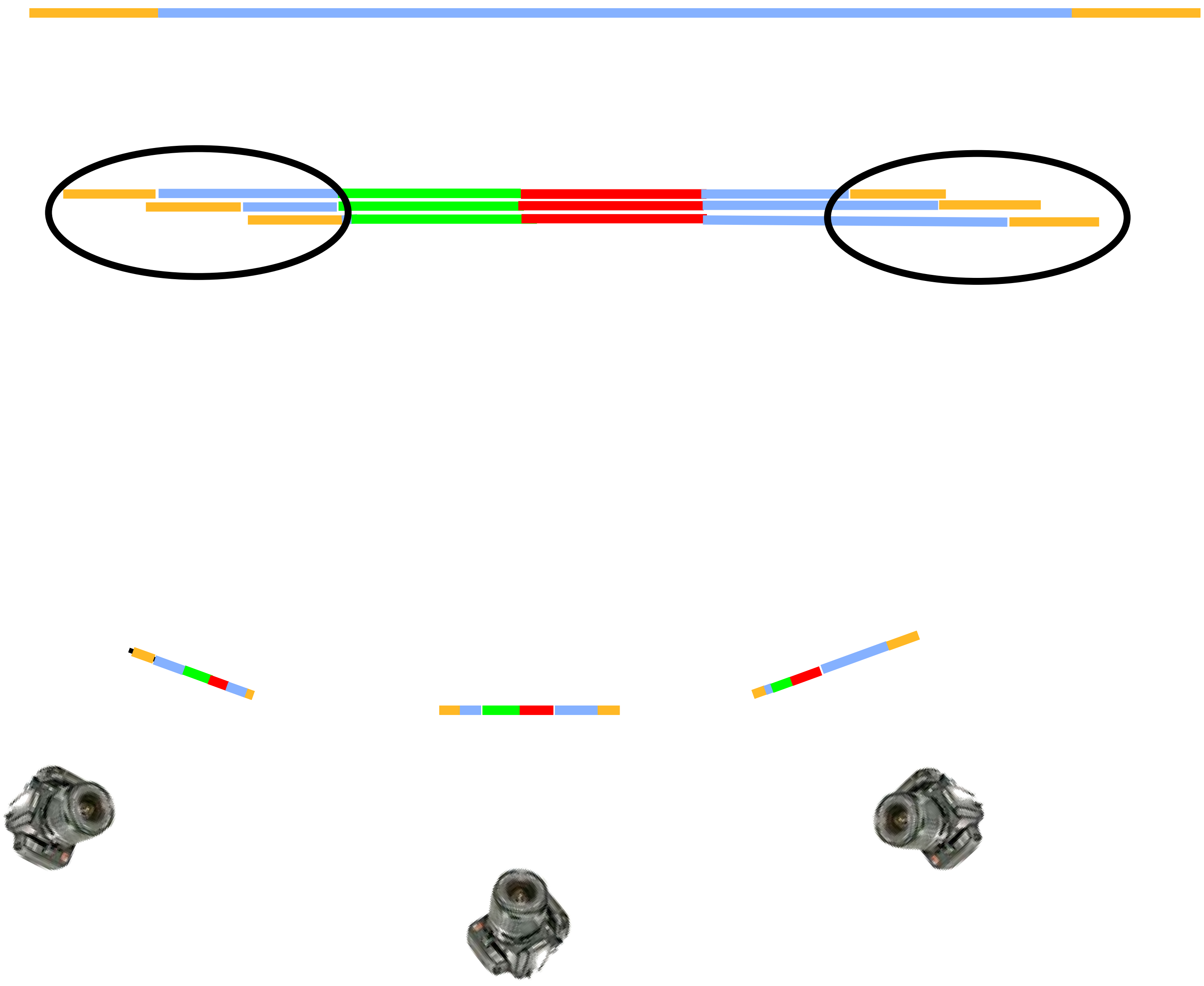


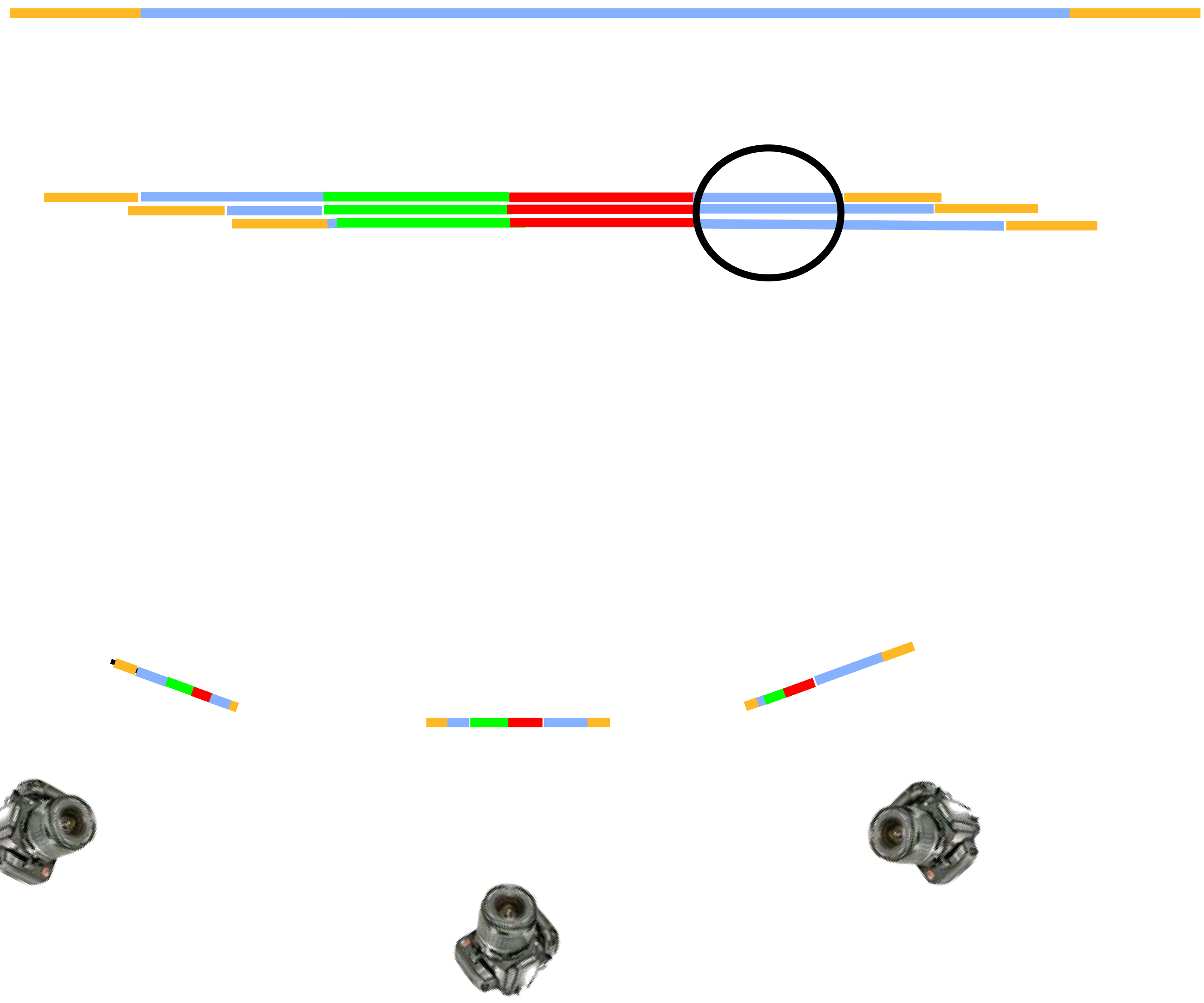






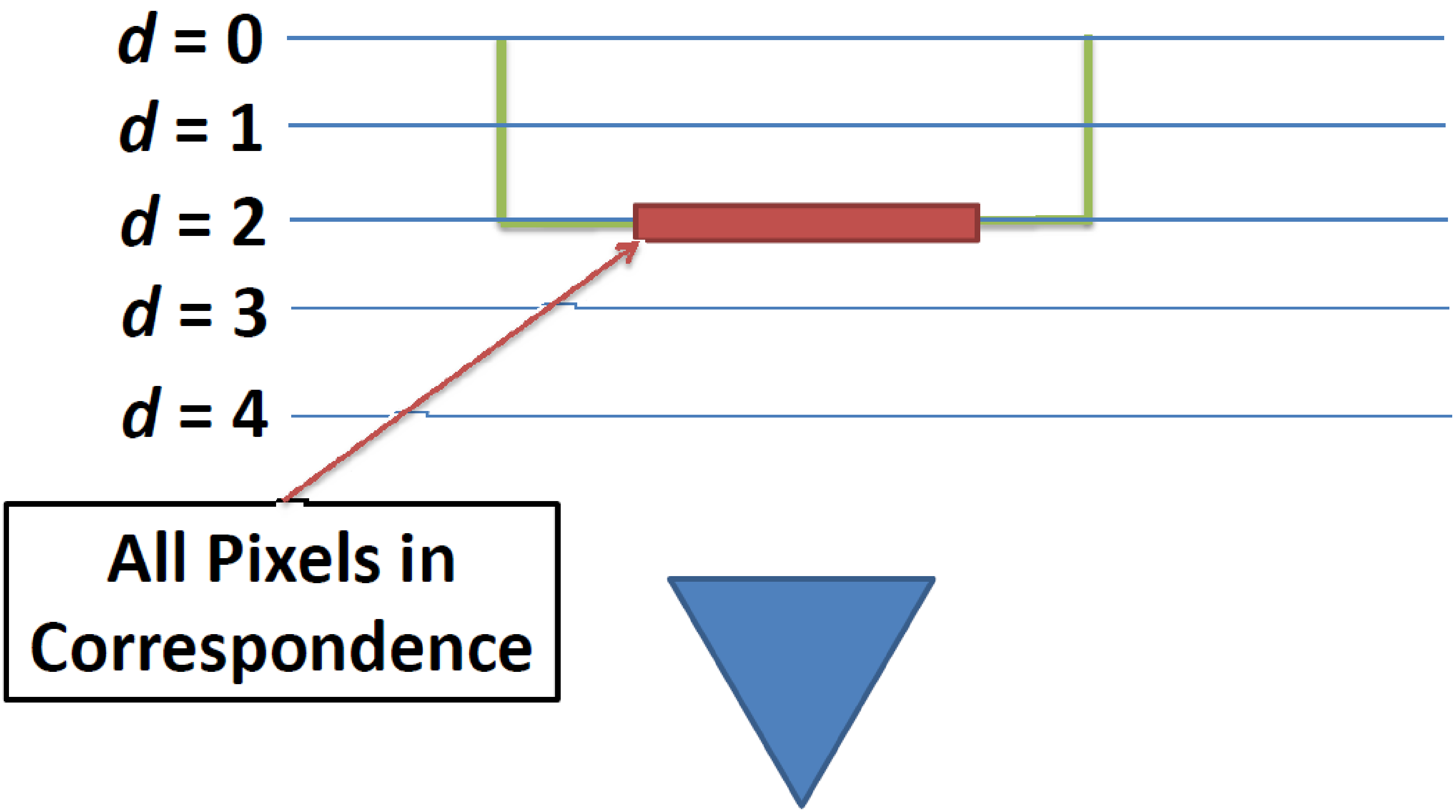






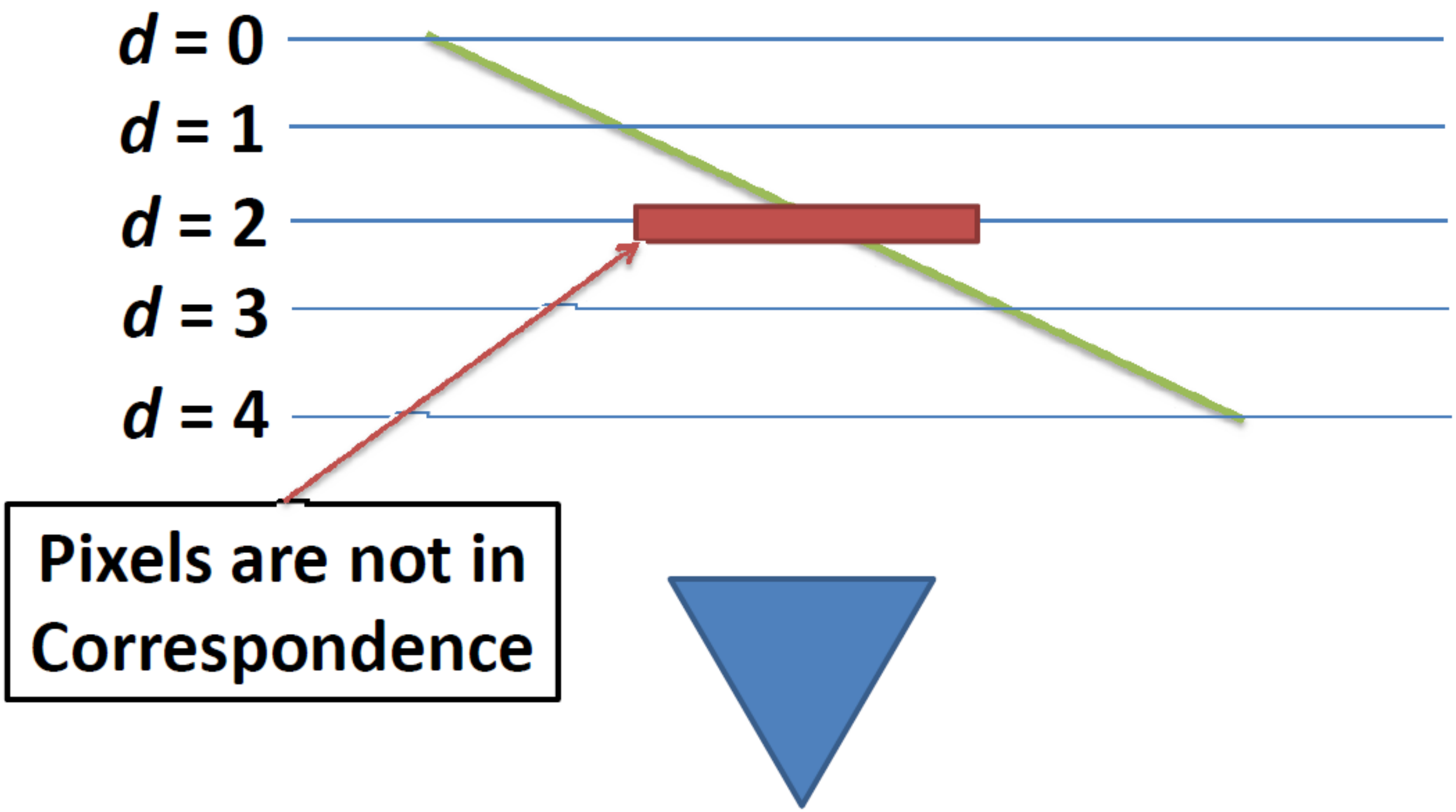
Limitation of Plane Sweeping

- Assumes that all surfaces are planes which are fronto-parallel to the camera



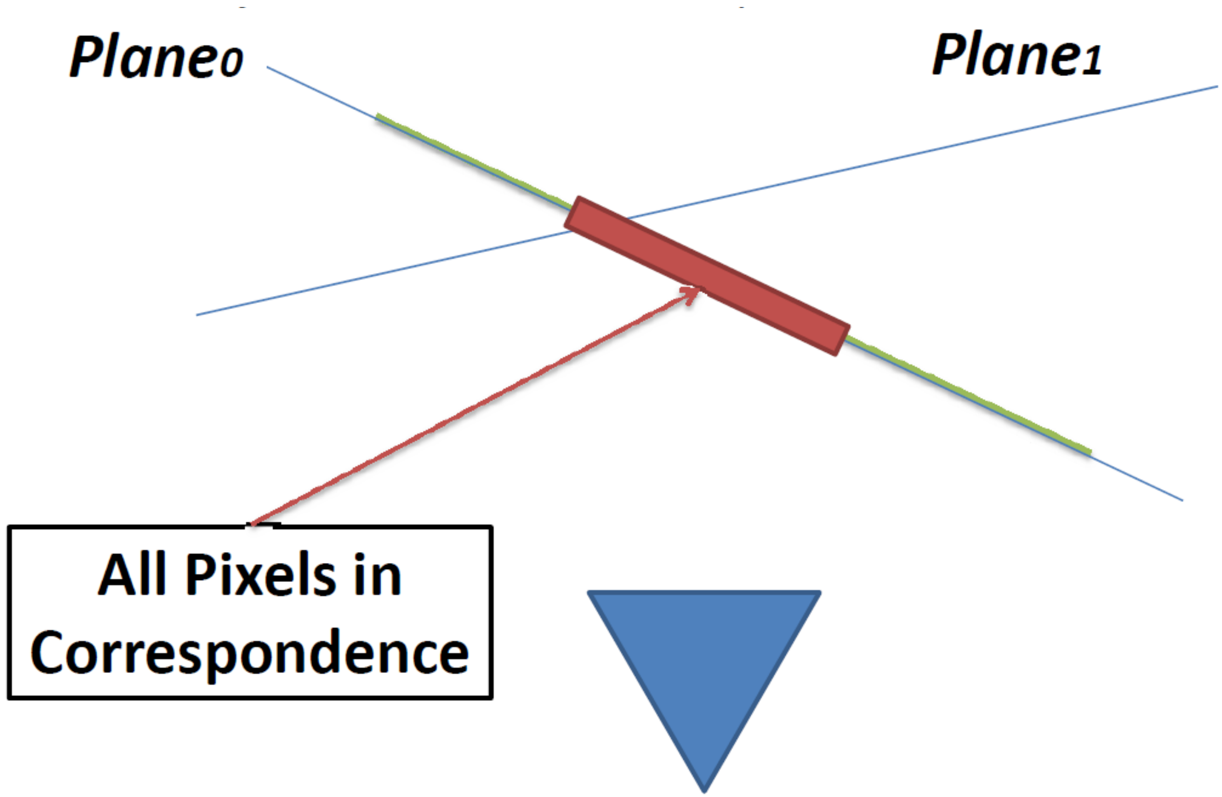
Limitation of Plane Sweeping

- Cannot handle slanted planes



Multi-way Plane Sweeping

- Solutions:
  - Detect planes in the scene and sweep parallel to them
  - Or, sweep in a few directions which are often sufficient



Depth Map Fusion

- Fuse depth maps to improve accuracy and minimize violations of visibility constraints

Depth Map Fusion

- Depth maps from plane sweeping are noisy
  – No occlusion handling
  – No consistency across views

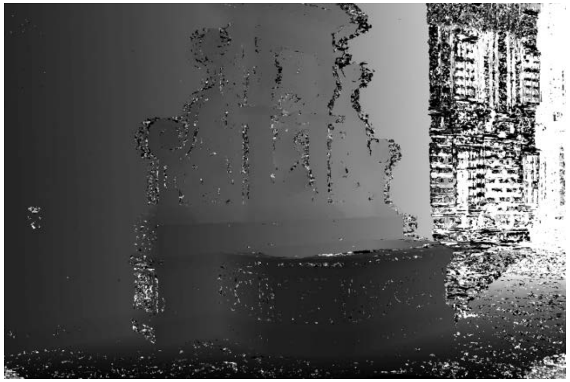
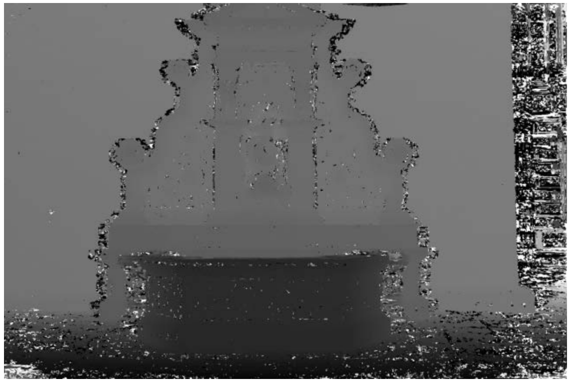
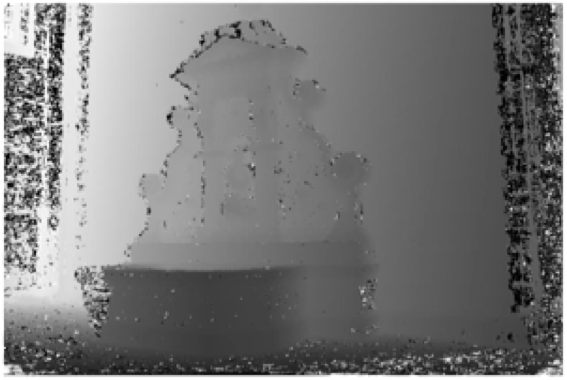
- Use multiple depth estimates to:
  – Correct errors
  – Reduce redundancy in the model
- Minimize *occlusions* and *free-space violations*
  – Occluded depth estimates are too far from the camera
  – Depth estimates violating free space of other surfaces are too close

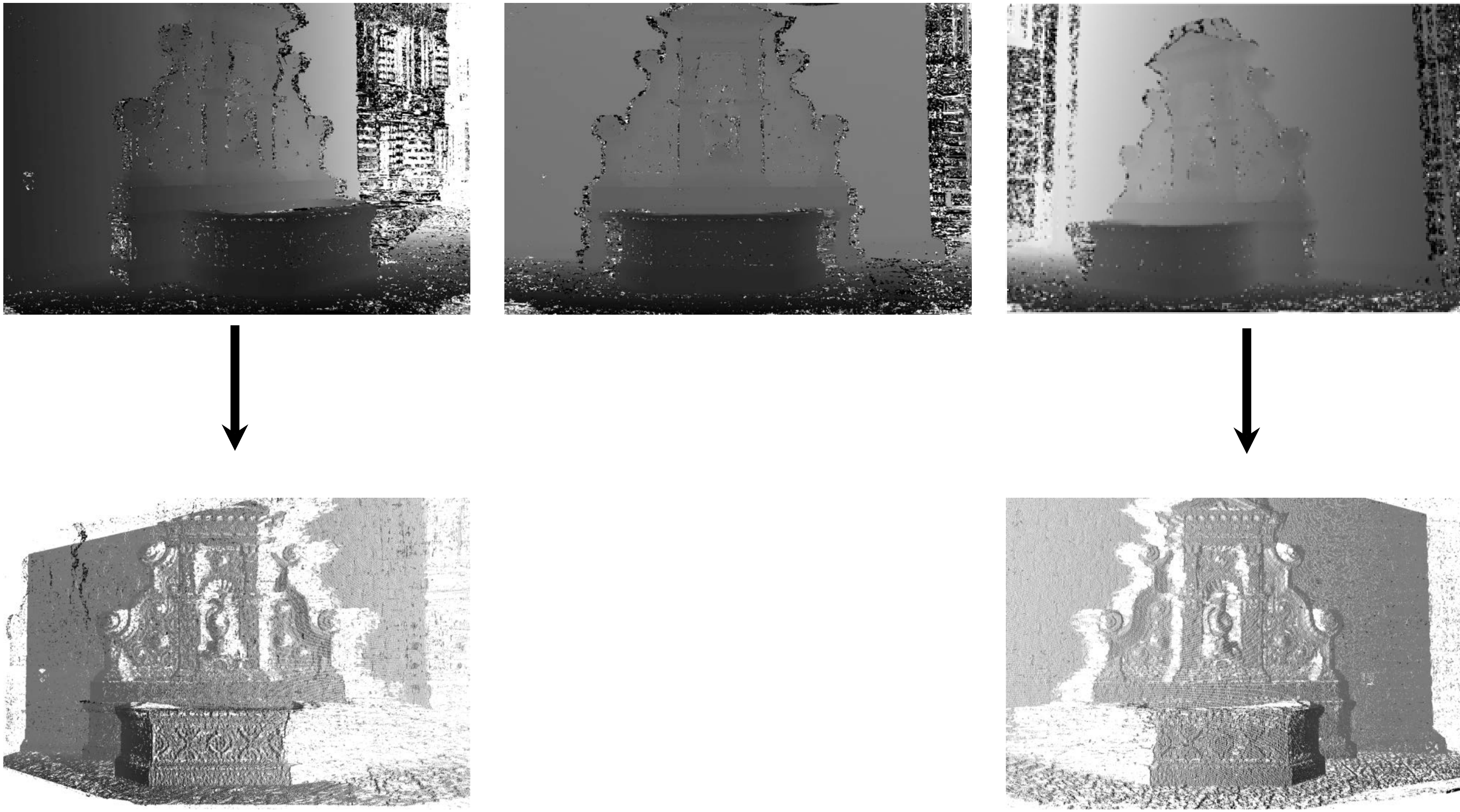
Inputs



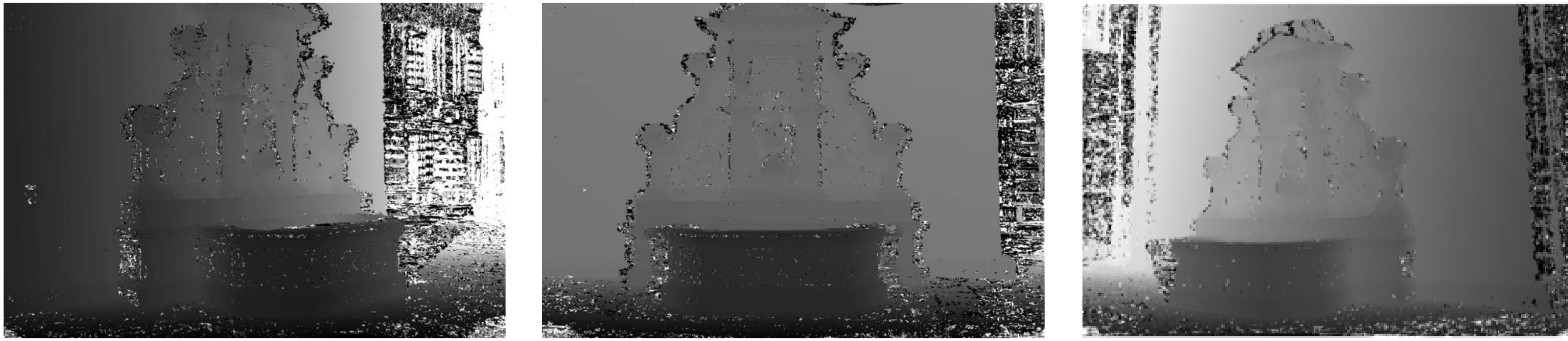
- Depth maps from plane sweeping
- Confidence maps (optionally)



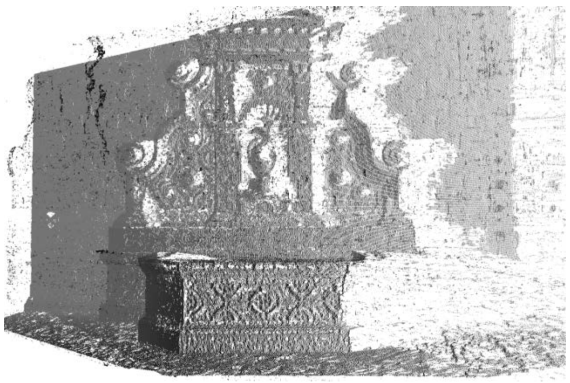







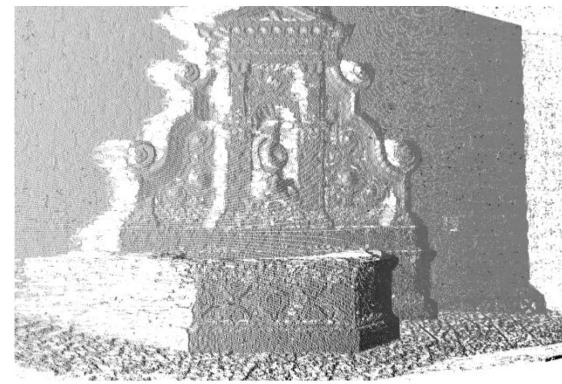
Render depth map of target views to reference view and fuse them according to visibility constraints and confidence

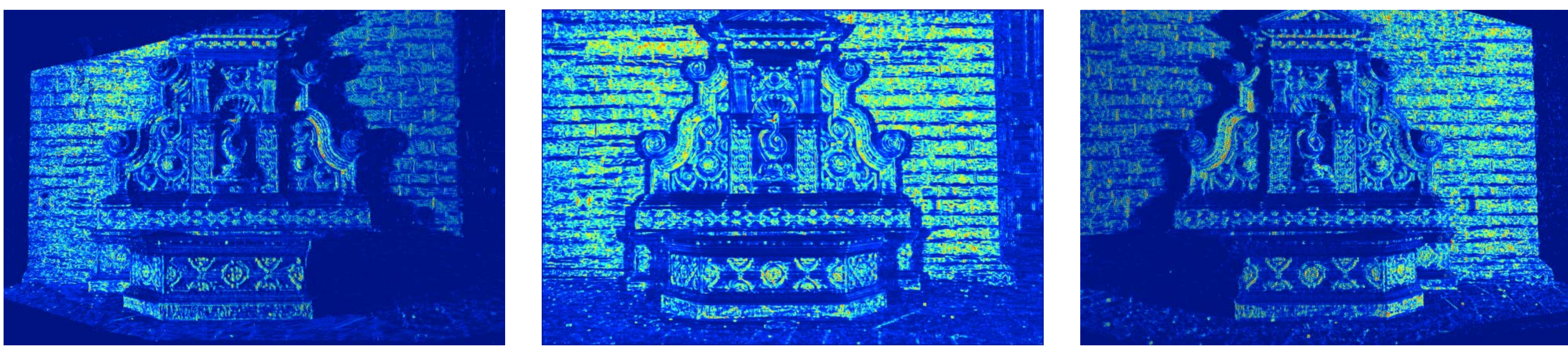


Input depth maps



Depth maps rendered
on reference view

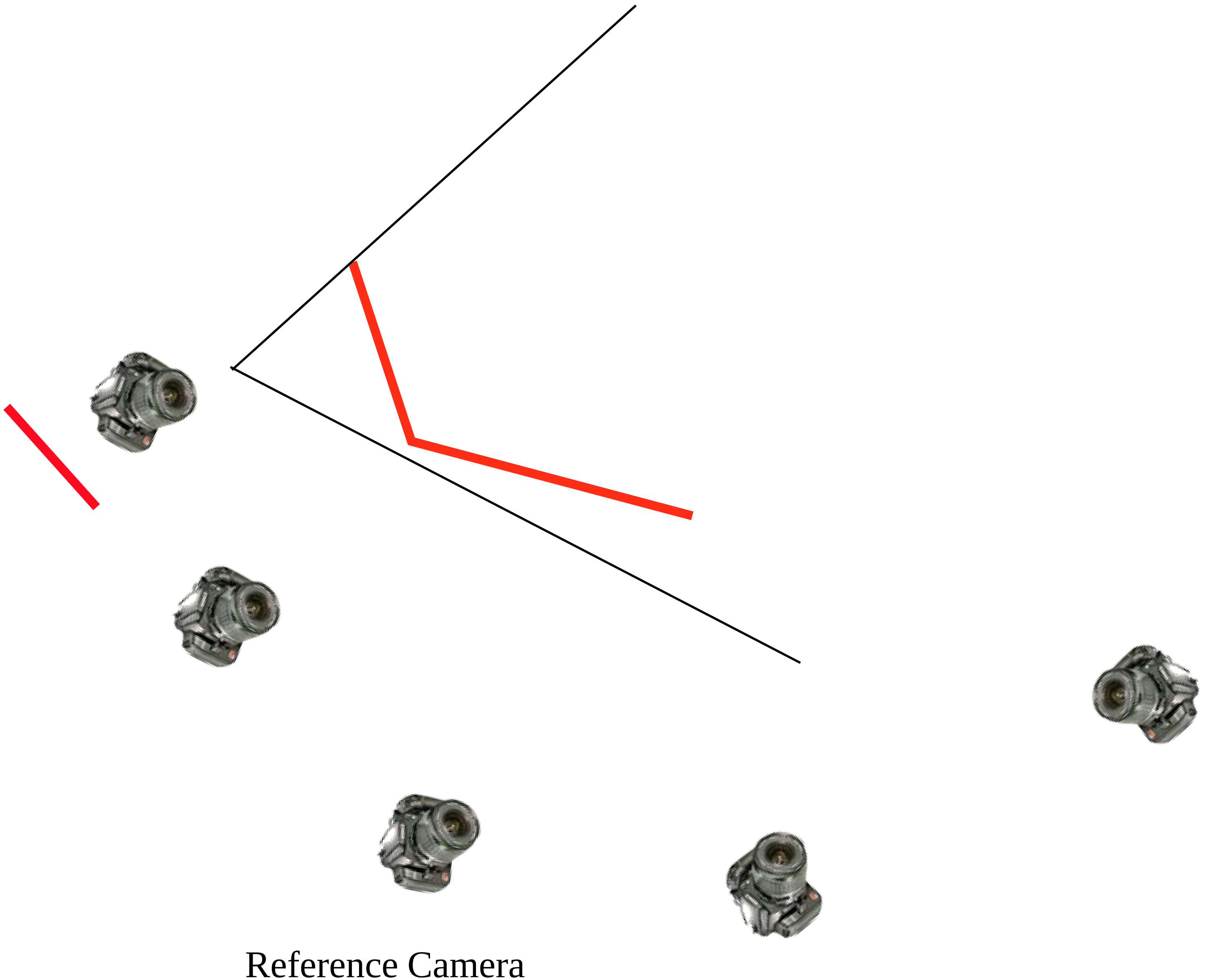


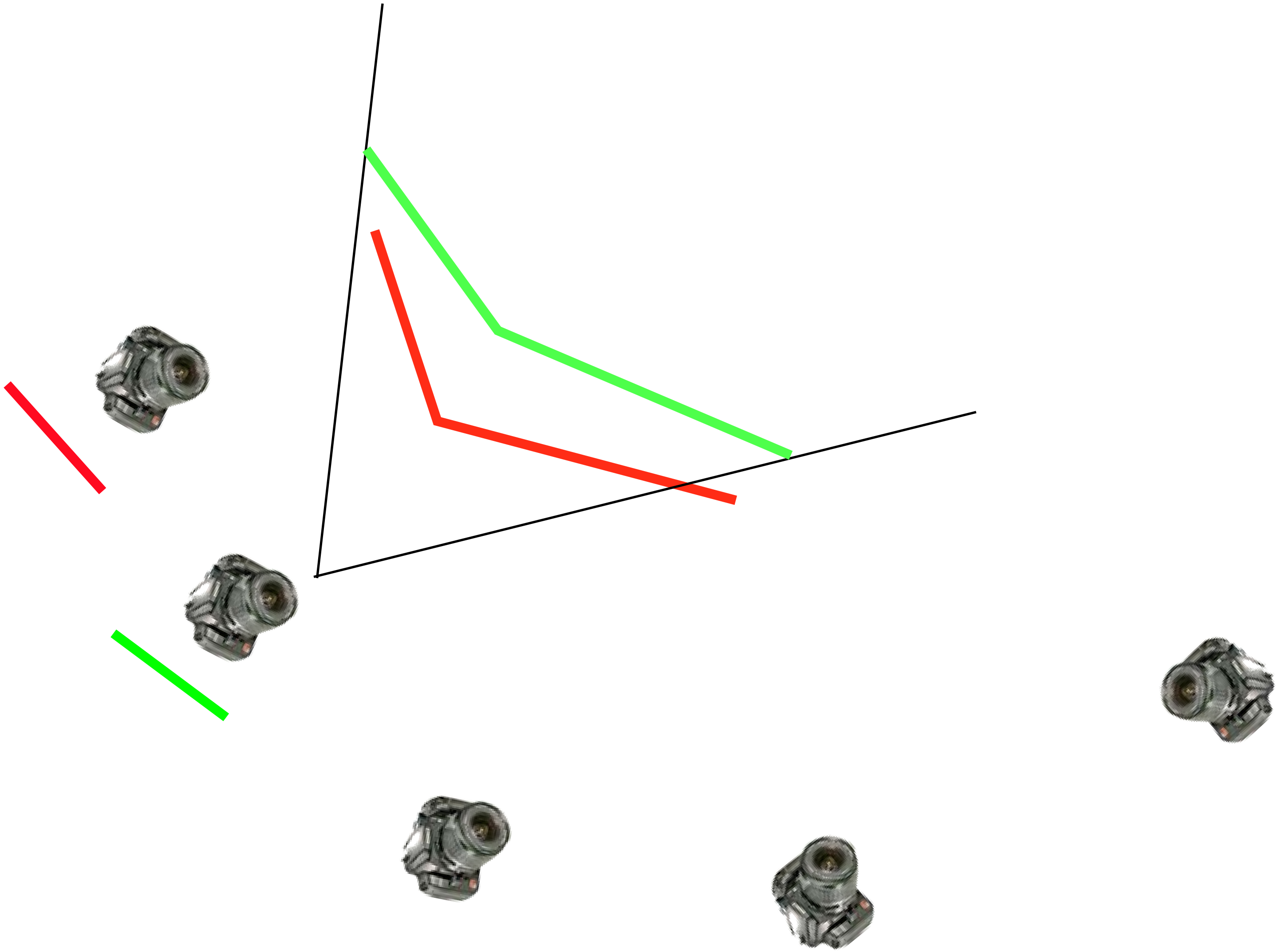


Confidence maps rendered on reference view

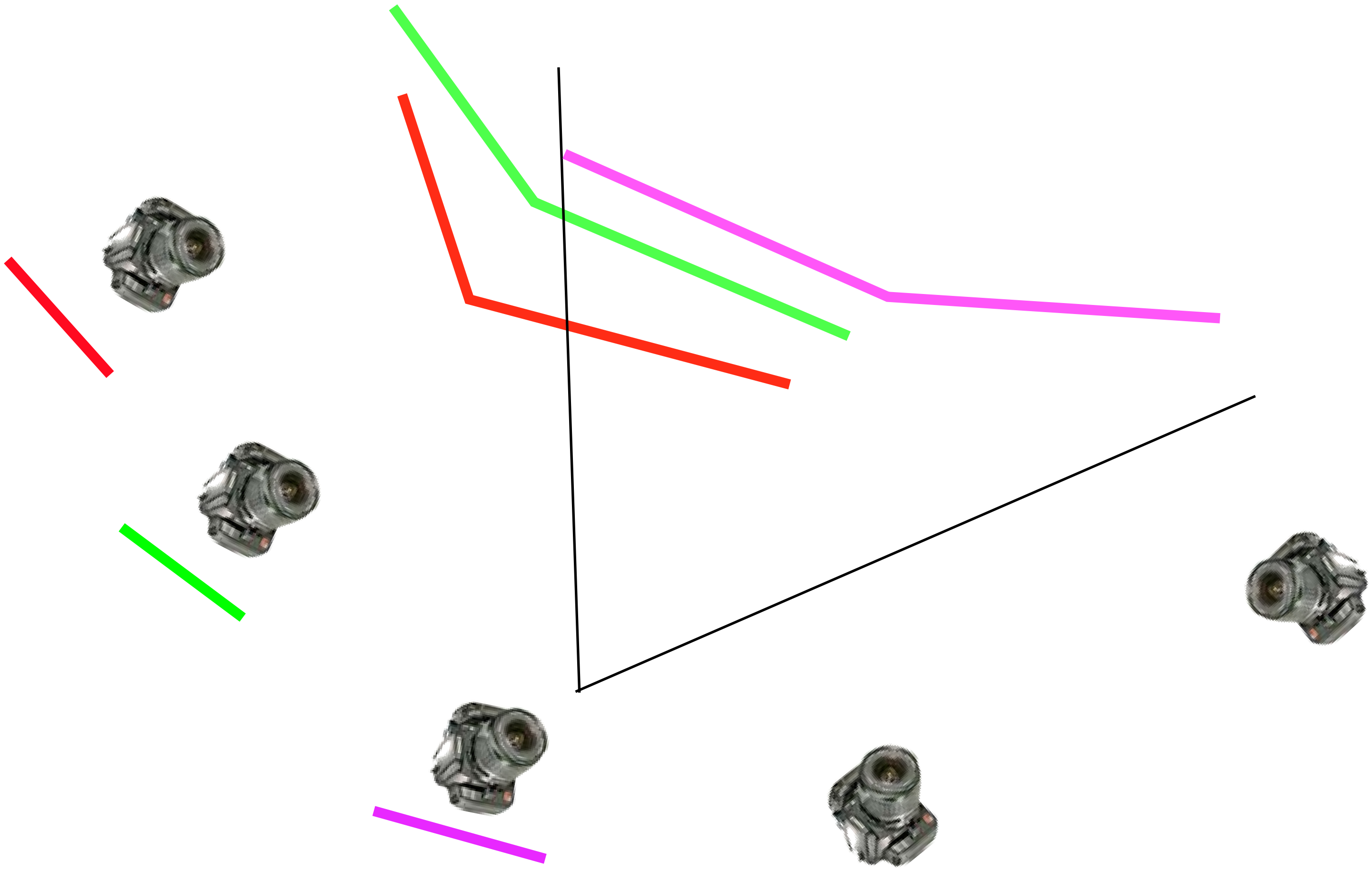
Render Depth Maps to Reference View

Reference Camera

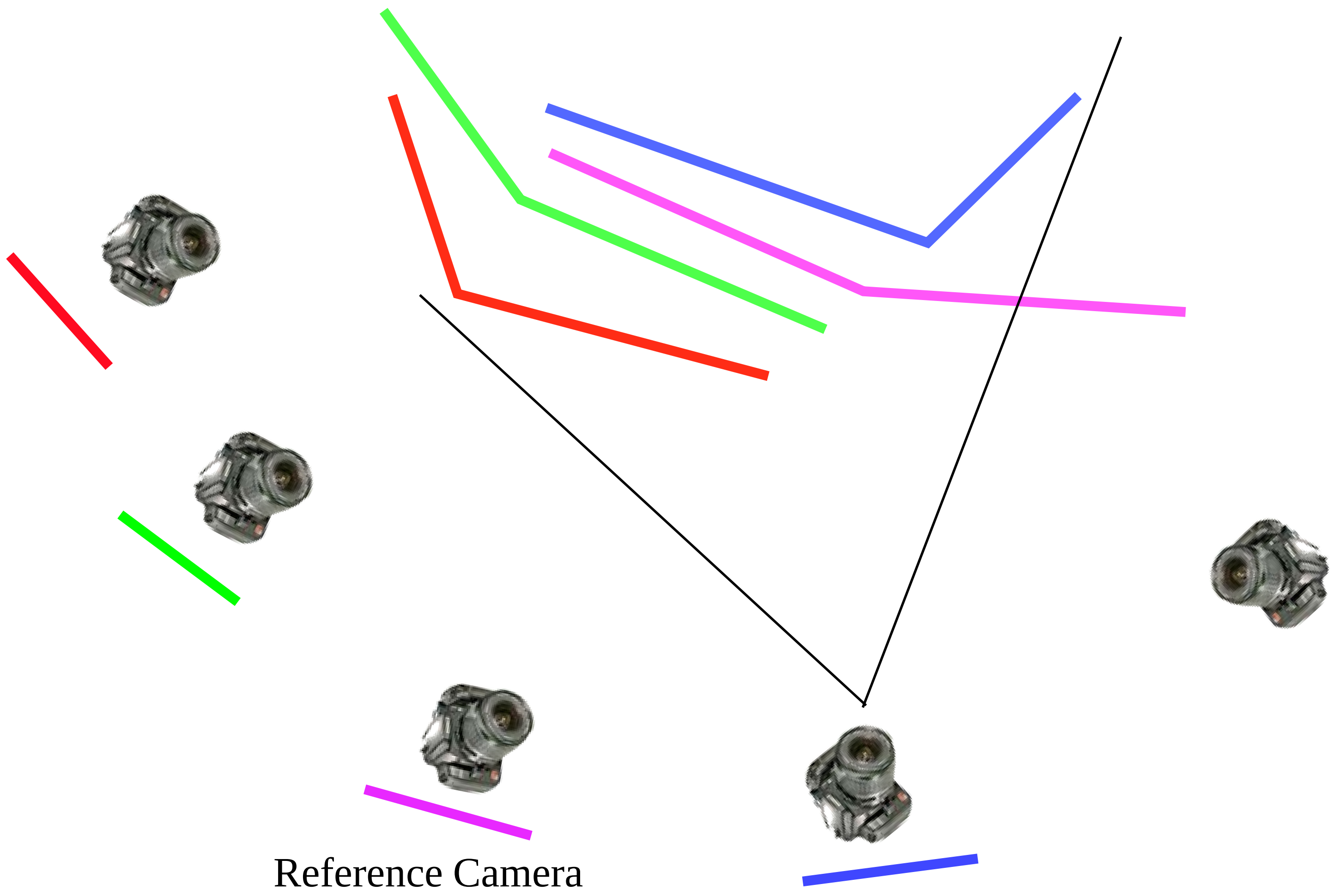

Reference Camera

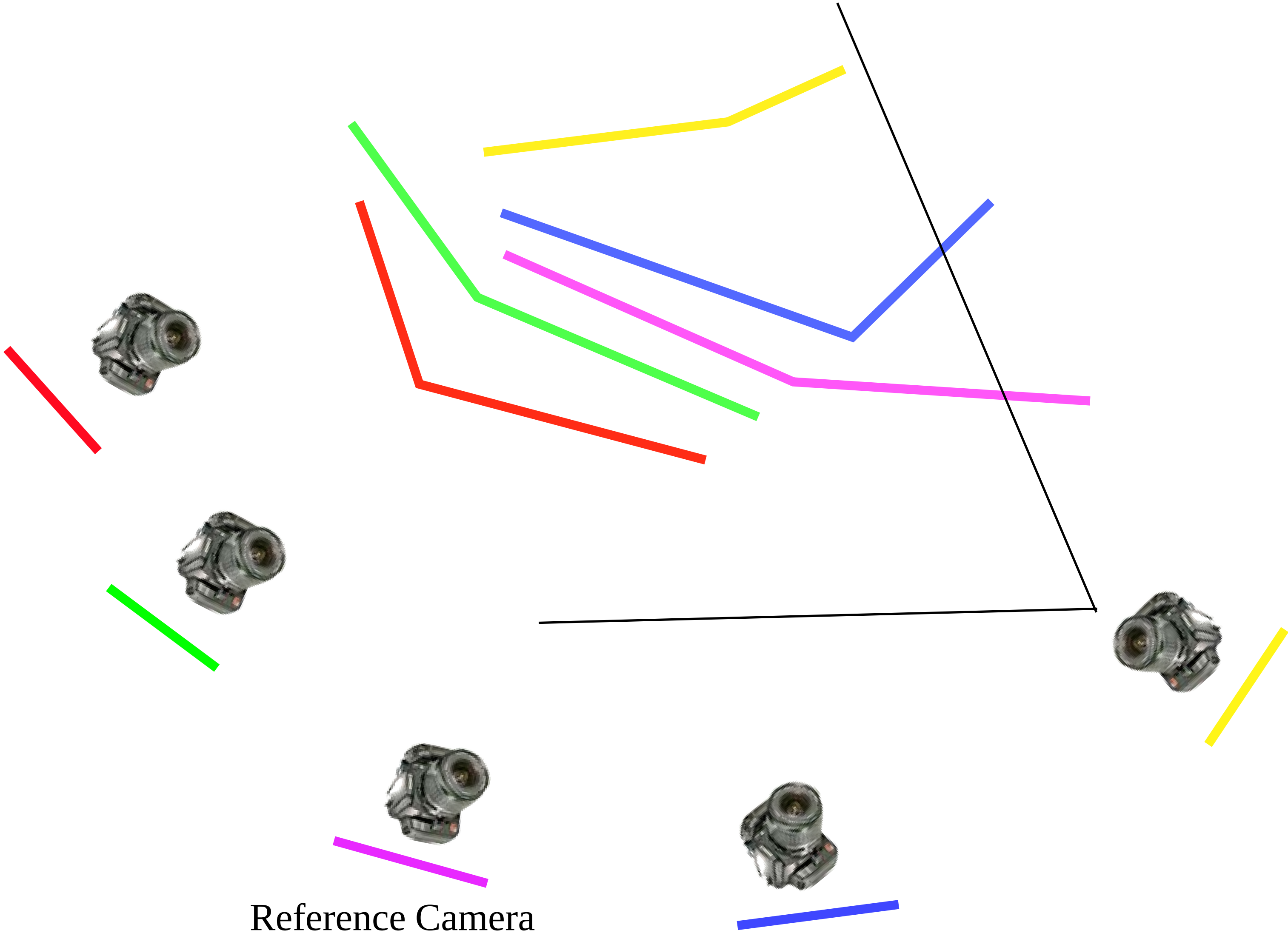


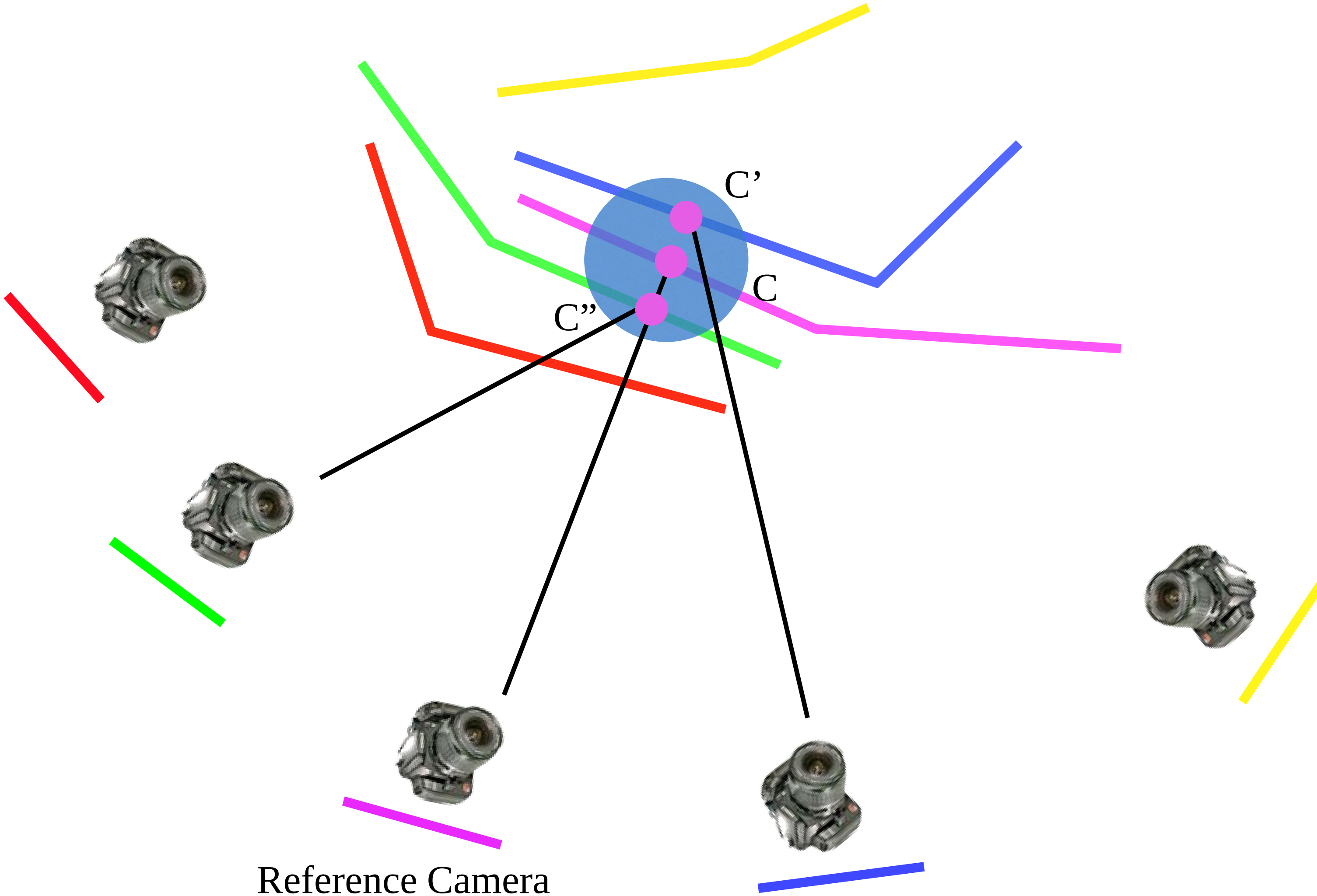
Reference Camera



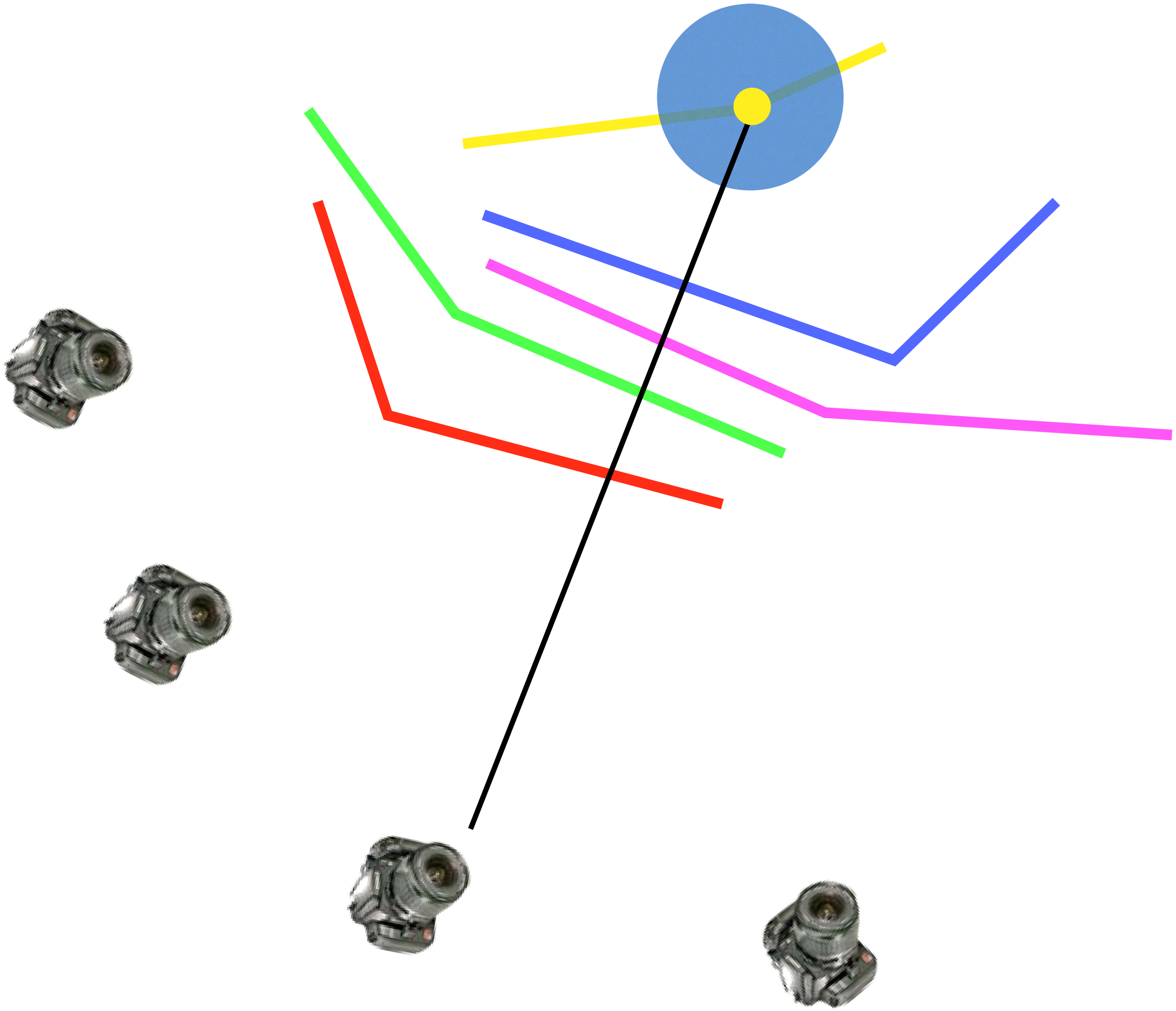
Reference Camera


Reference Camera

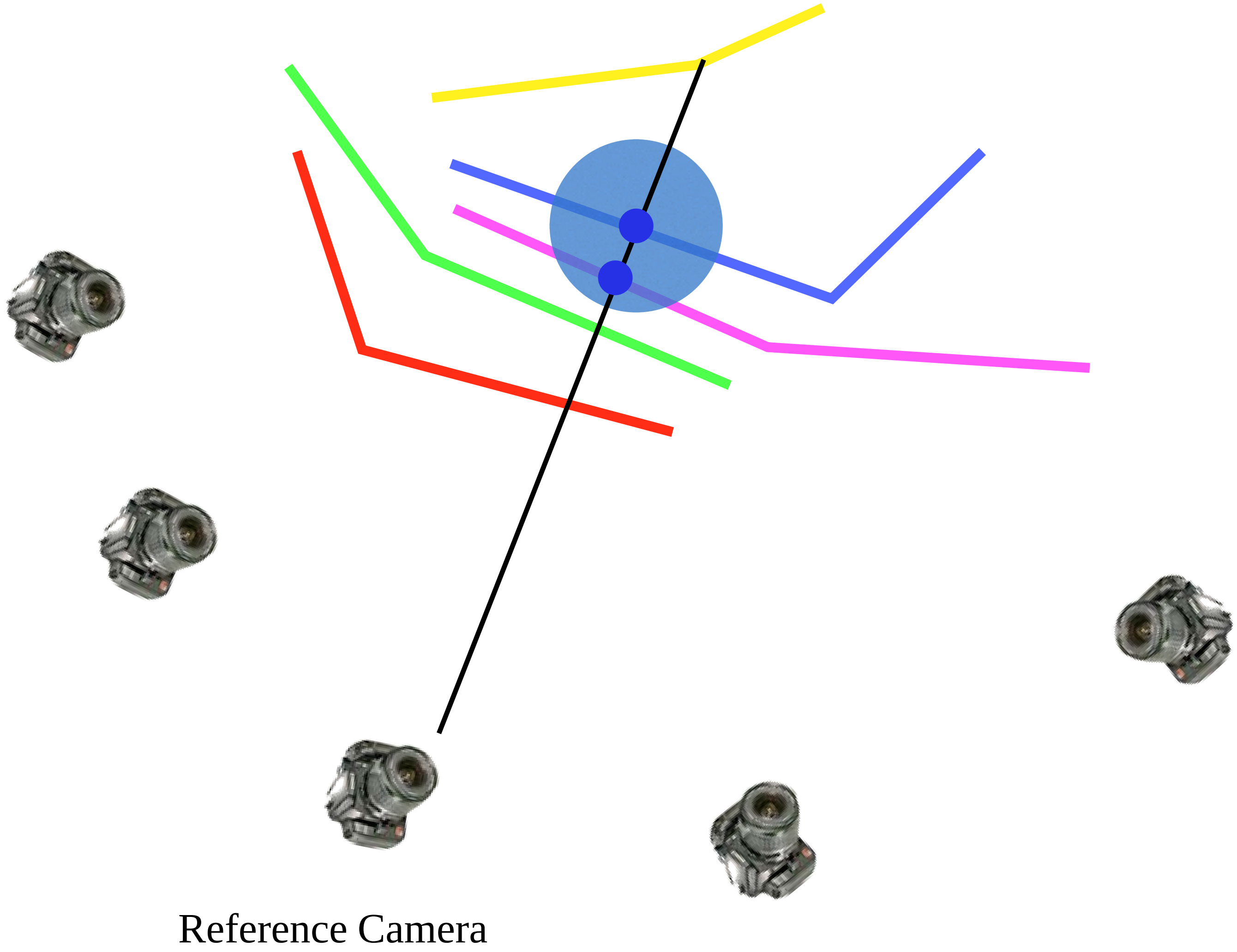

Reference Camera

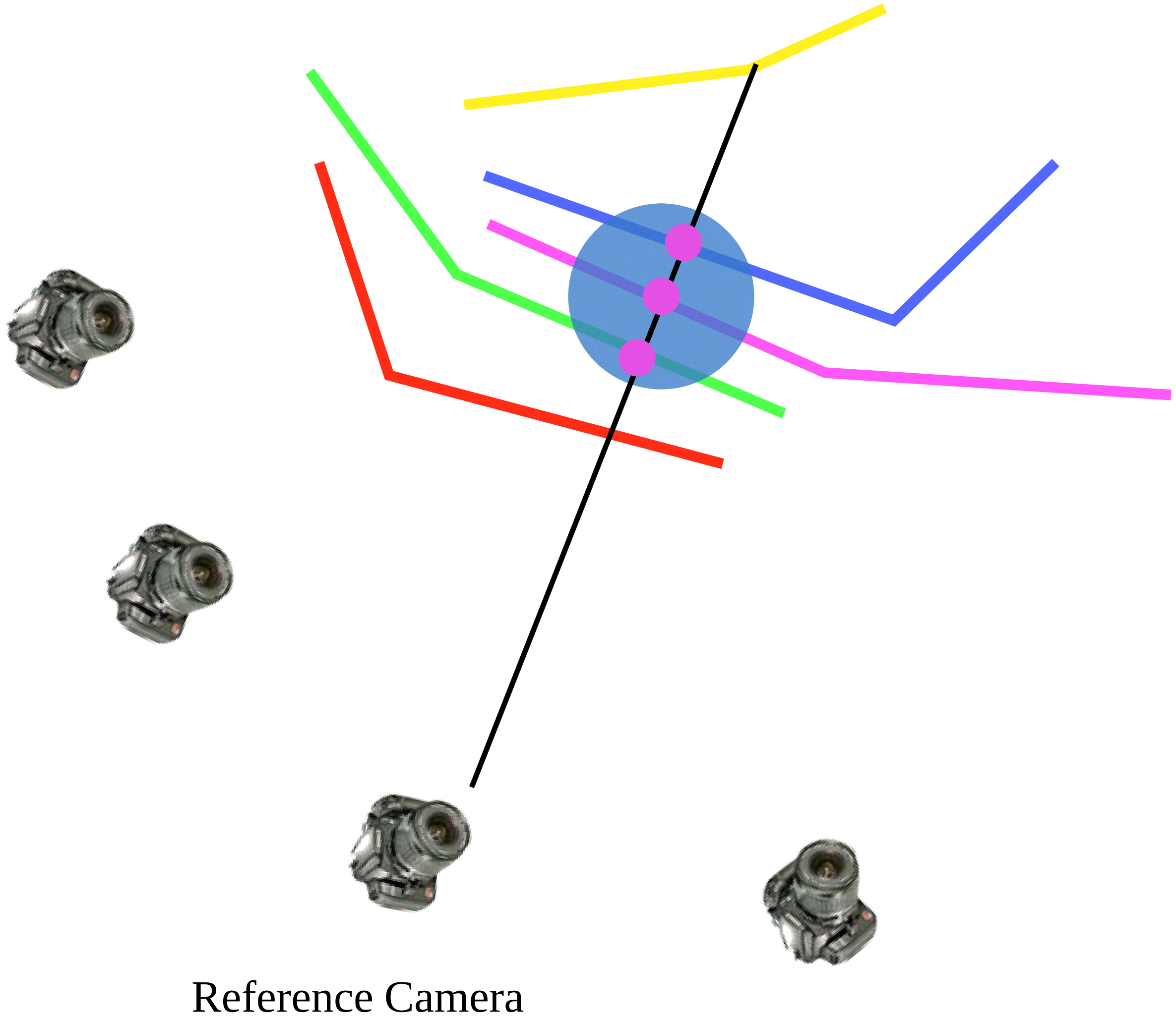


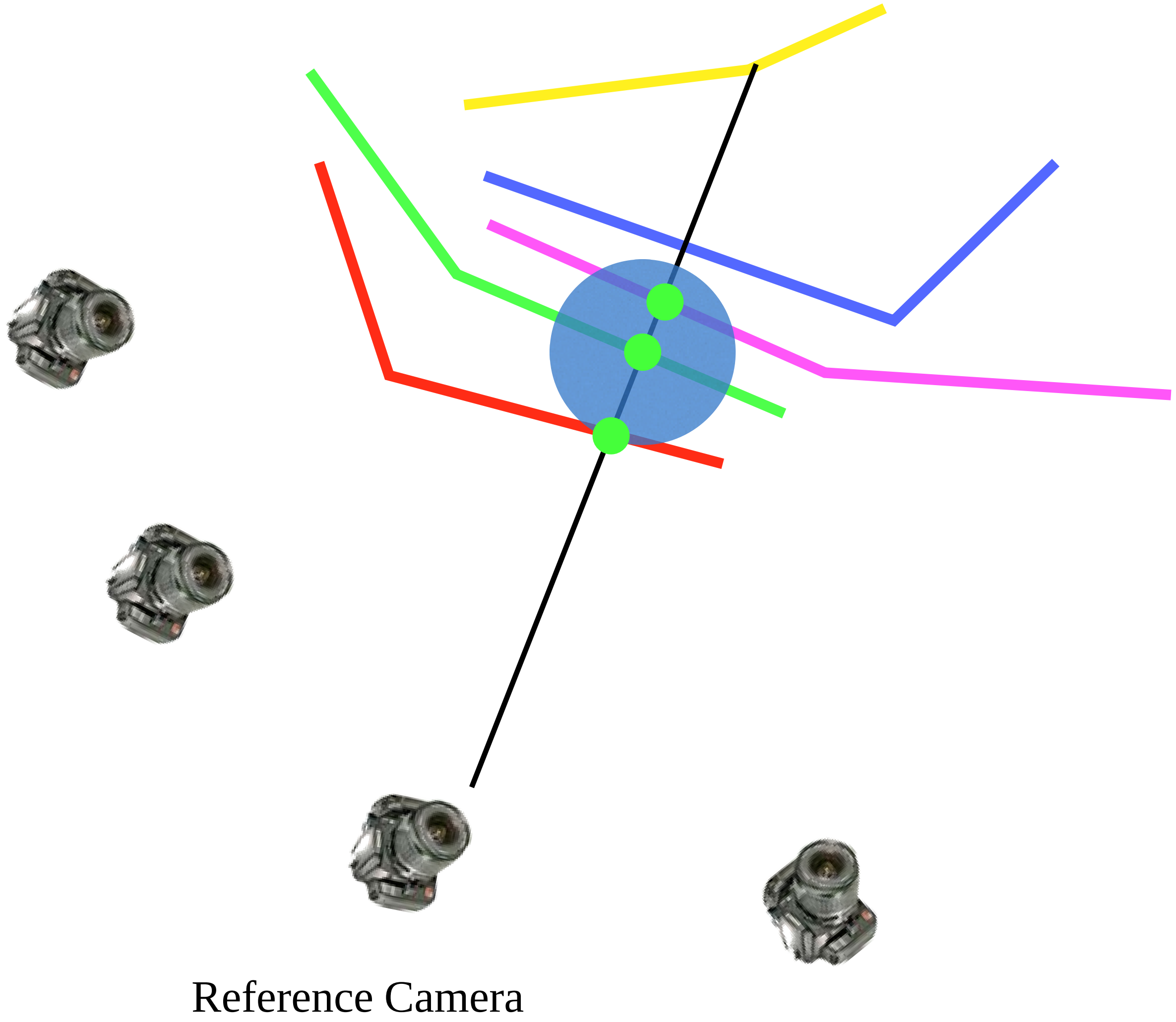
Support is calculated for every depth hypothesis within range determined by geometric uncertainty

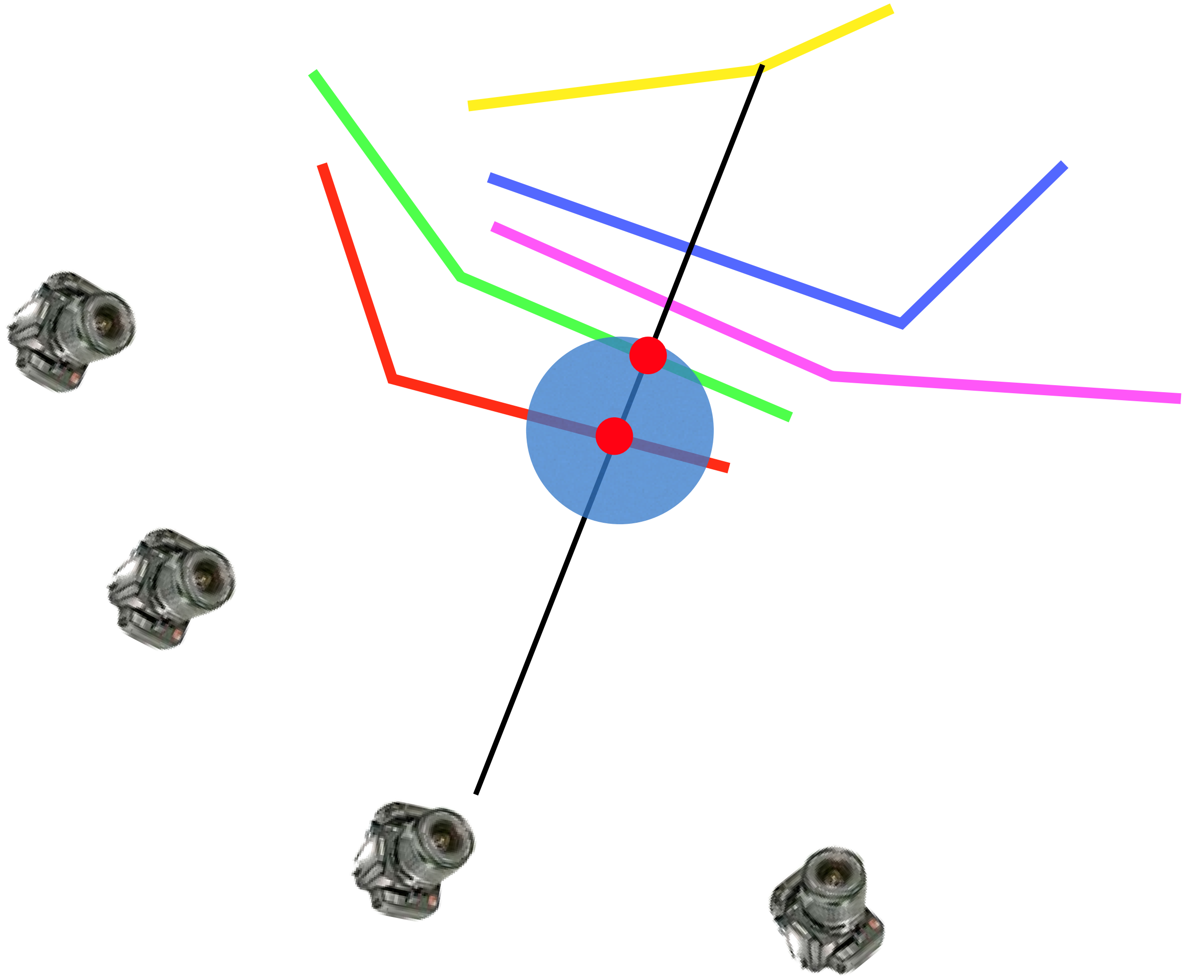


Reference Camera



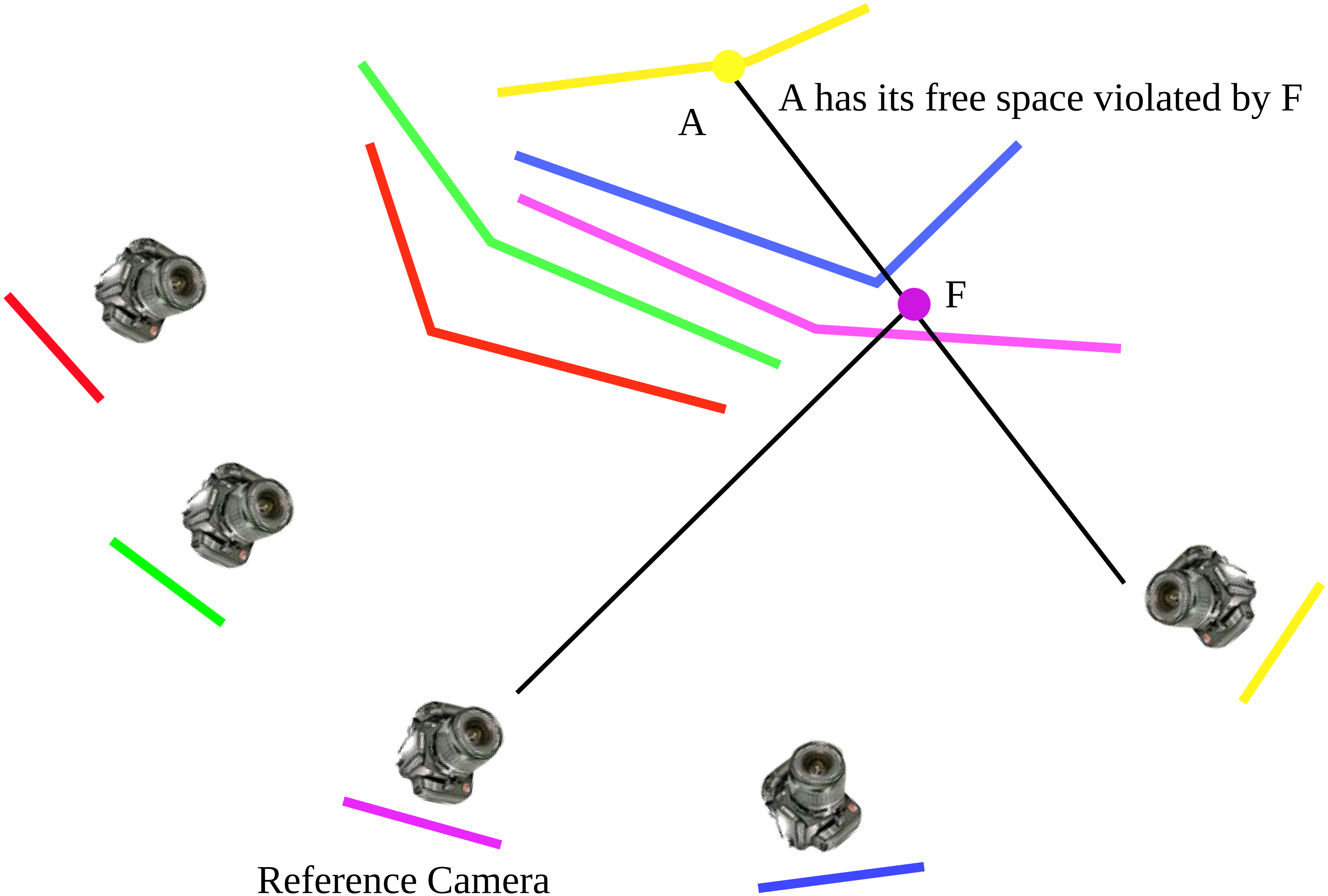

Reference Camera


Reference Camera



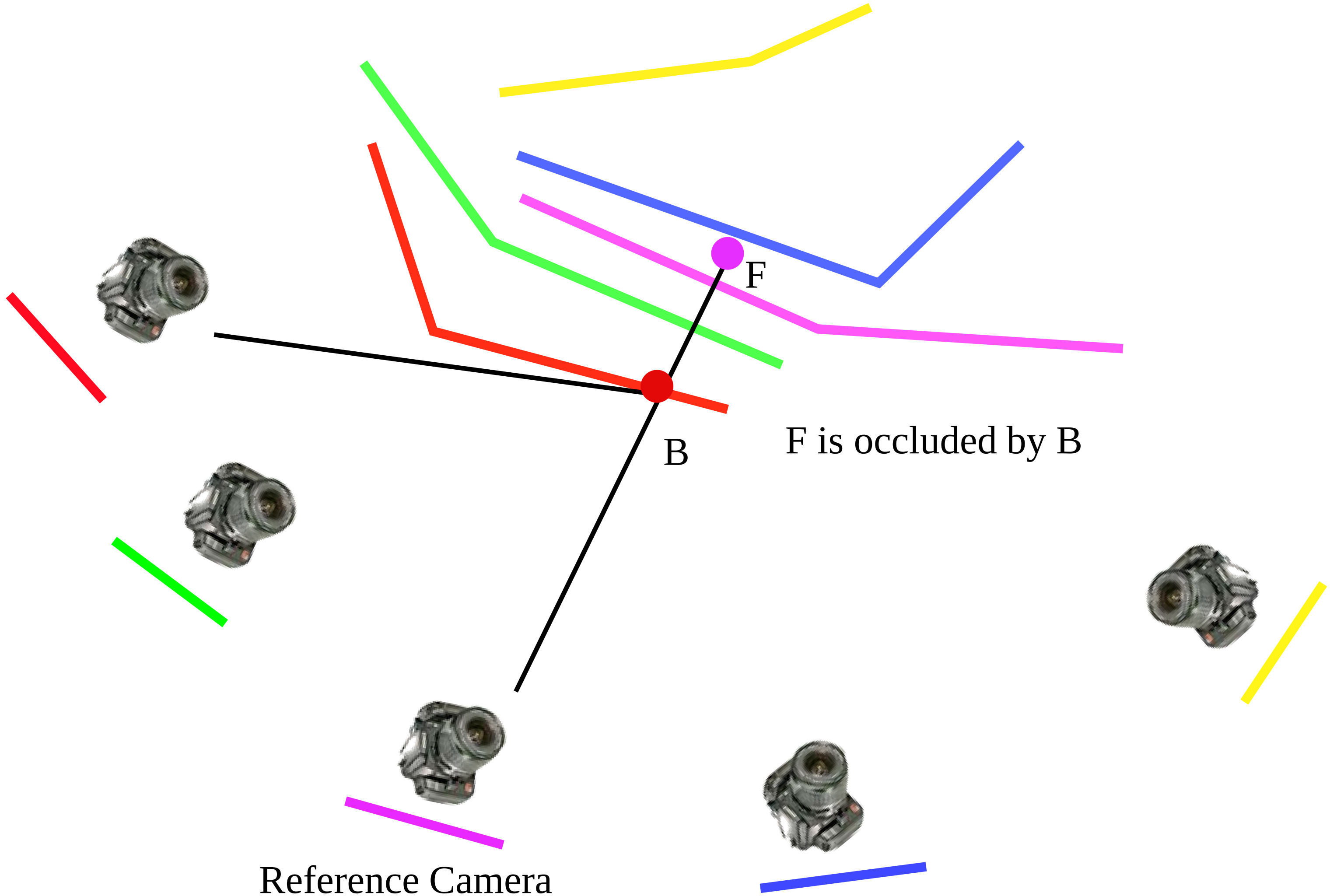
Reference Camera

Visibility Constraints



Free-space violations occur on rays of target views

Visibility Constraints



Occlusions occur on rays of the reference view

Confidence Updates

- Add confidence values of all supporting depth hypotheses
  - Fused depth is weighted average of supporting depths
- Decrease confidence if there are visibility violations

Hypothesis Selection

- Select blended hypothesis with the highest confidence for each pixel
- The number of supporting depth candidates is also considered
- Holes are filled by median filtering

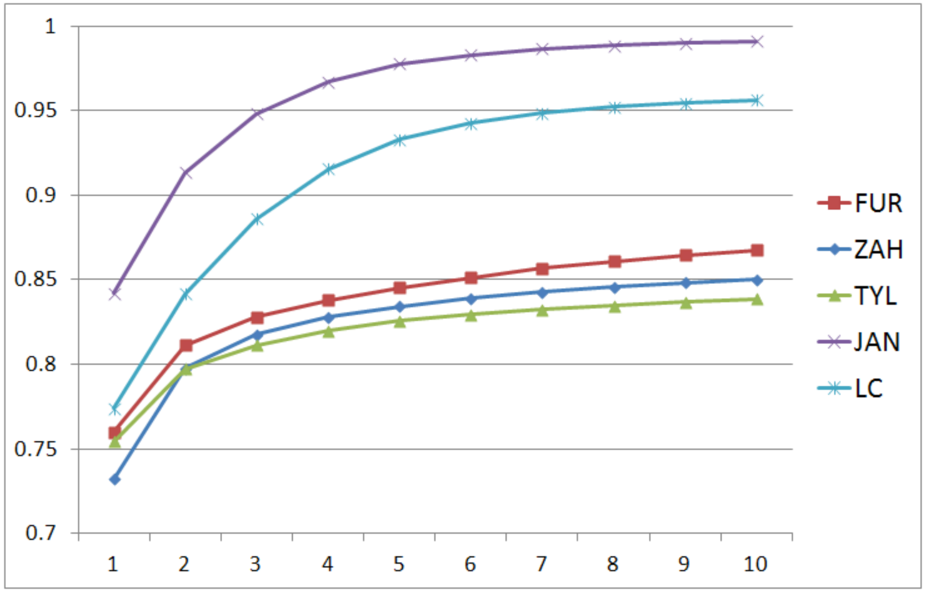
Experimental Results

Final fused depth maps are evaluated in terms of absolute errors and relative errors

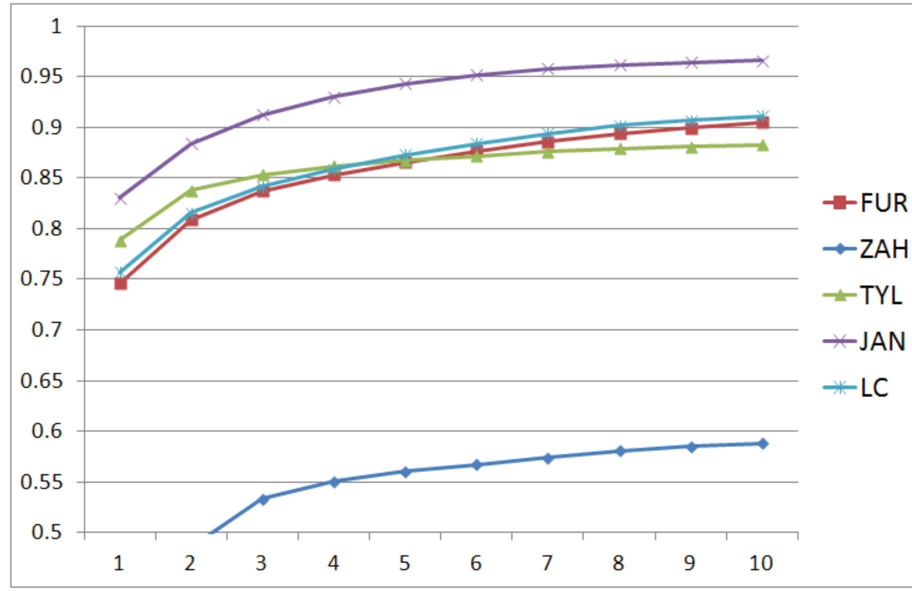
$$e_{abs} = |Z - Z_{GT}|$$

$$e_{rel} = \frac{|Z - Z_{GT}|}{\frac{Z_{GT}^2}{bf}}$$

Comparison With State of the Art



Relative error of fountain-P11

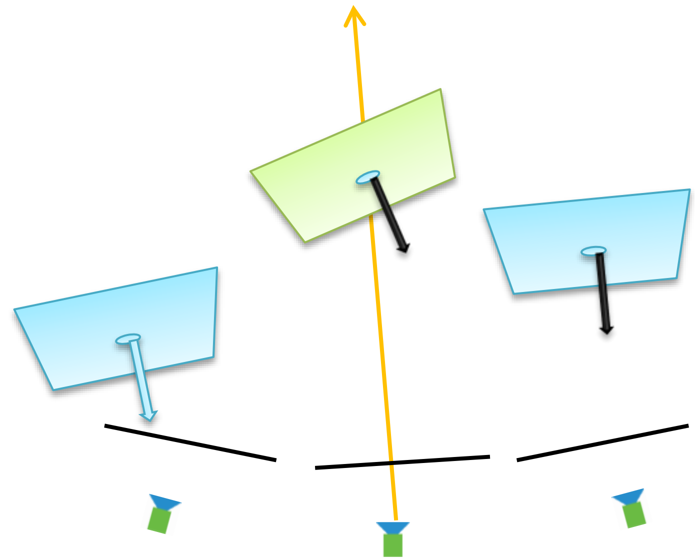


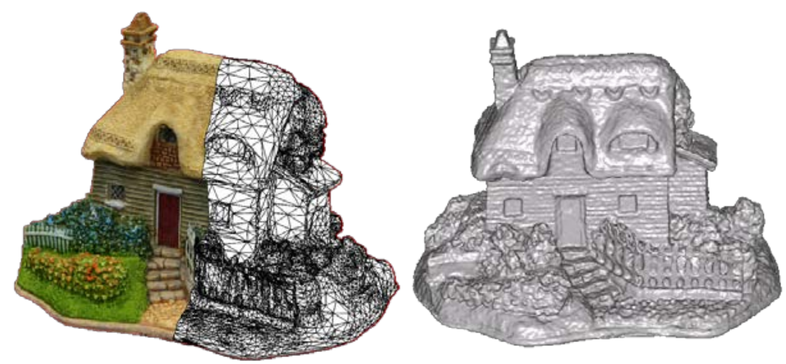
Relative error of Herz-Jesu-P8

LC: Hu and Mordohai, 2012.
FUR: Y. Furukawa and J. Ponce, 2010.
ZAH: A. Zaharescu, E. Boyer, and R. P. Horaud, 2011.
TYL: R. Tylecek and R. Sara, 2010.
JAN: M. Jancosek and T. Pajdla, 2011.

Representations

- Depth maps
- Point clouds
- Surface patches
- Level sets
- Voxels grids
- Meshes
- …





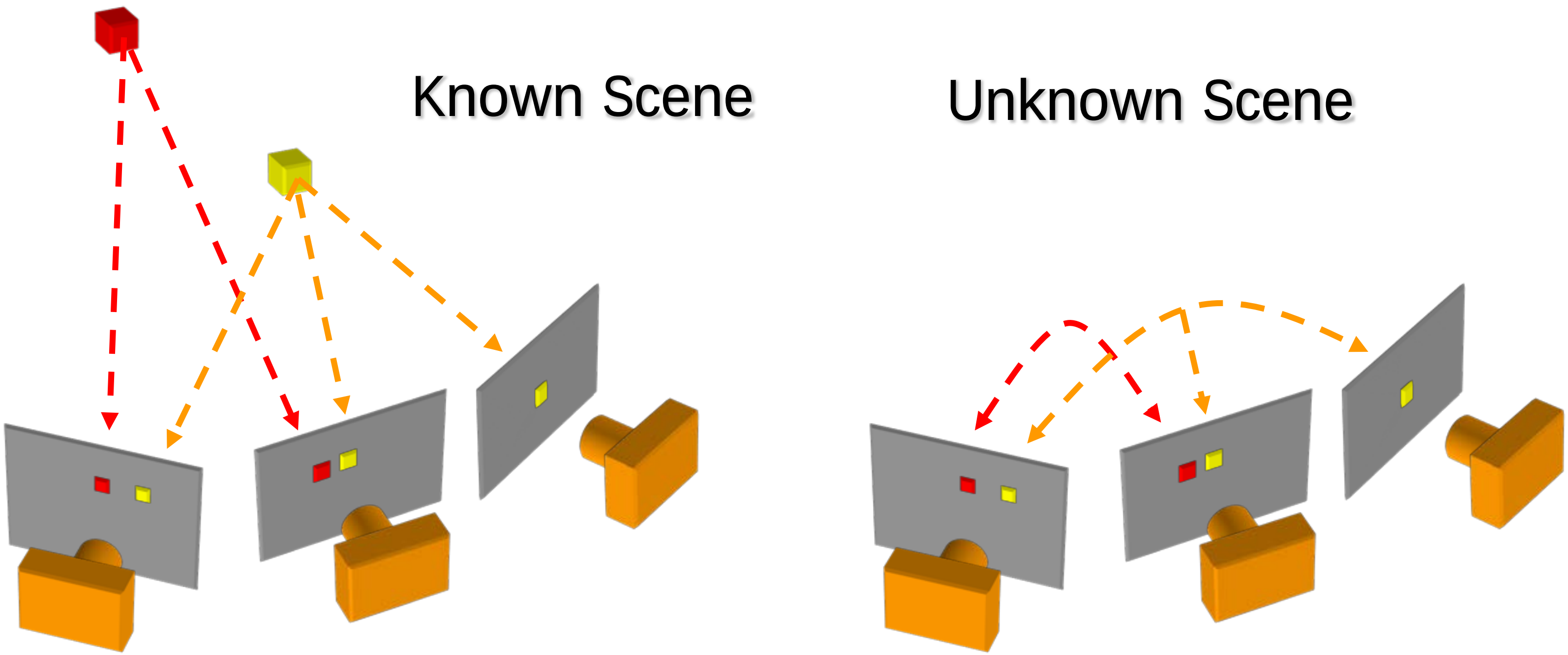
Depth Maps

- Compact representation
  - 3D quantities (points) can be indexed via pixel coordinates
  - Easy to determine neighborhood relationships and connectivity
  - Plane sweeping can be done very efficiently on GPUs
- Enable straightforward visibility estimation
- Viewpoint dependent
  - Does not allow more than one layer (2 ½ D representation)
  - Viewpoint cannot be altered without revealing holes

- Depth maps are no consistent after fusion
  - They are redundant when they are consistent…

The Visibility Problem

- Which points are visible in which images?



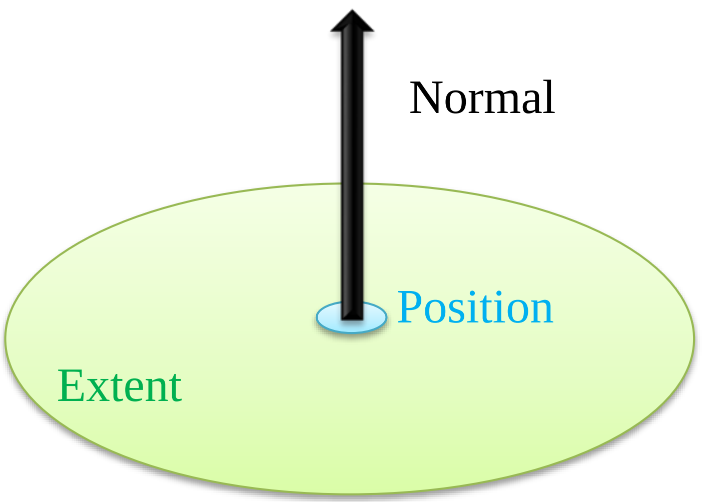
Forward Visibility *Inverse Visibility*

Patch-based MVS (PMVS)

Y. Furukawa and J. Ponce (PAMI 2010)

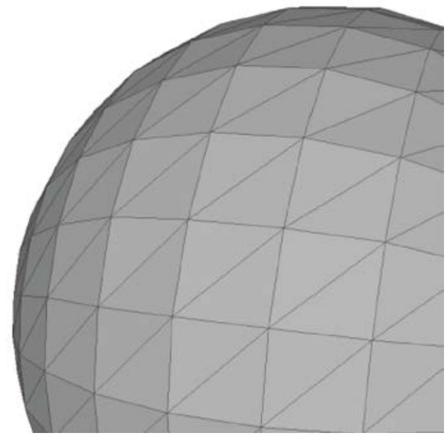
What is a Patch?

- Patch consists of
  - Position (x, y, z)
  - Normal (n_x, n_y, n_z)
  - Extent (*radius*)
- Tangent plane approximation

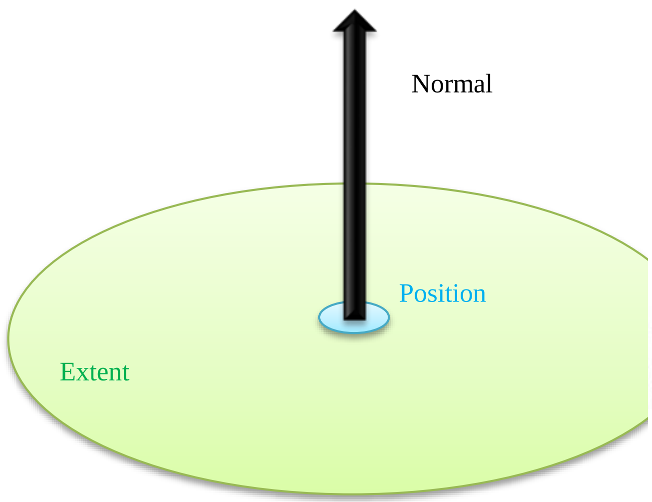


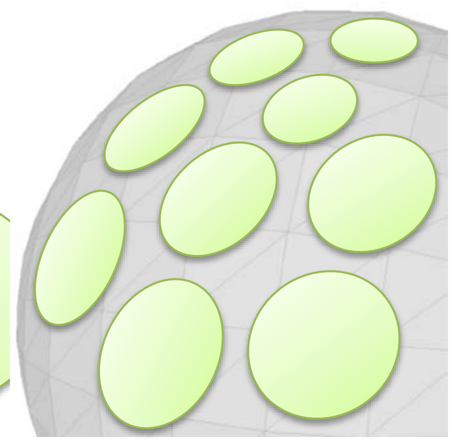
What is a Patch?

- Patch consists of
  - Position (*x, y, z*)
  - Normal (*nx, ny, nz*)
  - Extent (*radius*)
- Tangent plane approximation



Mesh





Patch

Why Patches?

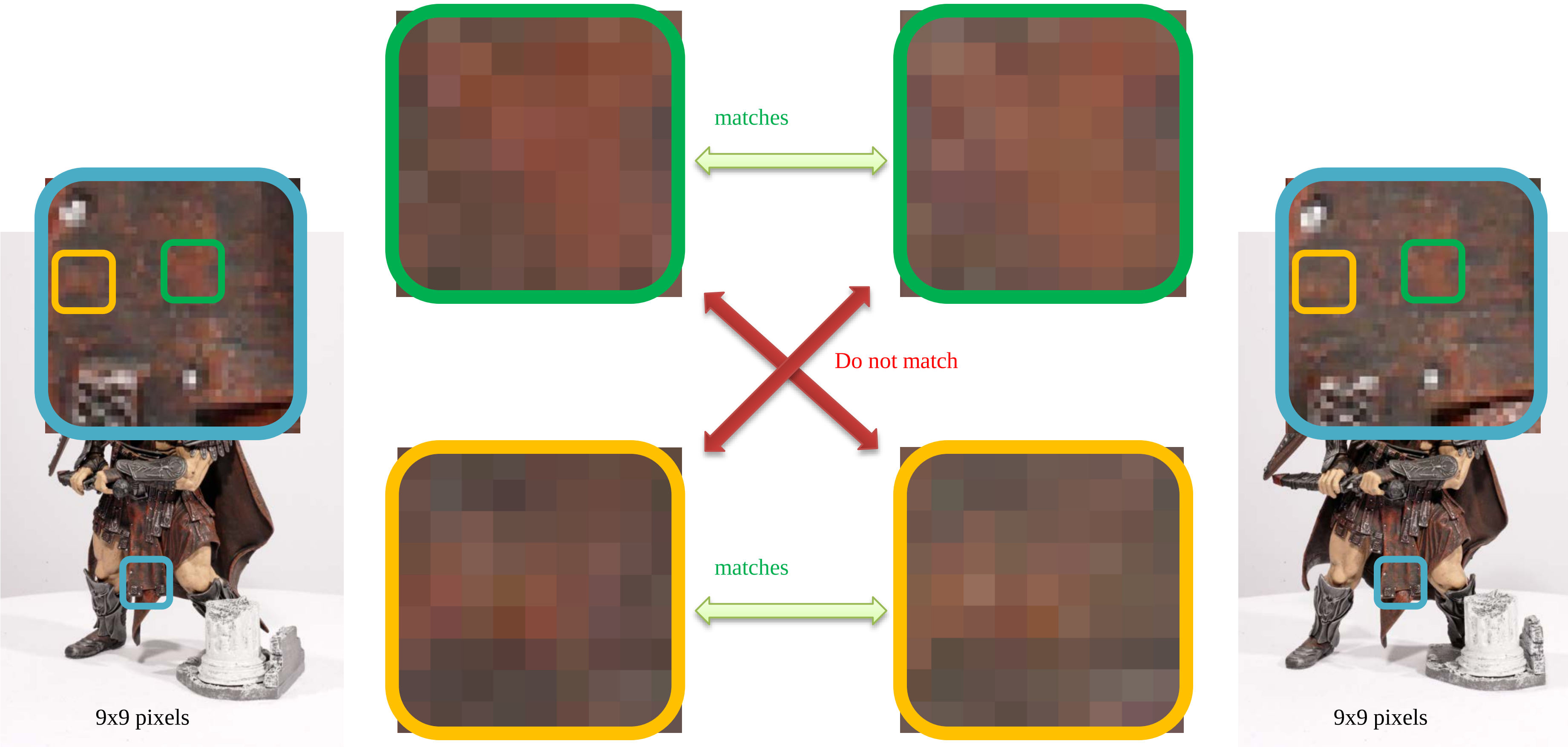
- Flexible

Why Patches?

- Flexible ⟺ Hard to enforce regularization

Why Patches?

- Flexible ⟺ Hard to enforce regularization



Why Patches?

- Flexible ⟺ Hard to enforce regularization

Regularization not really necessary

because

Local image patch is descriptive enough

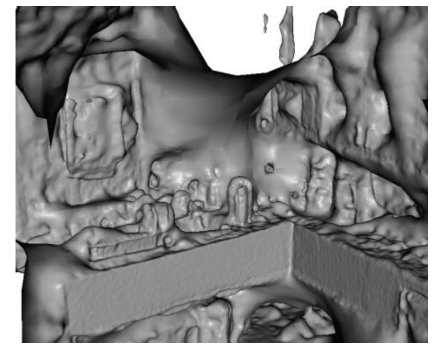
9x9 pixels

Why Patches?

- Pure 3d data w/o interpolation

Image

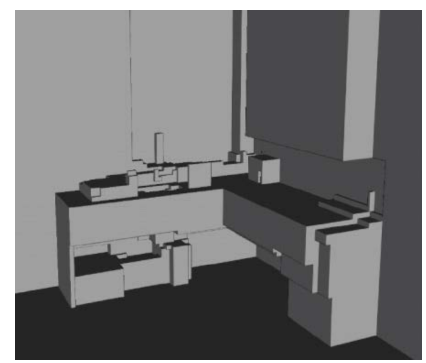
Meshing w/
standard interpolation



Patches (pure 3d data)

Scene analysis from
pure 3d data

Meshing w/
smart interpolation



By the way…

- Depthmap-fusion
  - is also “flexible”
  - can also extract pure 3d data

Patches vs. Multiple Depth Maps (according to Y. Furukawa)

- Patches → Single global 3D model
  Depthmaps → Multiple redundant 3D models

- Patches → Clean 3D points
  Depthmaps → Noisy without merging

- Patches → Hard to compute fast
  Depthmaps → Easy to compute fast

Patch-based MVS

[Lhuillier and Quan,
PAMI 05]

[Furukawa and Ponce,
CVPR 07 and PAMI 2010]

[Habbecke and Kobbelt,
CVPR 07]

Patch Definition

- Patch p is defined by
  - Position c(p)
  - Normal n(p)
  - Visible images V(p)
- Extent is set so that p is roughly 9x9 pixels in V(p)

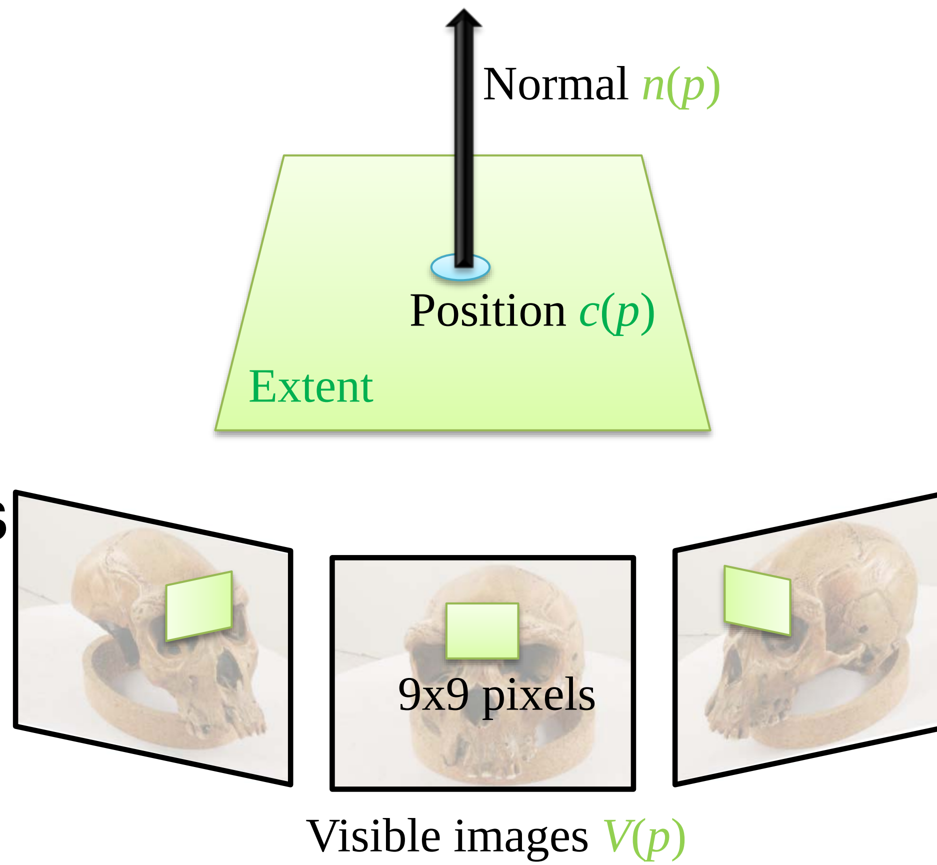


Photo-consistency

- Photo-consistency N(I, J, p) of p between two images I and J

I_{xy}: pixel color in image I

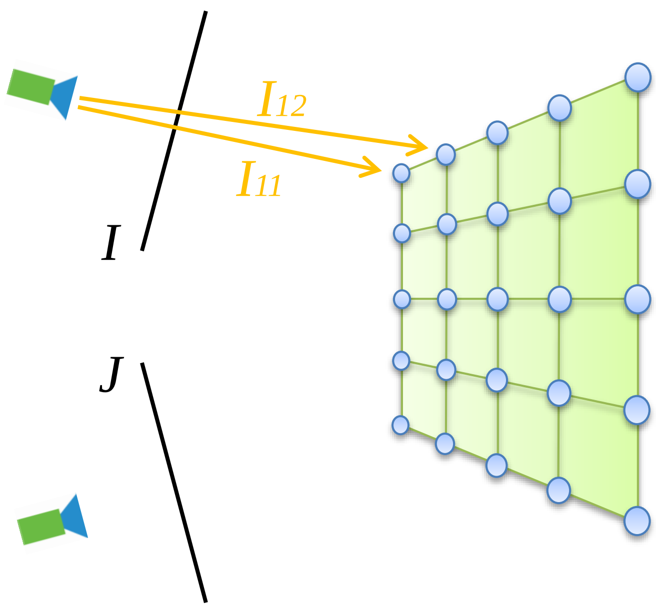
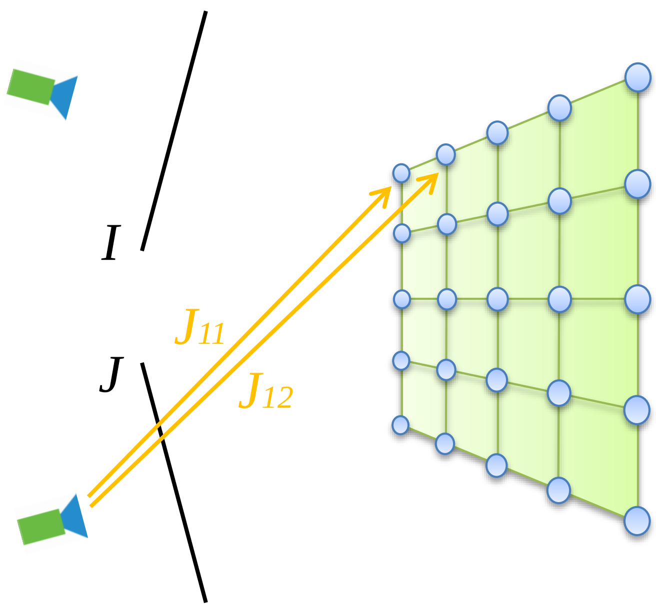


Photo-consistency

- Photo-consistency N(I, J, p) of p between two images I and J



I_{xy}: pixel color in image I

J_{xy}: pixel color in image J

$$N(I,J,p) = \frac{\sum (I_{xy} - \overline{I_{xy}}) \cdot (J_{xy} - \overline{J_{xy}})}{\sqrt{(I_{xy} - \overline{I_{xy}})^2} \sqrt{(J_{xy} - \overline{J_{xy}})^2}}$$

Photo-consistency

- Photo-consistency N(I, J, p) of p between two images I and J

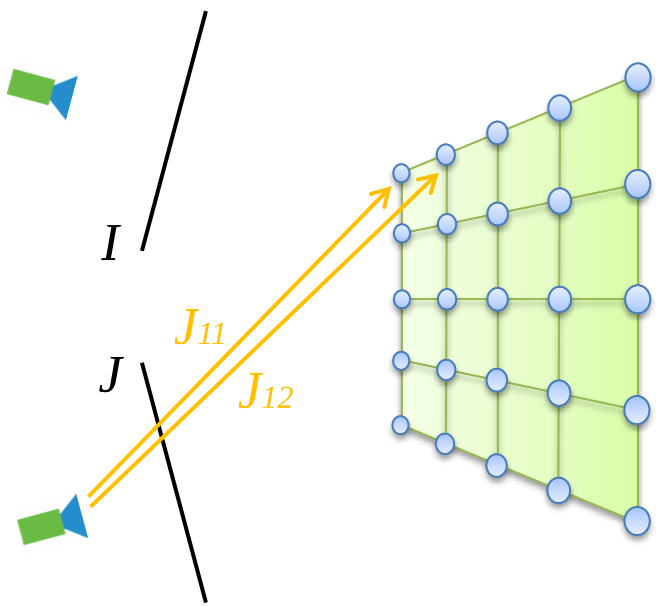
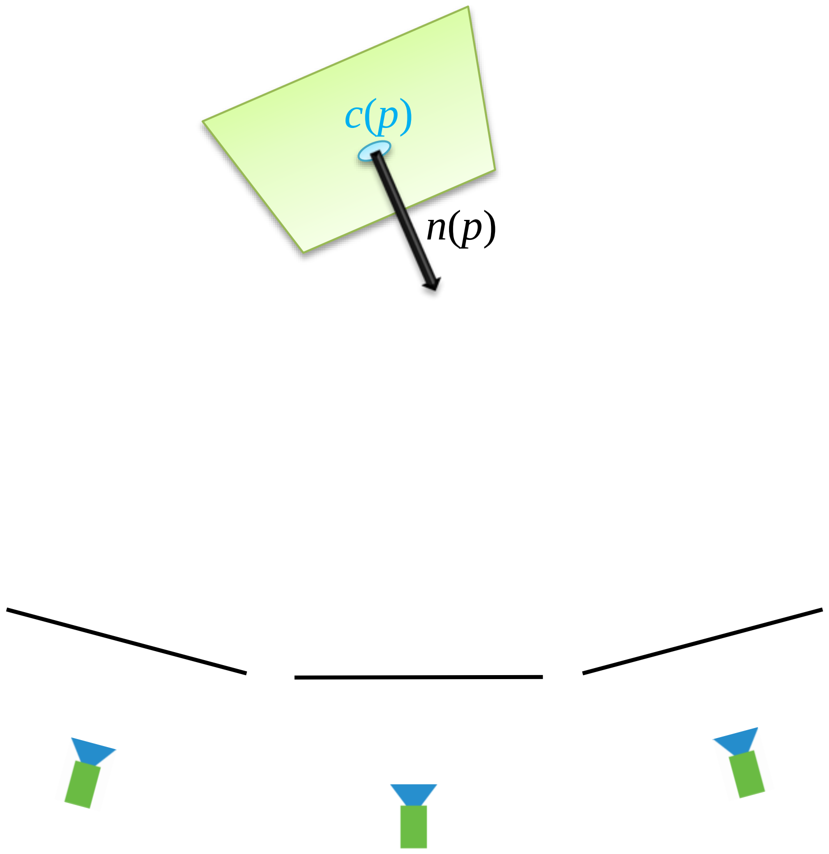


Photo-consistency $N(p)$ of p with visible images $V(p) = \{I_1, I_2, \ldots, I_n\}$

$$N(p) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} N(I_i, I_j, p)}{(n+1)n/2}$$

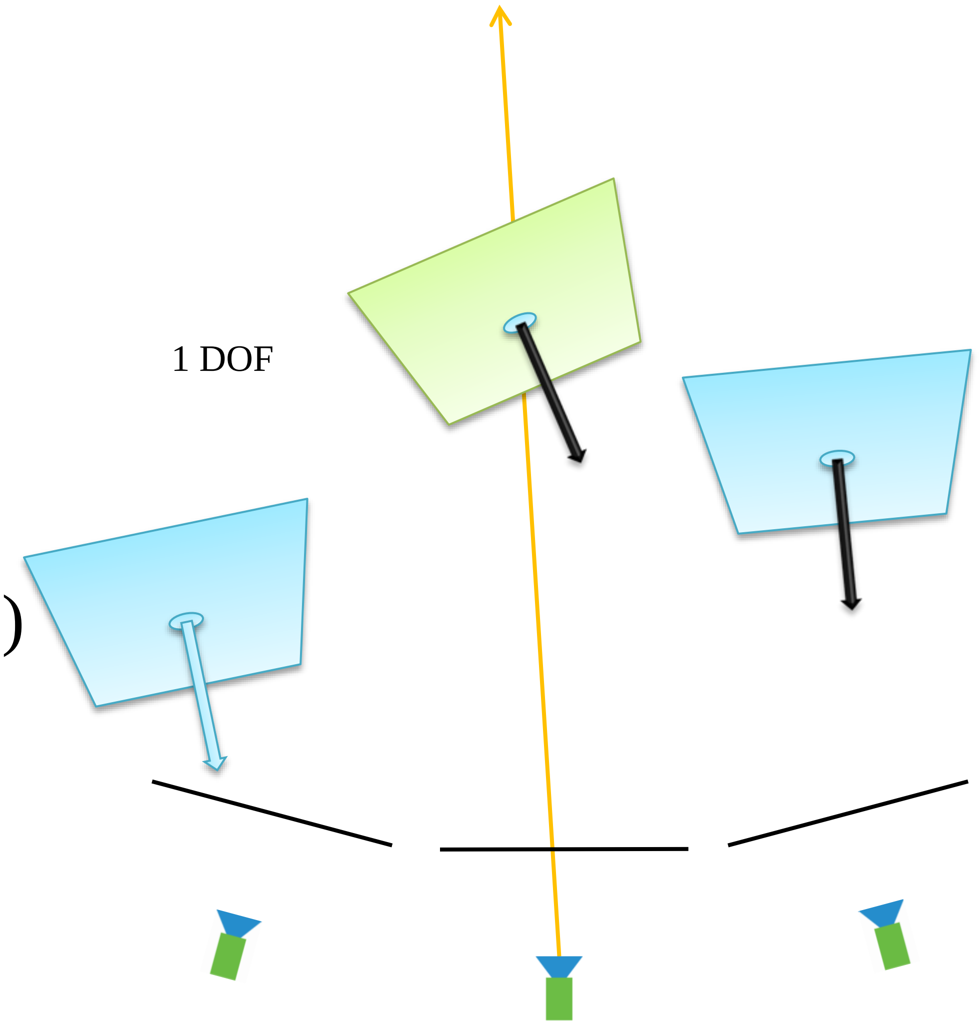
Reconstruct Patch p

- Given initial estimates of
  - Position c(p)
  - Normal n(p)
  - Visible images V(p)
- $\{c(p), n(p)\} = \underset{\{c(p),n(p)\}}{\arg\max}\, N(p)$



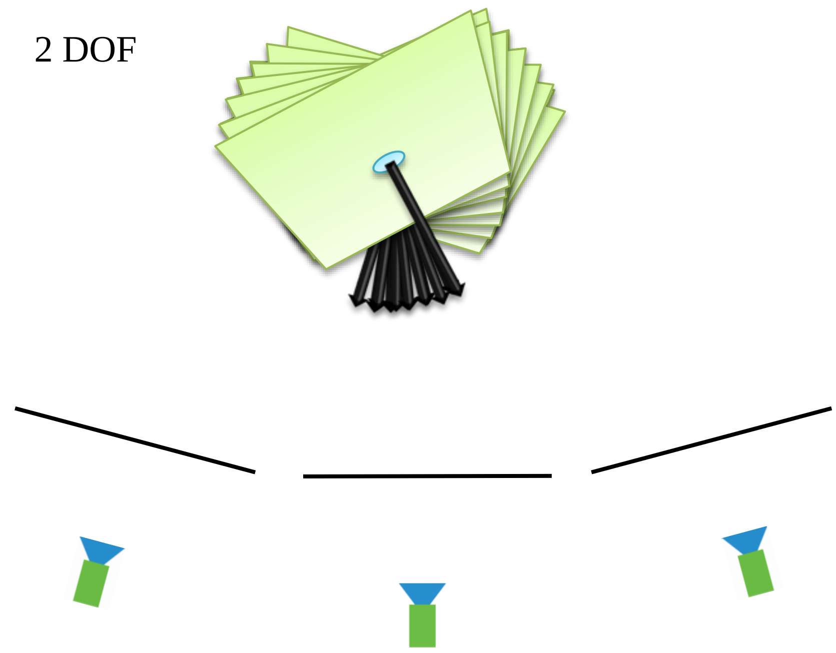
Reconstruct Patch p

- Given initial estimates of
  - Position c(p)
  - Normal n(p)
  - Visible images V(p)
- $\{c(p), n(p)\} = \underset{\{c(p),n(p)\}}{\arg\max}\, N(p)$



Reconstruct Patch p

- Given initial estimates of
  - Position c(p)
  - Normal n(p)
  - Visible images V(p)
- $$\{c(p), n(p)\} = \underset{\{c(p), n(p)\}}{\arg\max}\, N(p)$$



Verify a Patch

- Textures may match by accident
- Photo-consistency must be reasonably high

- Verification process
  - Keep only high photo-consistency images in $V(p)$
  - Accept if $|V(p)| \geq 3$

Update V(p)

$$V(p)=\{\text{Image1}, \text{Image2}, \cancel{\text{Image3}}, \text{Image4}\}$$

Image1

Image2

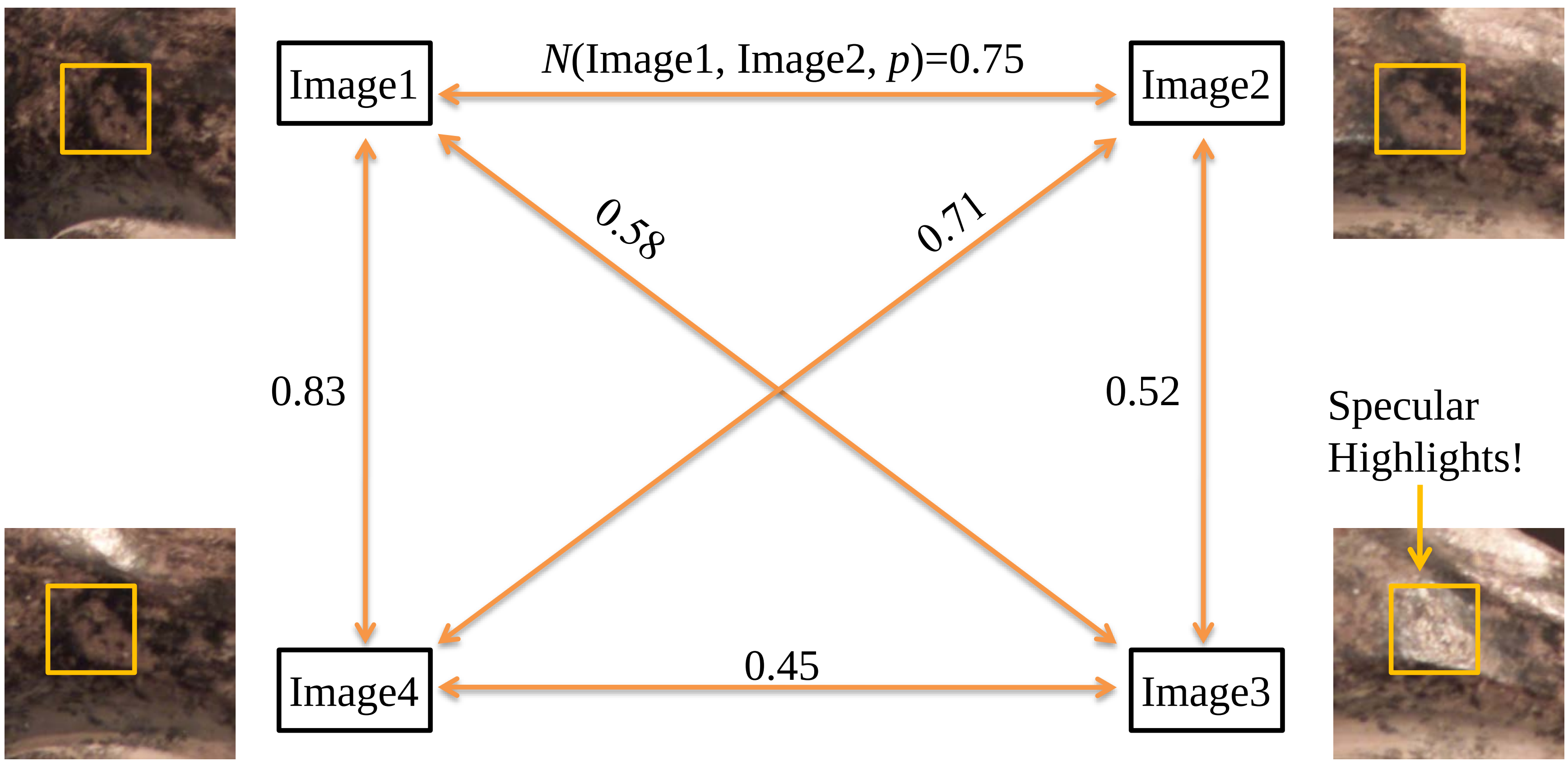
Specular Highlights!

Image4

Image3

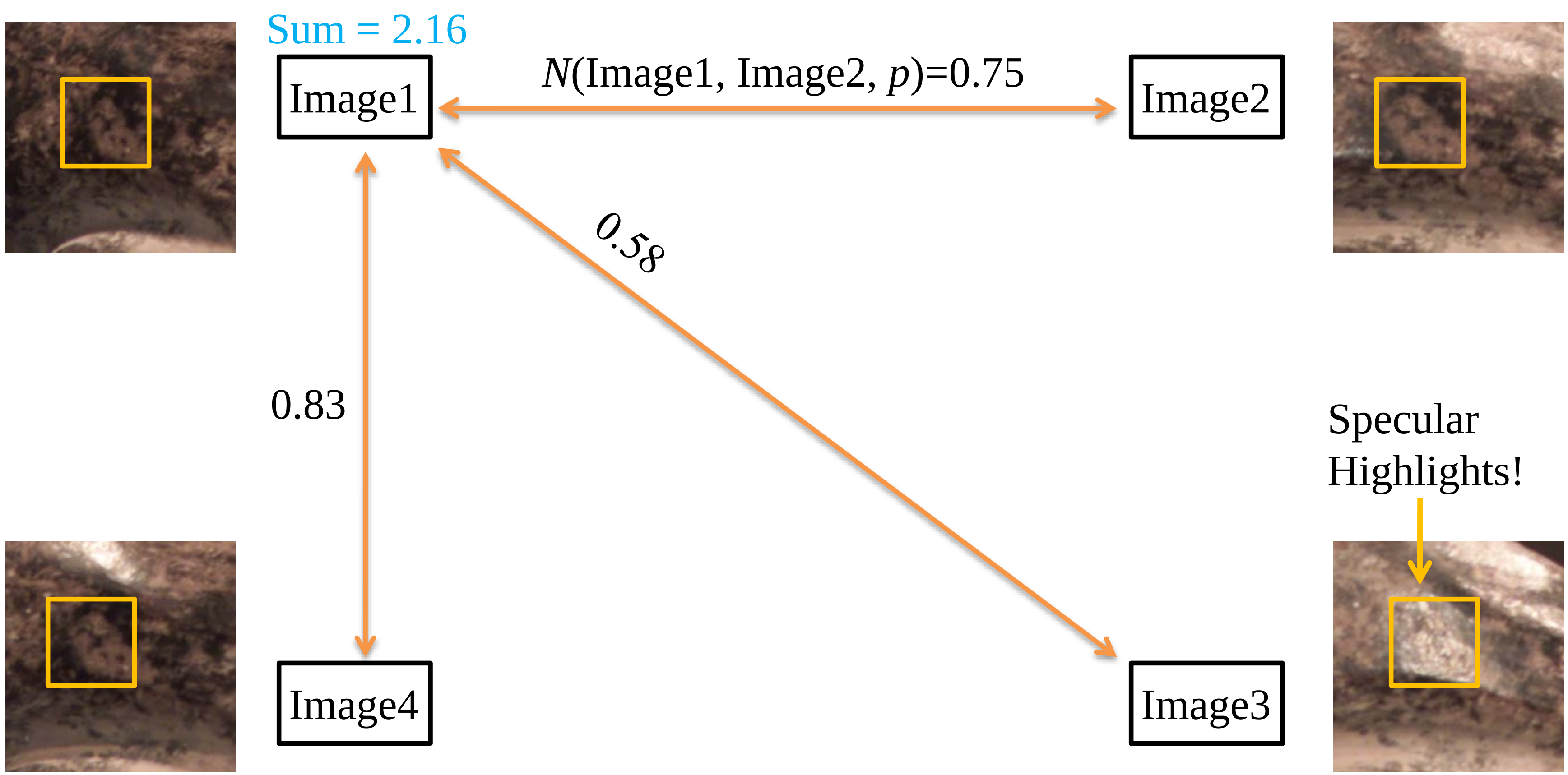
Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}



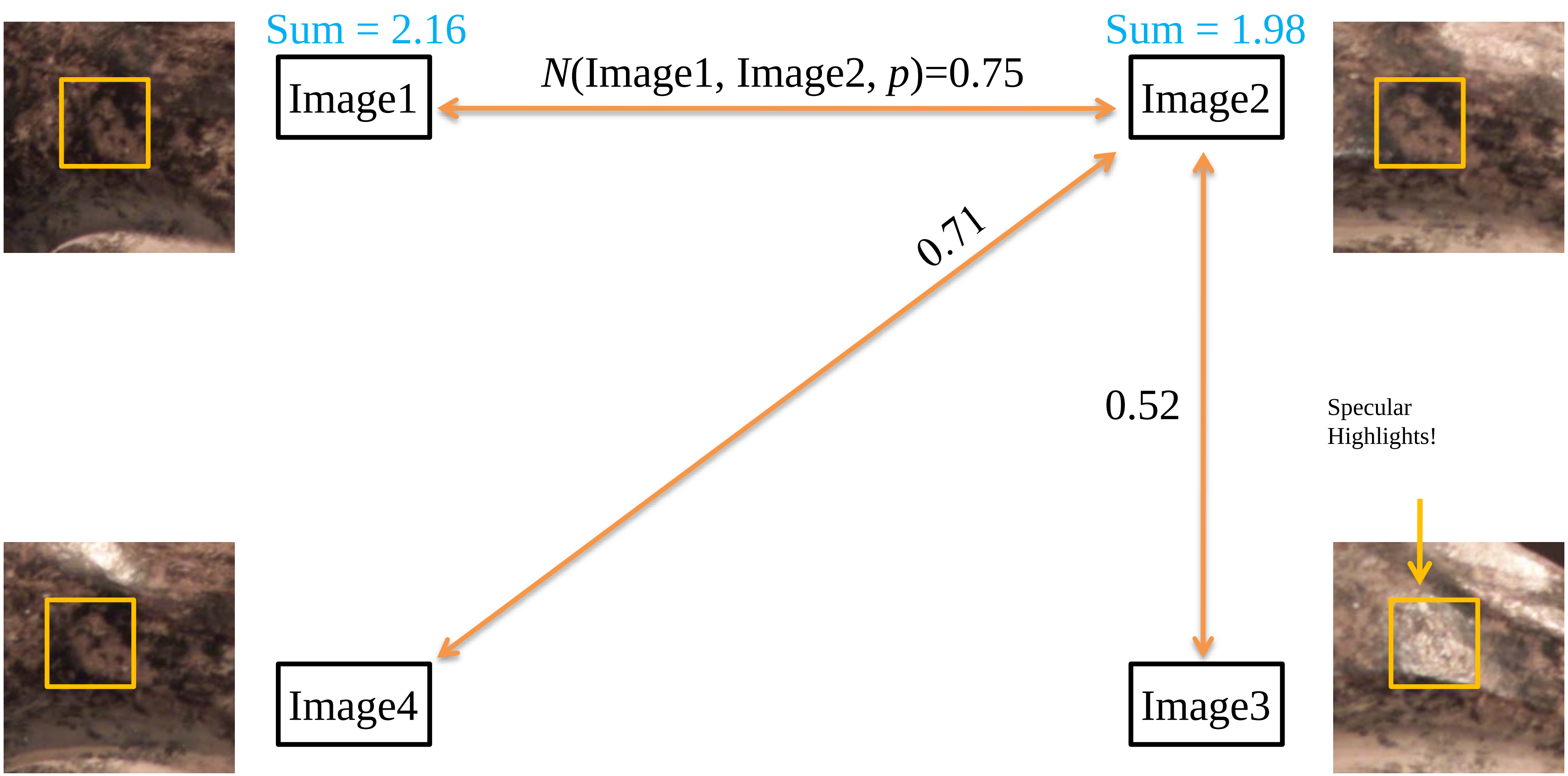
Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}



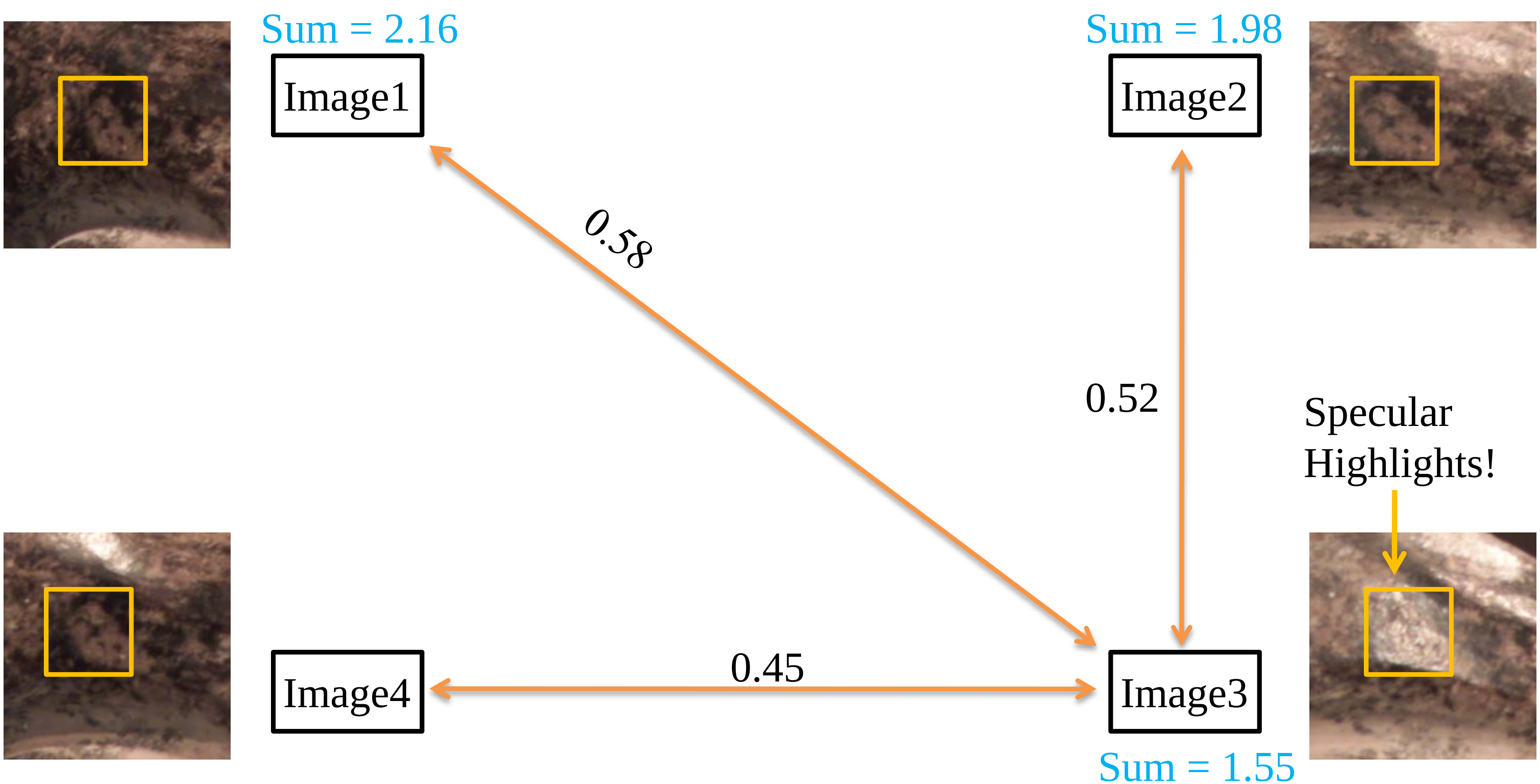
Update V(p)

$$V(p)=\{\text{Image1, Image2, Image3, Image4}\}$$



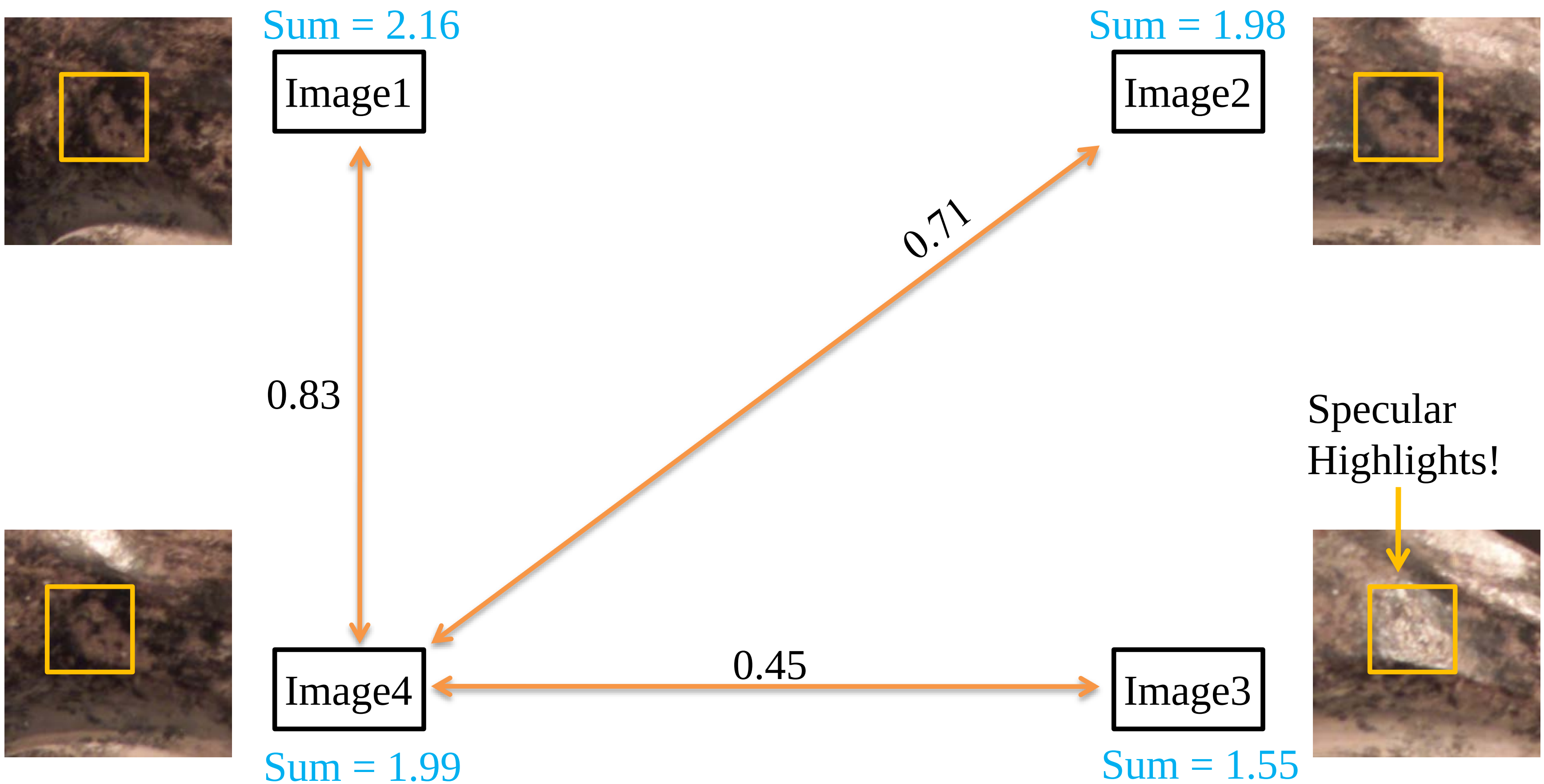
Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}



Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}



Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}

Sum = 2.16

Image1

Sum = 1.98

Image2

Specular Highlights!

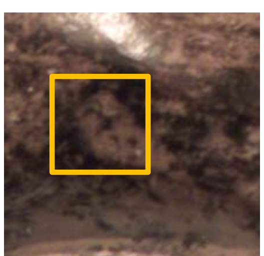


Image4

Sum = 1.99

Image3

Sum = 1.55

Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}

Sum = 2.16

Image1

Image2

Specular Highlights!

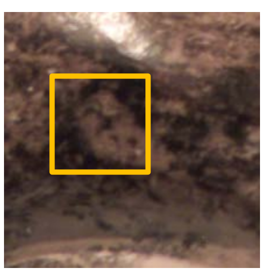


Image4

Image3

Update V(p)

$V(p)$={Image1, Image2, Image3, Image4}

Sum = 2.16

Image1

N(Image1, Image2, p)=0.75

Image2

Add Image2, because > 0.7

Specular Highlights!

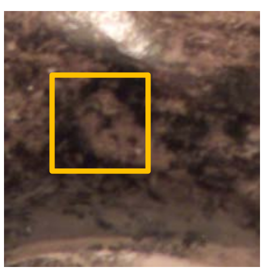


Image4

Image3

Update V(p)

$V(p)=\{\text{Image1}, \text{Image2}, \cancel{\text{Image3}}, \text{Image4}\}$

Sum = 2.16

Image1

Image2

0.58

Remove Image3, because < 0.7

Specular Highlights!

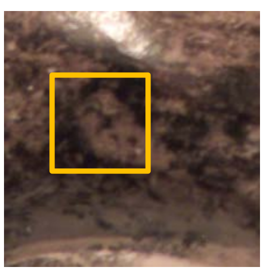


Image4

Image3

Update V(p)

$V(p)=\{\text{Image1}, \text{Image2}, \cancel{\text{Image3}}, \text{Image4}\}$

Sum = 2.16

Image1

Image2

Add Image4, because > 0.7

0.83

Specular
Highlights!

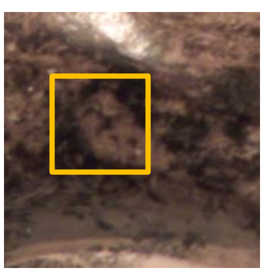


Image4

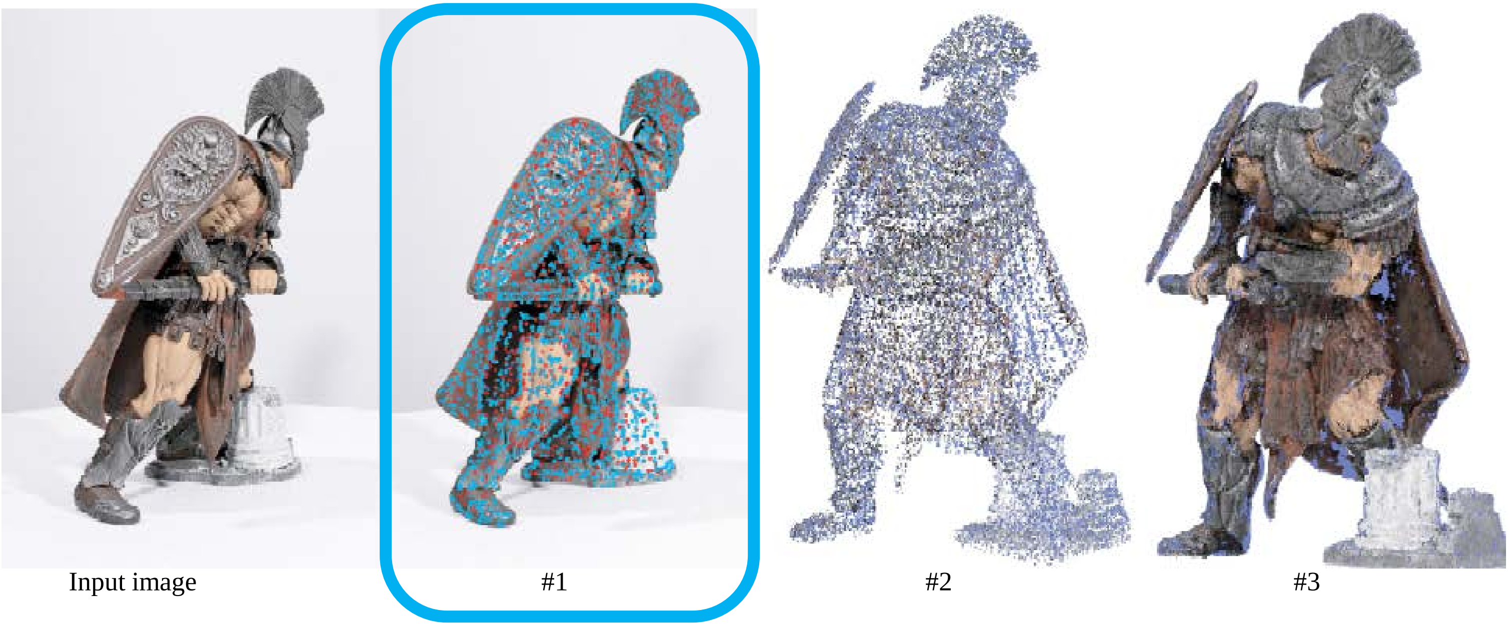
Image3

Algorithm Overview

#1. Feature detection

#2. Initial feature matching

#3. Patch expansion and filtering



Input image #1 #2 #3

Feature Detection

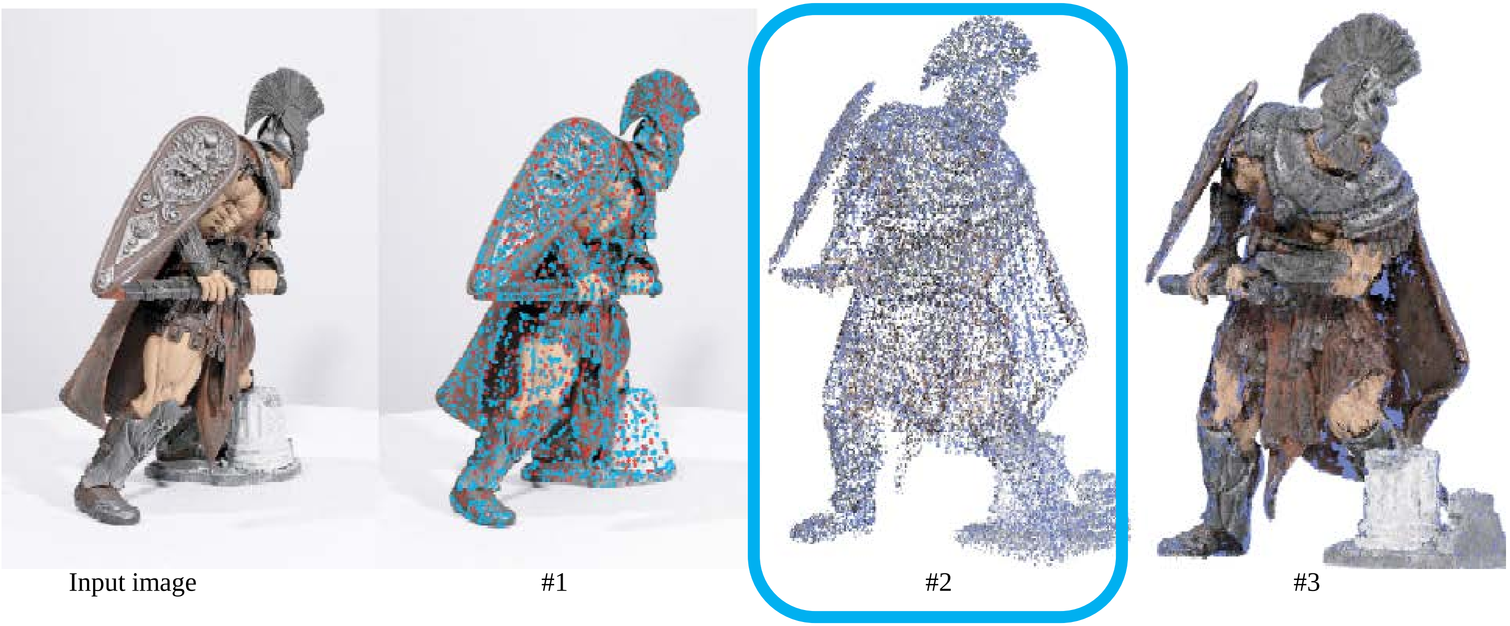
- Extract local maxima of
  - Harris corner detector (corners)
  - Difference of Gaussian (blobs)


Algorithm Overview

#1. Feature detection

#2. Initial feature matching

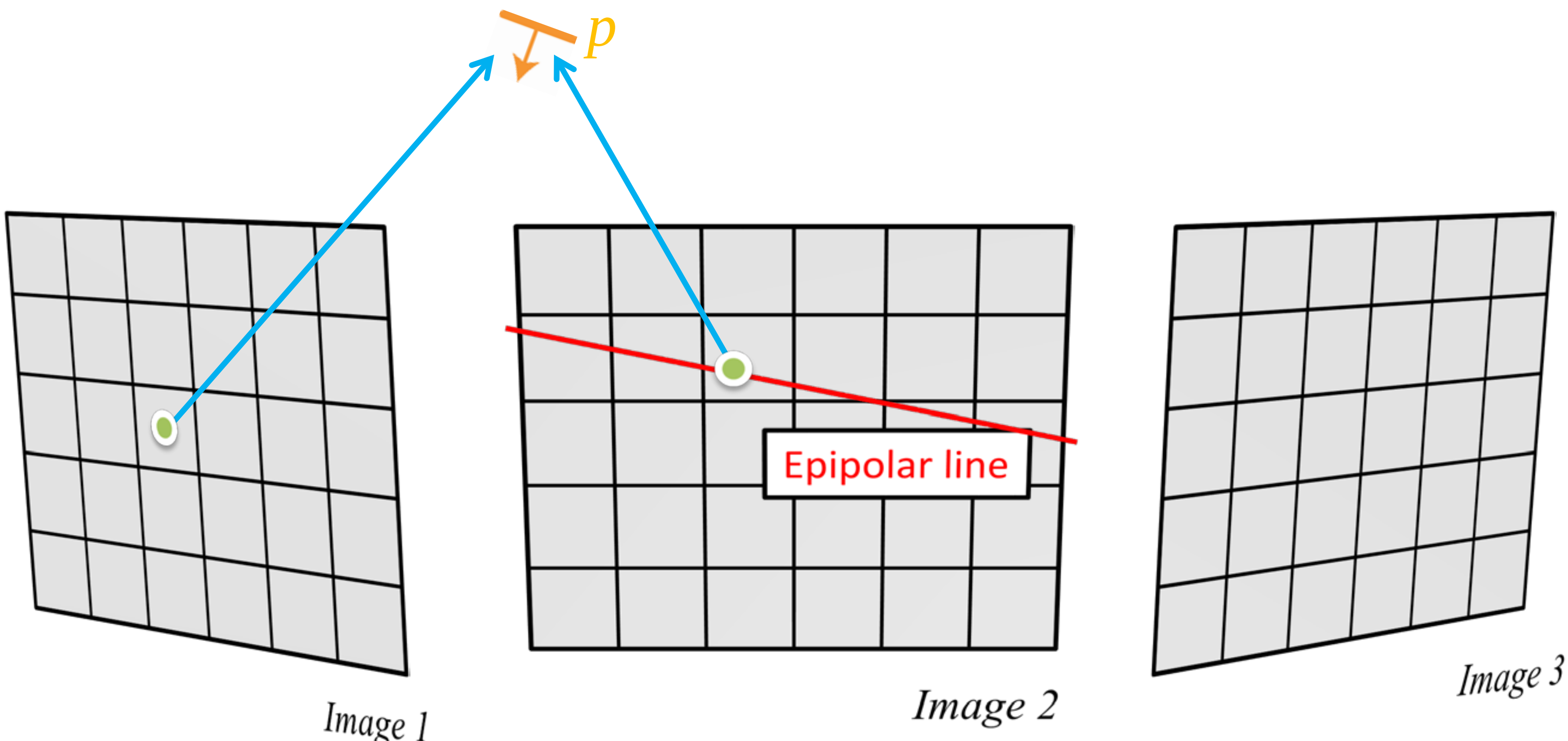
#3. Patch expansion and filtering



Input image #1 #2 #3

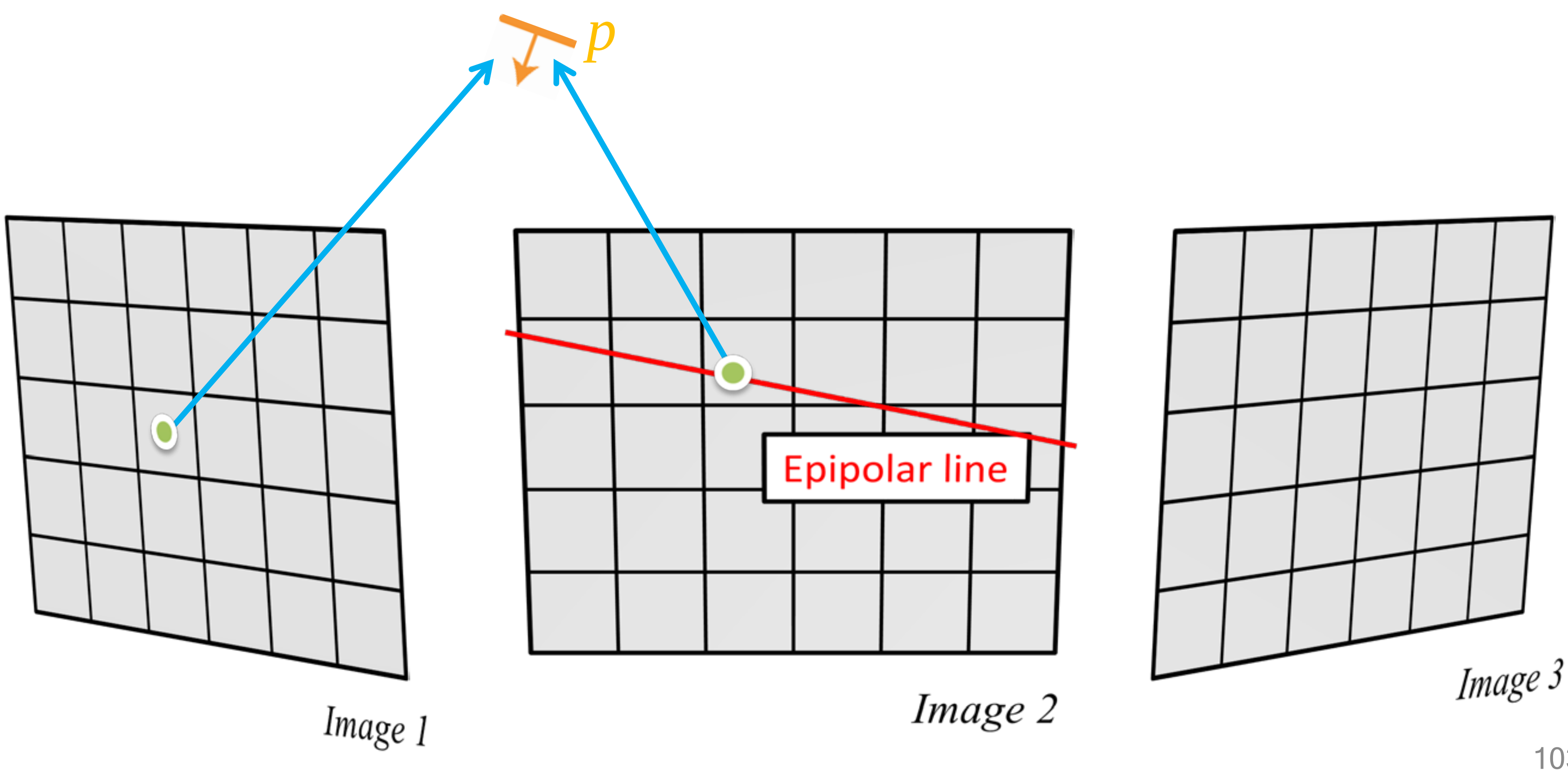
Initial feature matching

$c(p)$: triangulation
$n(p)$:
$V(p)$:



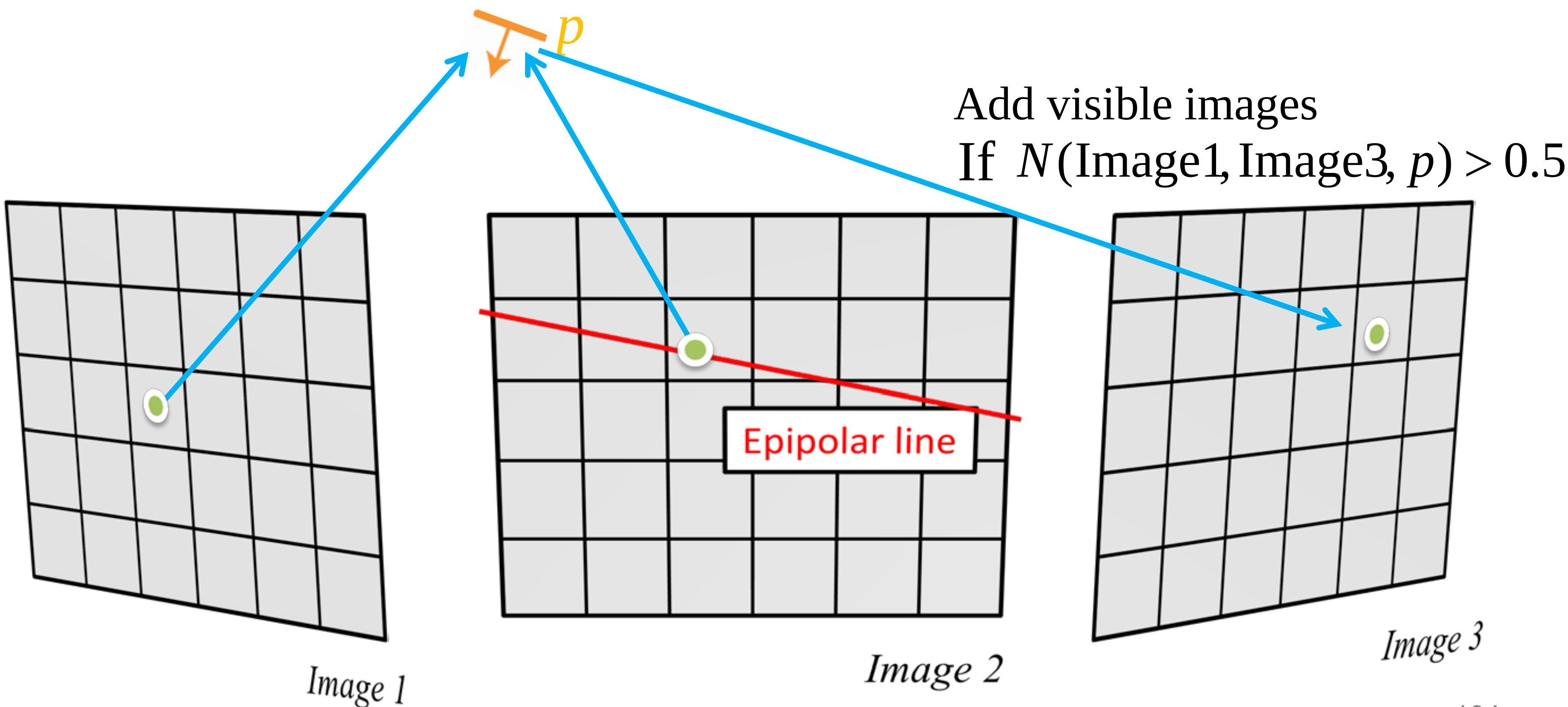
Initial feature matching

$c(p)$: triangulation
$n(p)$: parallel to *Image1*
$V(p)$: {*Image1*, *Image2*}



Initial feature matching

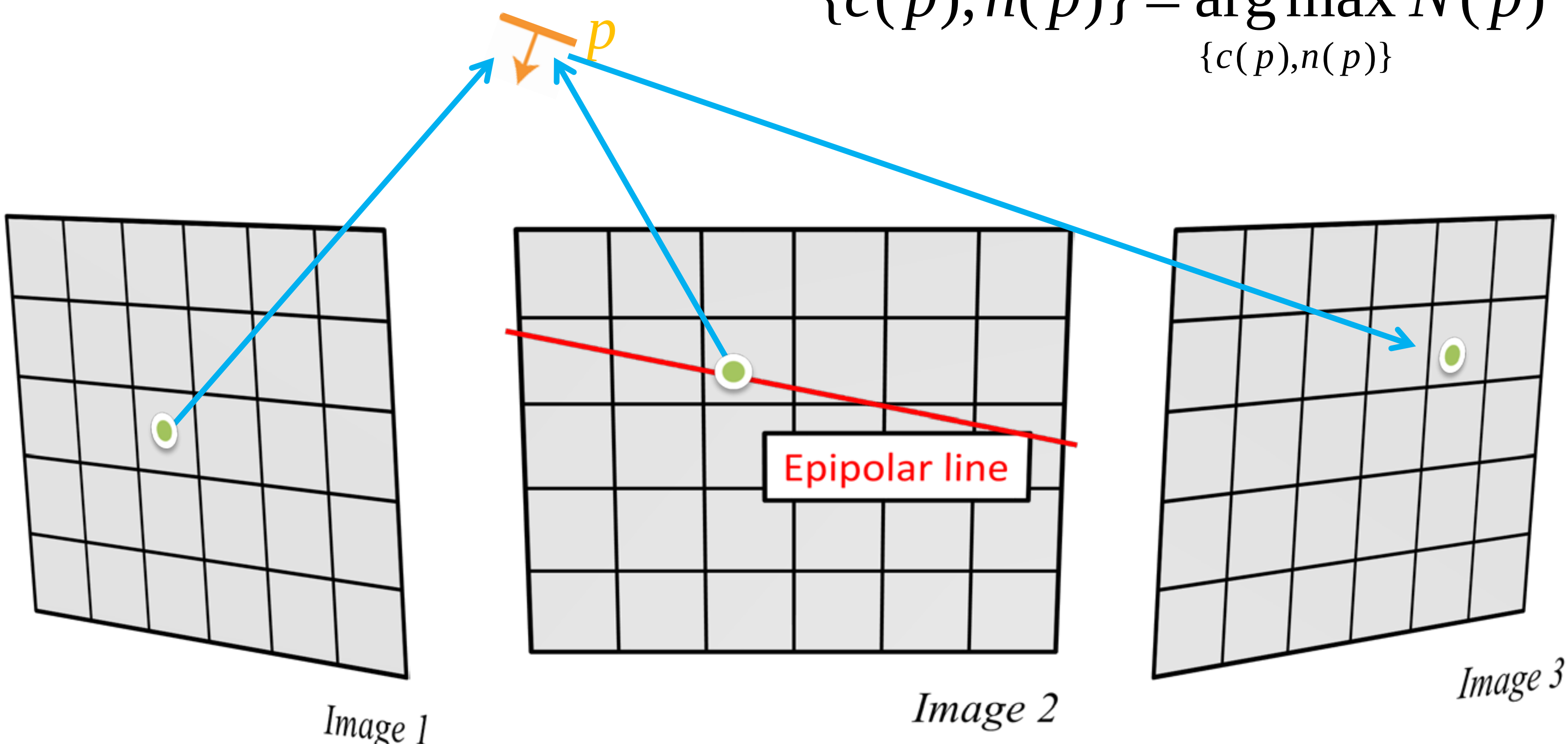
$c(p)$: triangulation
$n(p)$: parallel to *Image1*
$V(p)$: $\{$*Image1*, *Image2*$\}$*Image3*$\}$



Initial feature matching

$c(p)$: refine
$n(p)$: refine
$V(p)$: $\{Image1, Image2, Image3\}$

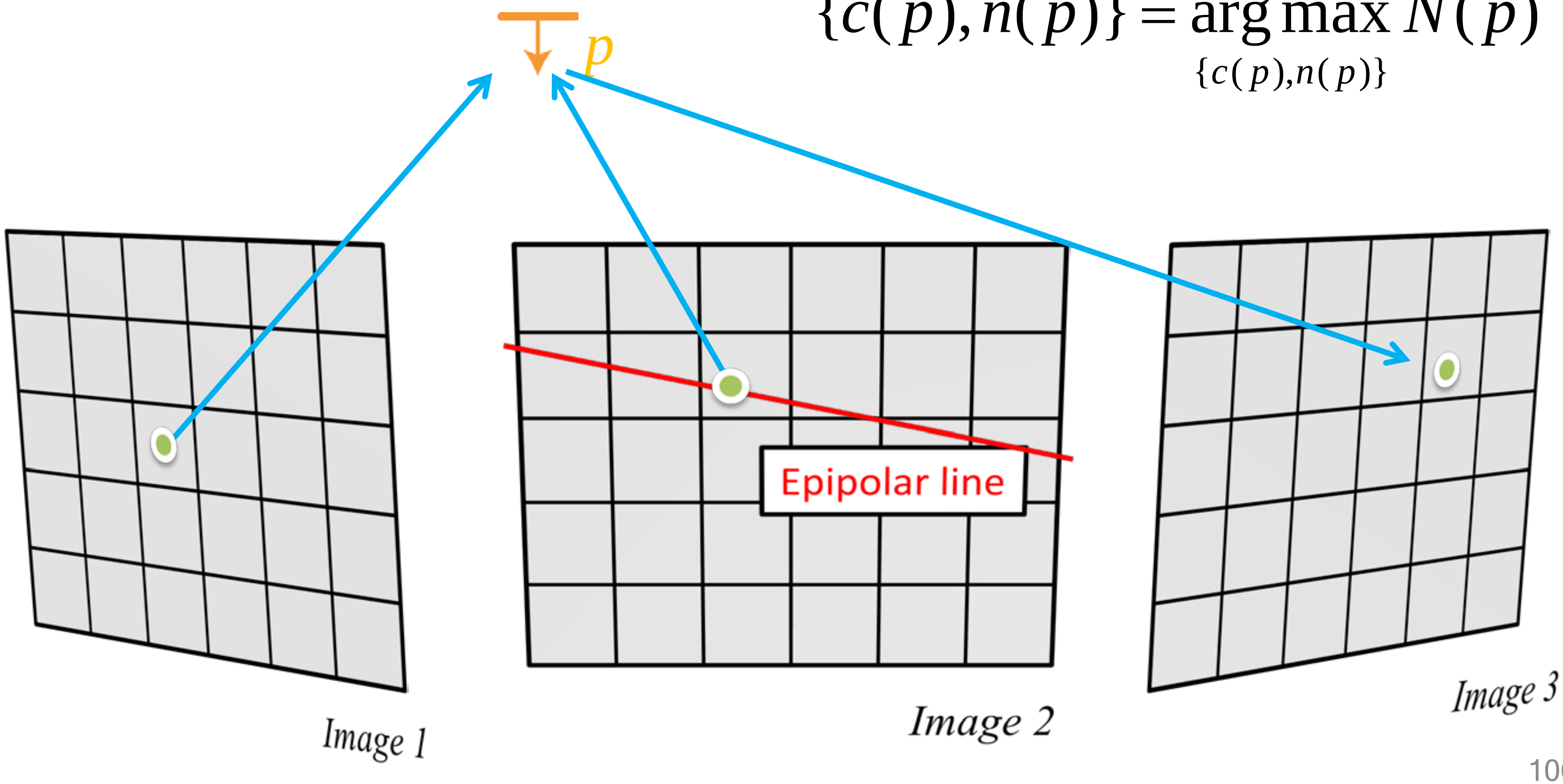
$$\{c(p), n(p)\} = \underset{\{c(p),n(p)\}}{\arg\max}\, N(p)$$



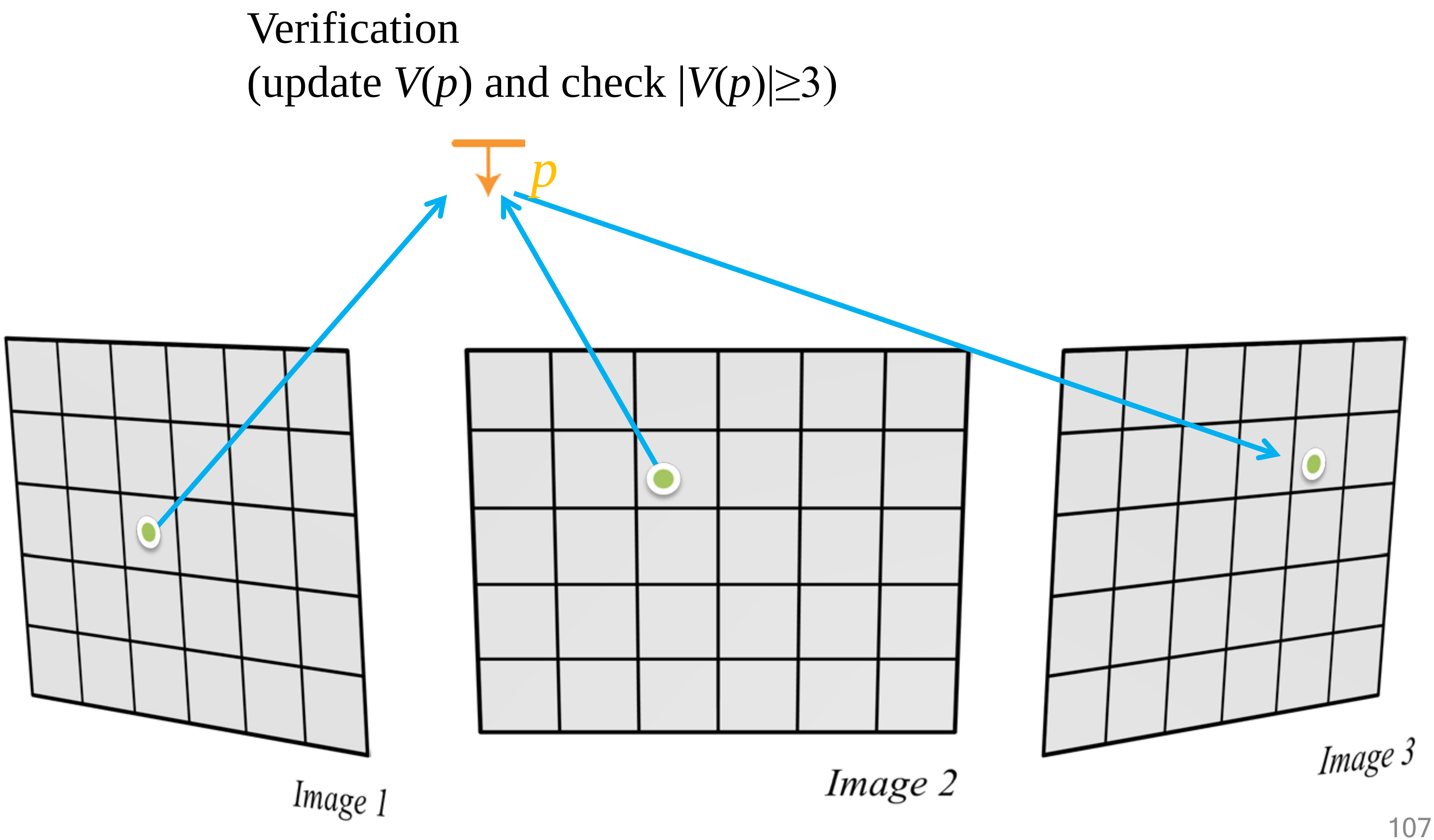
Initial feature matching

$c(p)$: refine
$n(p)$: refine
$V(p)$: $\{Image1, Image2, Image3\}$

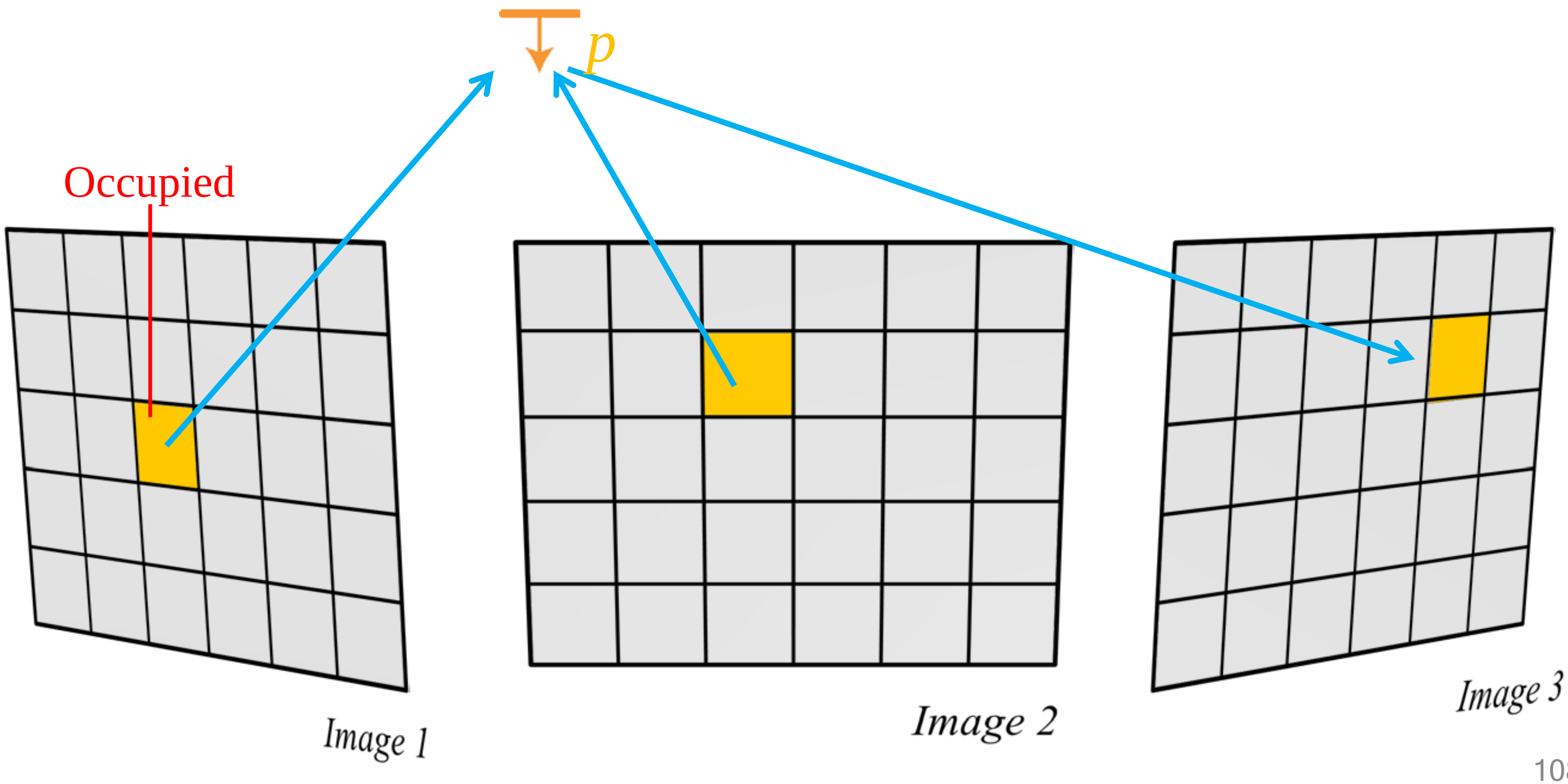
$$\{c(p), n(p)\} = \underset{\{c(p),n(p)\}}{\arg\max}\, N(p)$$



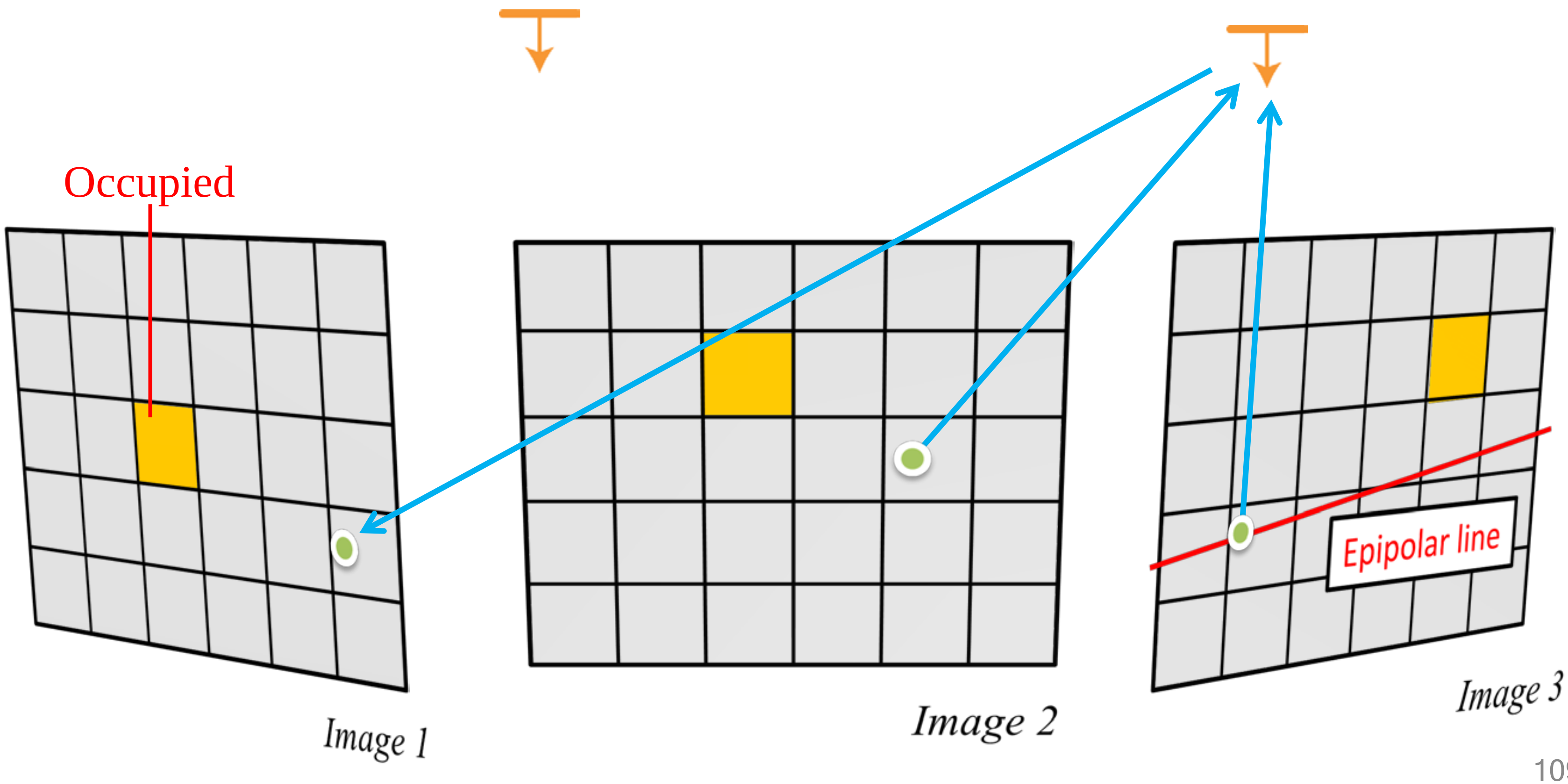
Initial feature matching



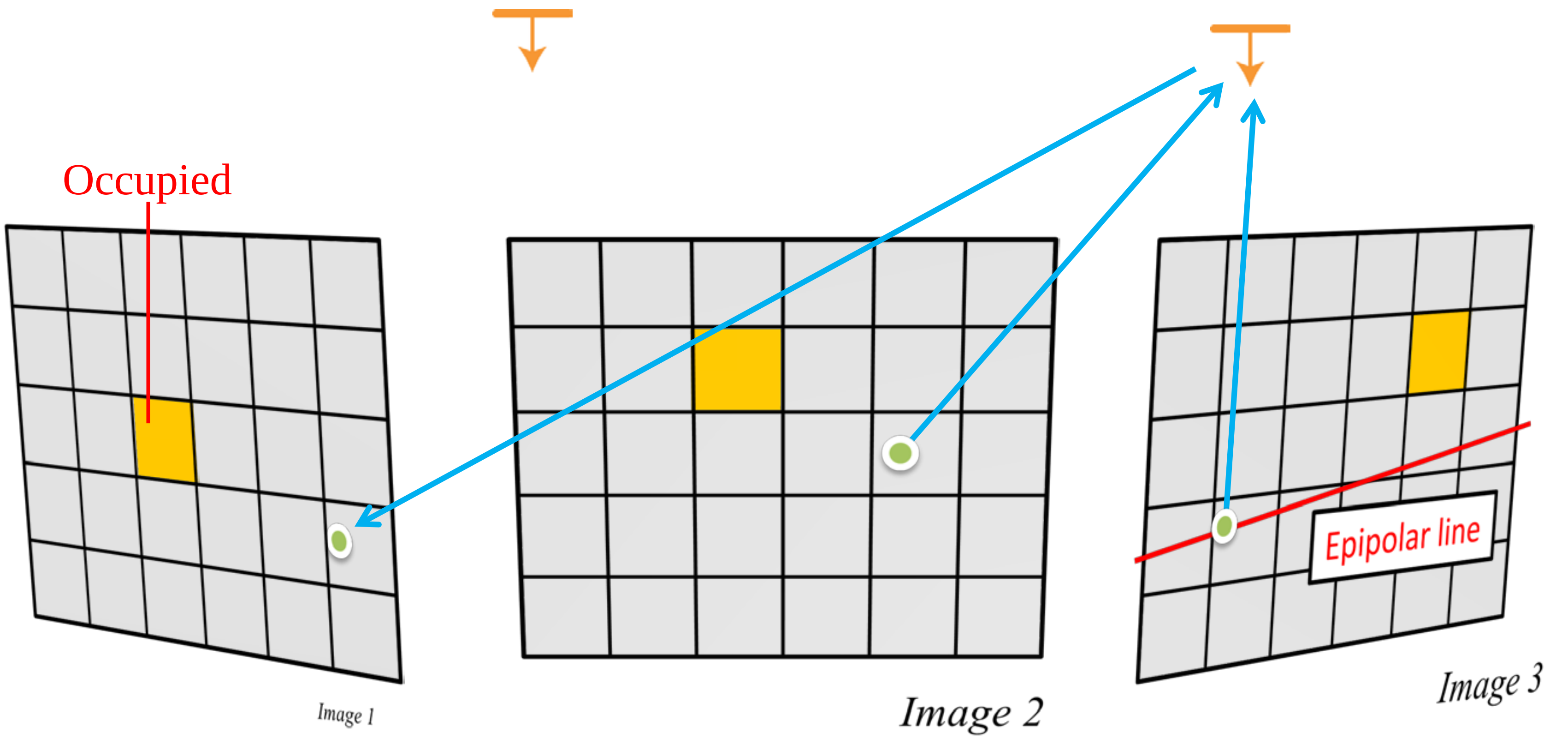
Initial feature matching



Initial feature matching

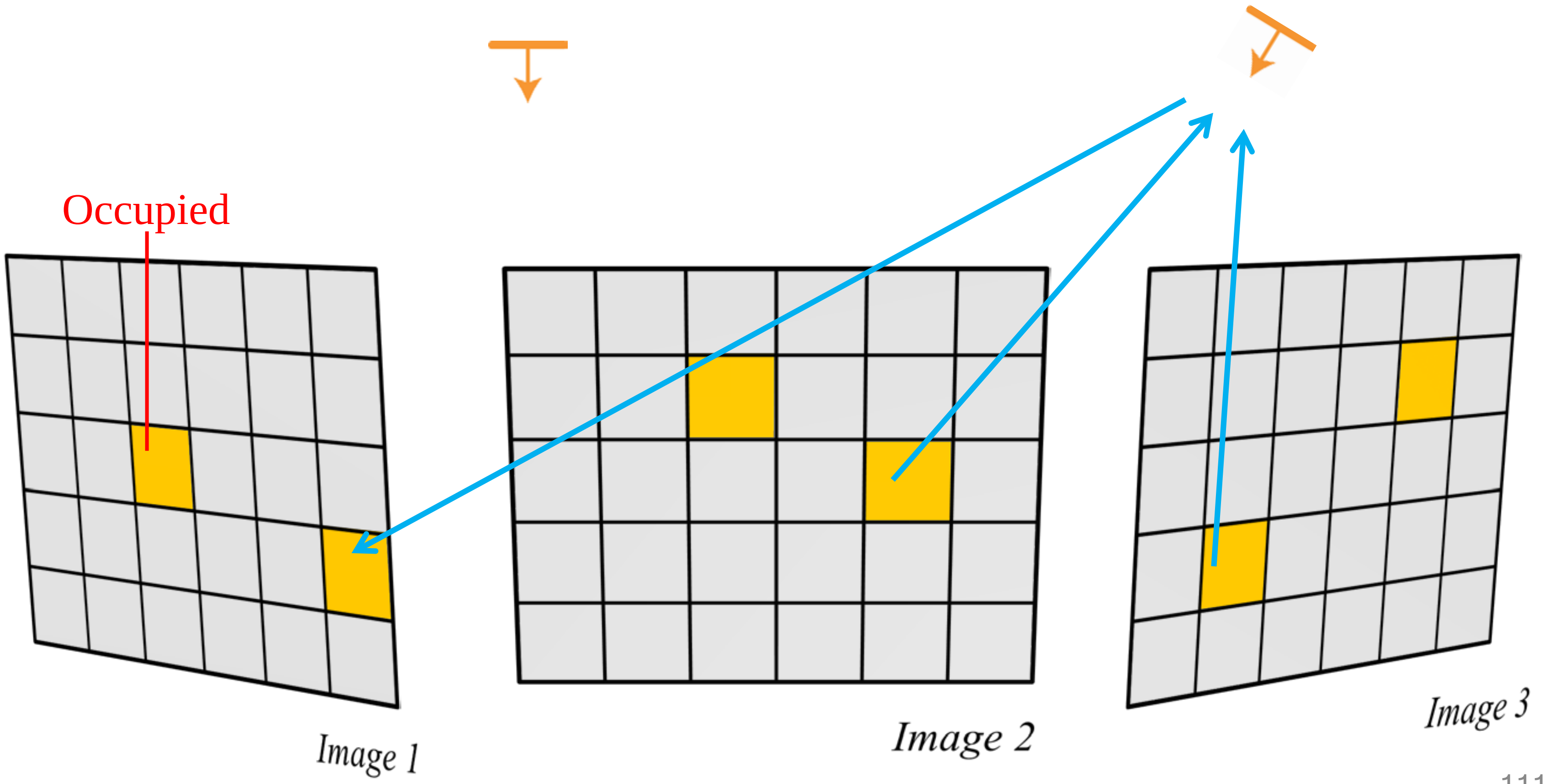


Initial feature matching



Initial feature matching

- Repeat for all image features

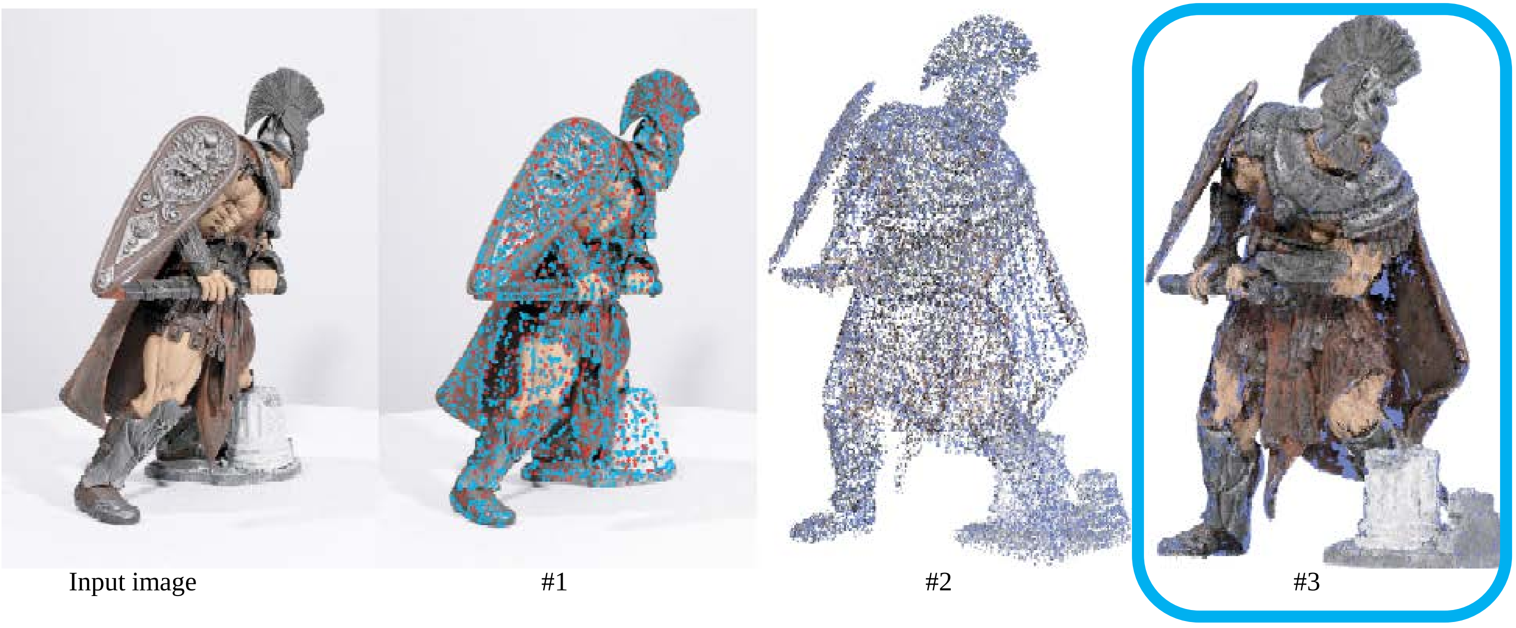


Algorithm Overview

#1. Feature detection

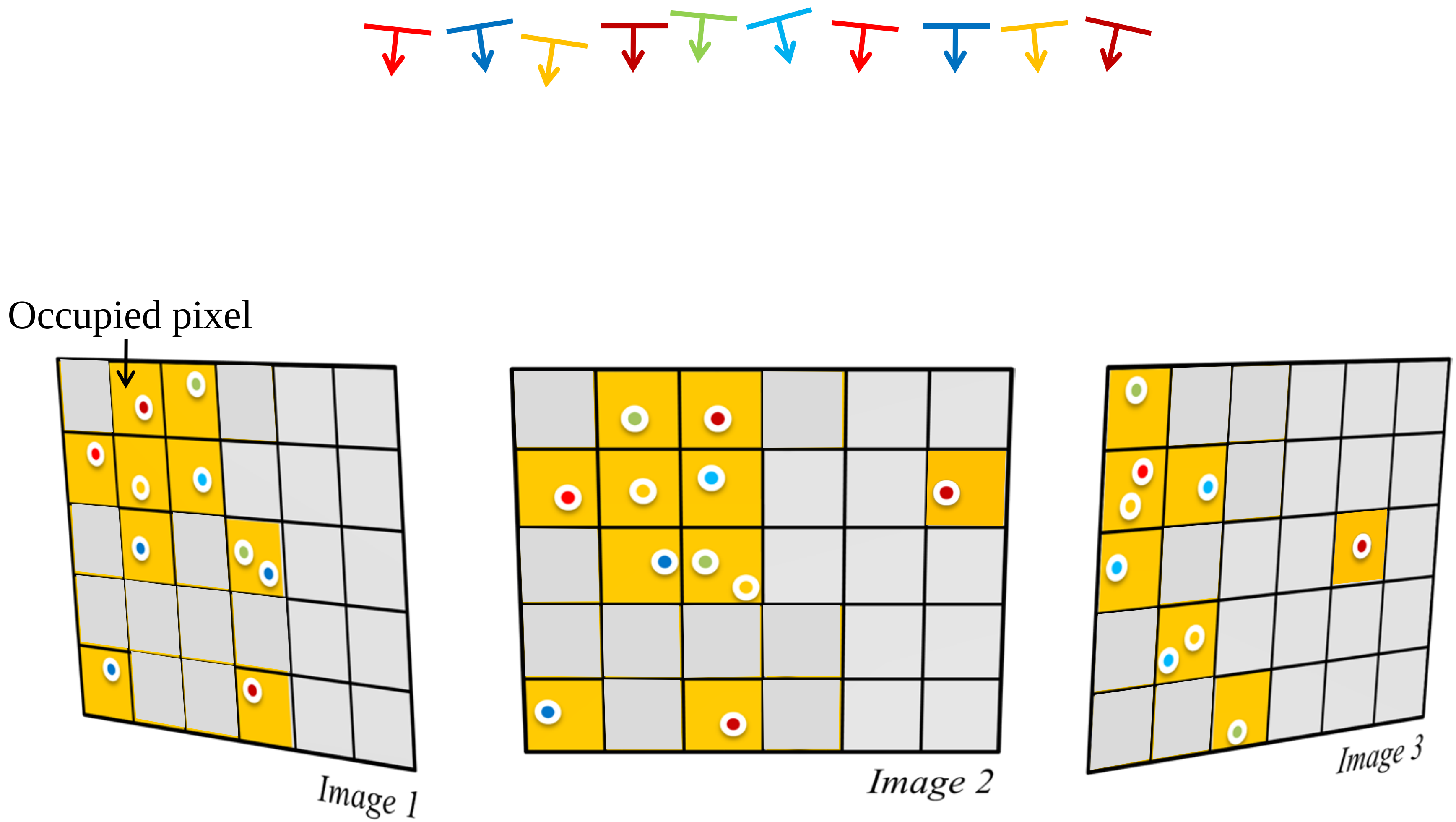
#2. Initial feature matching

#3. Patch expansion and filtering

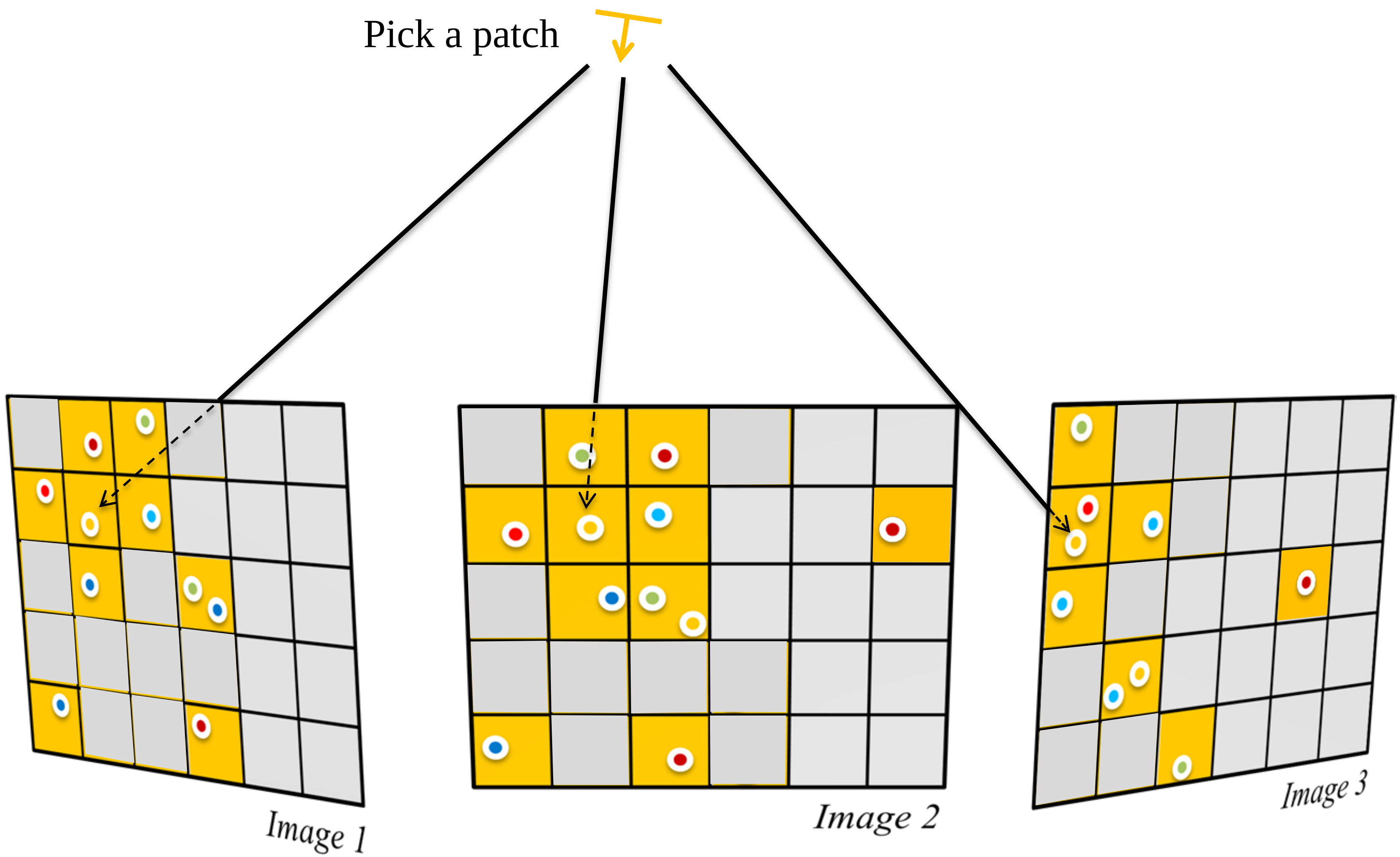


Input image #1 #2 #3

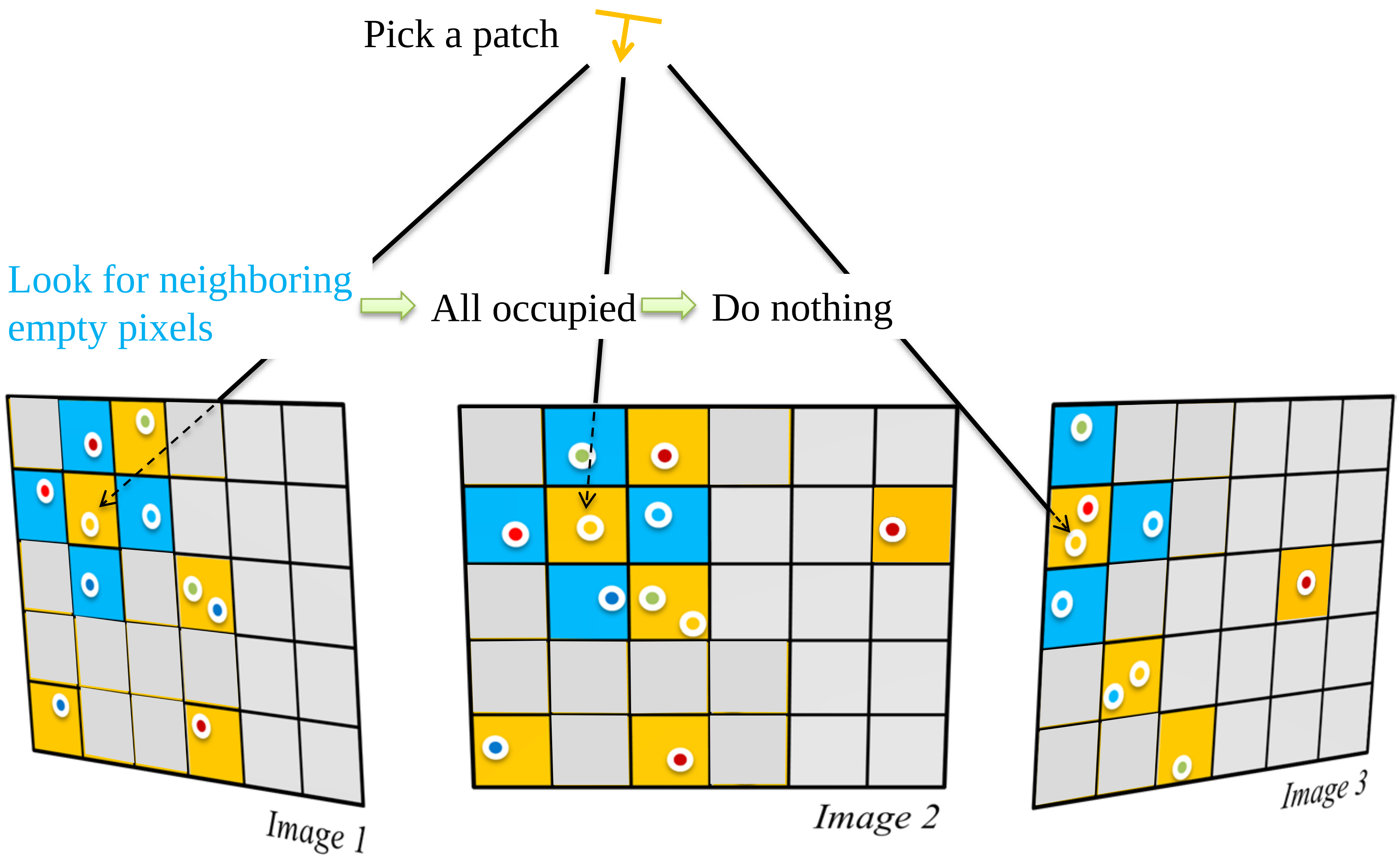
Patch expansion



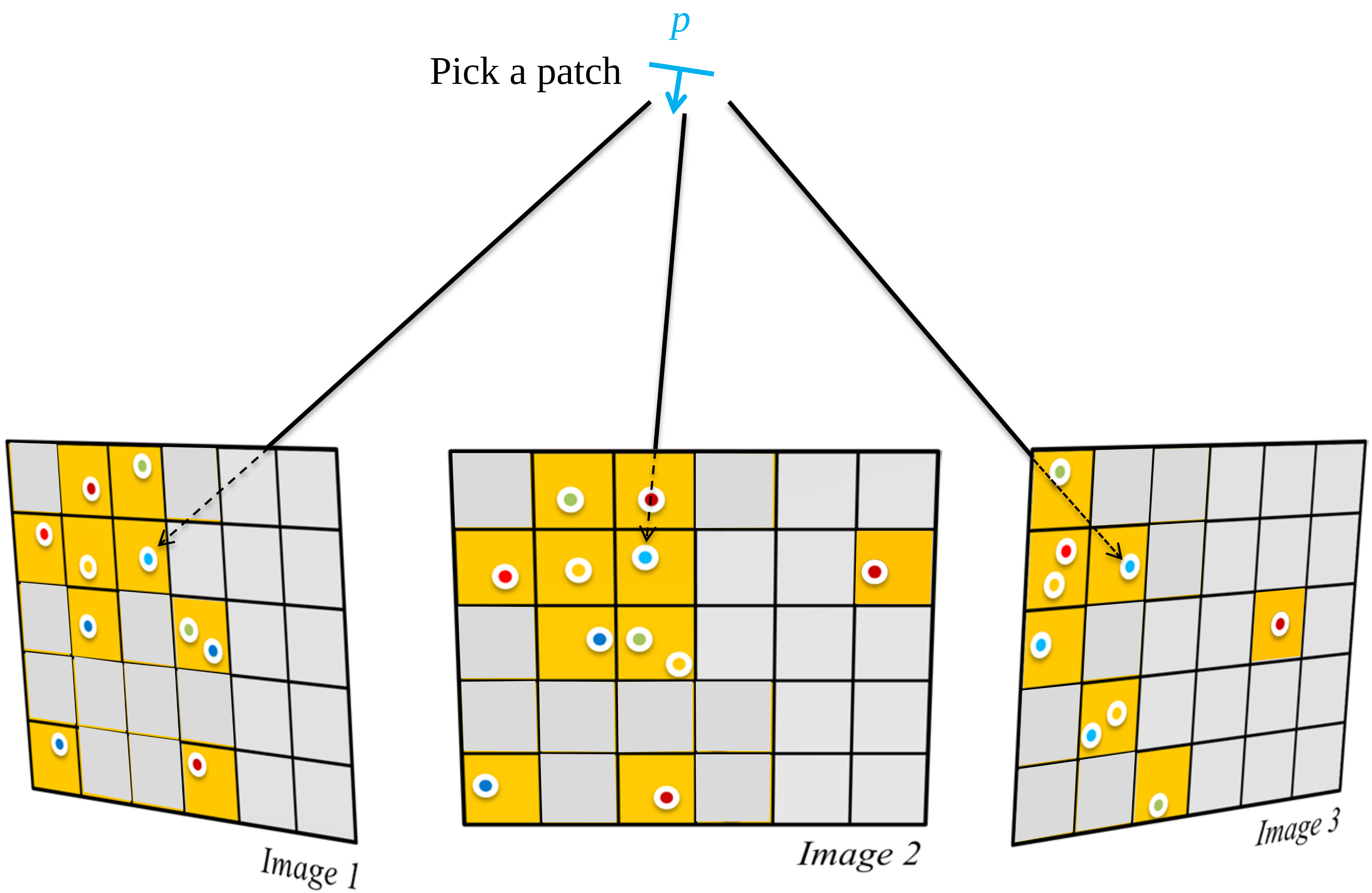
Patch expansion



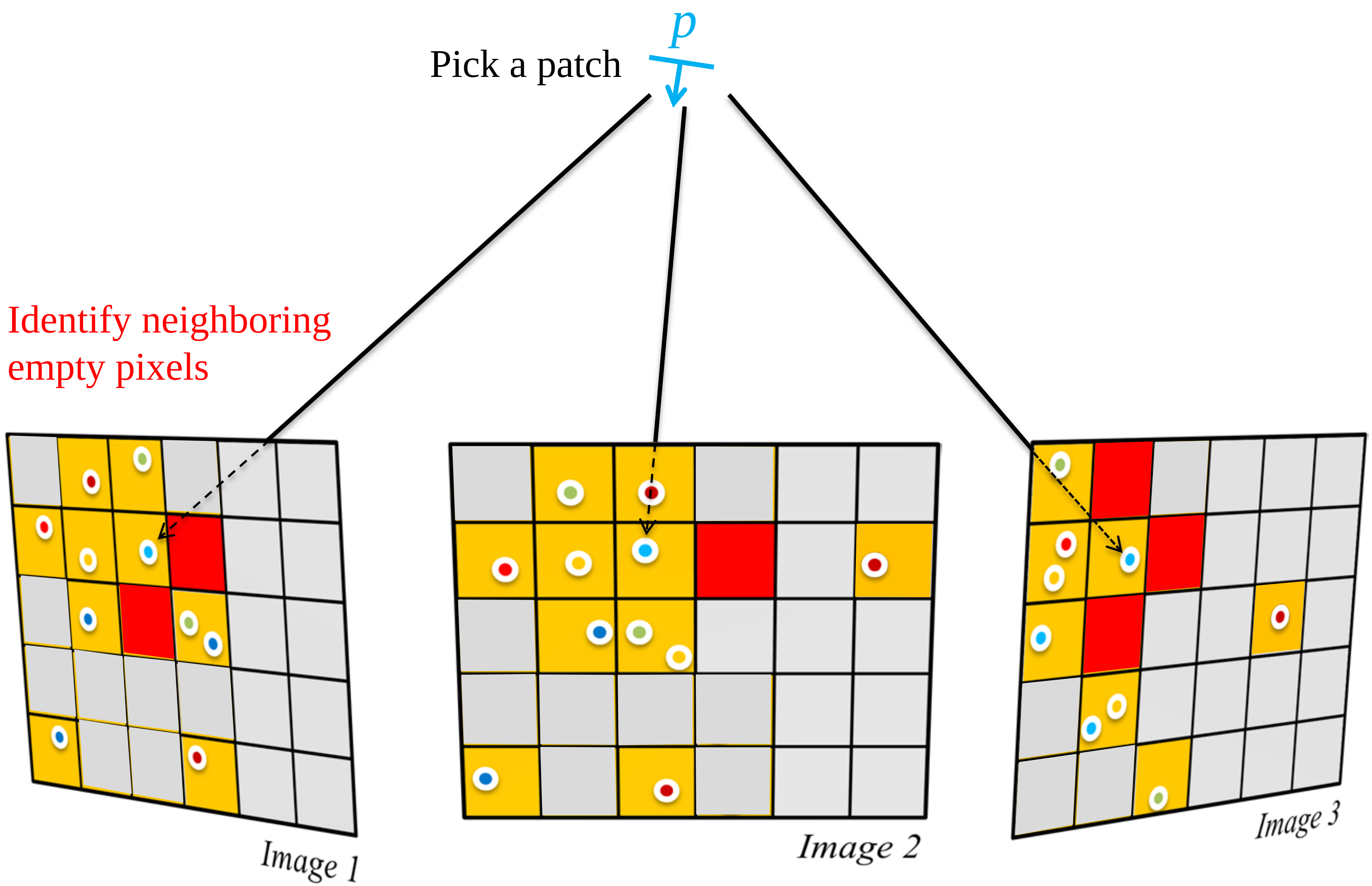
Patch expansion



Patch expansion

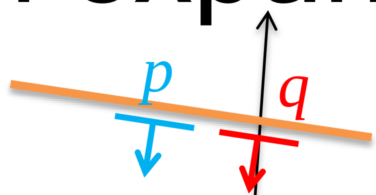


Patch expansion



Patch expansion

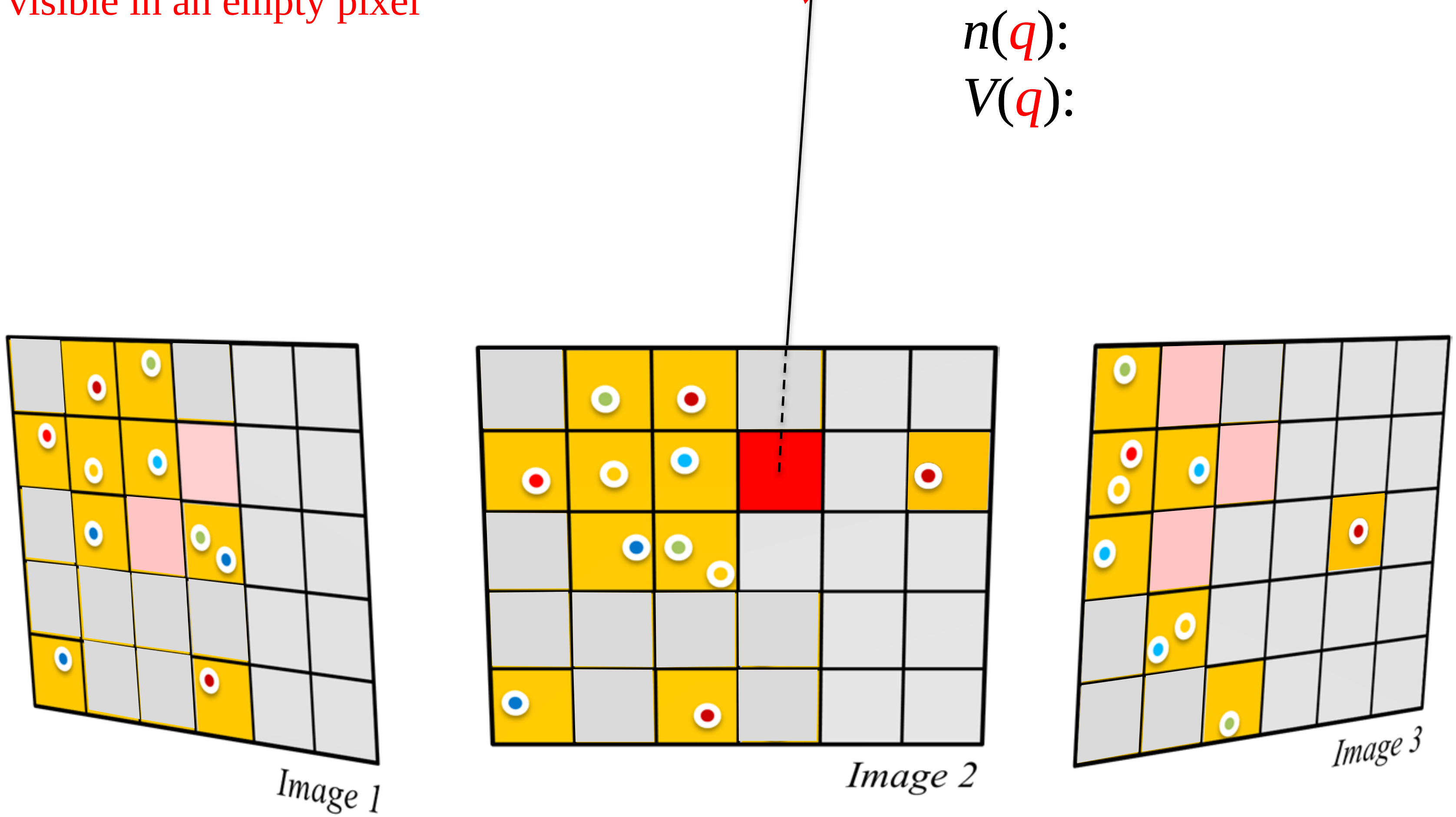
Reconstruct a patch visible in an empty pixel



$c(q)$: {tangent plane of p intersects w/ ray}

$n(q)$:

$V(q)$:



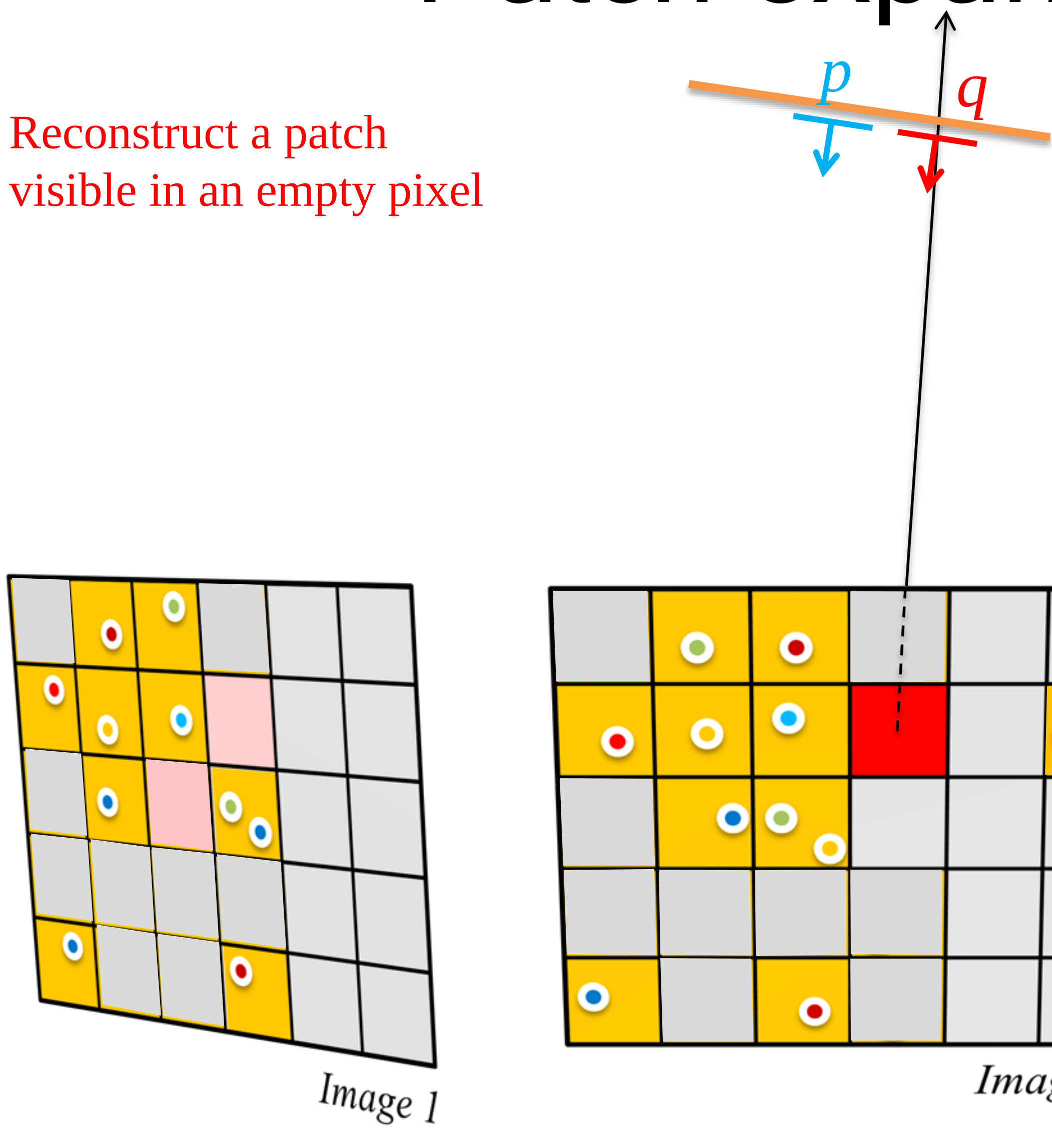
Patch expansion

Reconstruct a patch
visible in an empty pixel

$c(q)$: {tangent plane of p
intersects w/ ray}

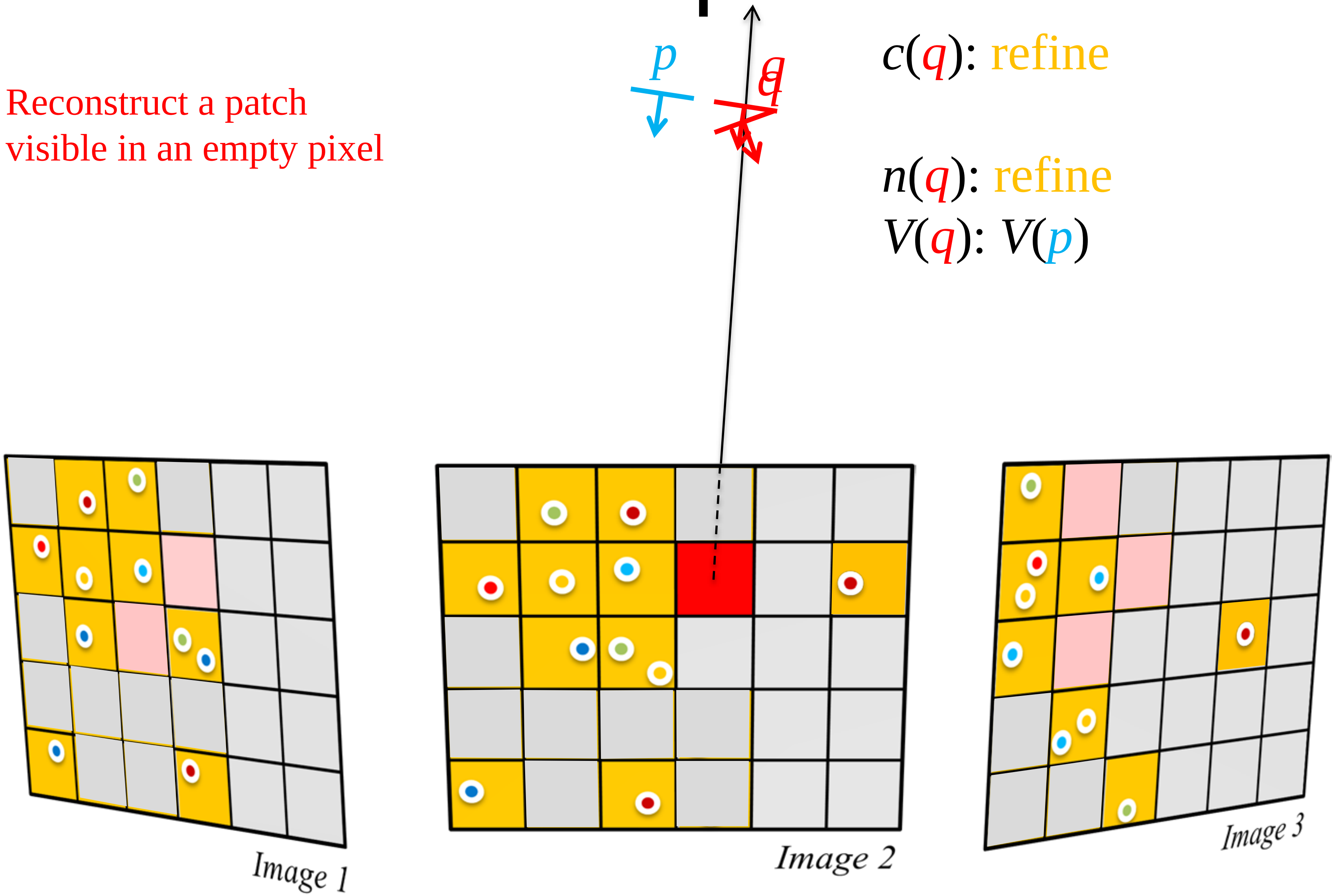
$n(q)$: $n(p)$

$V(q)$: $V(p)$



Patch expansion

Reconstruct a patch
visible in an empty pixel



$c(q)$: refine

$n(q)$: refine

$V(q)$: $V(p)$

Patch expansion

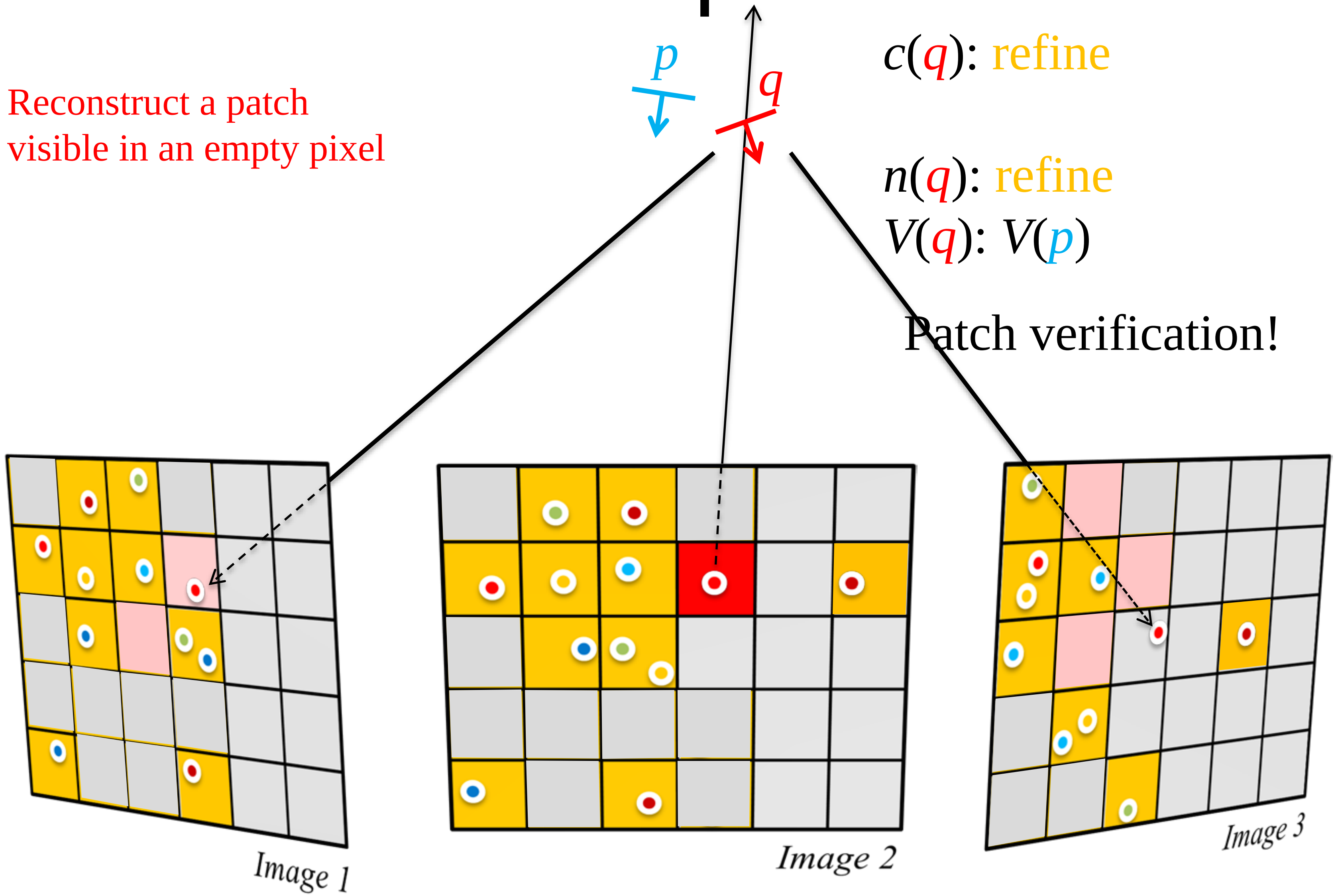
Reconstruct a patch visible in an empty pixel

$c(q)$: refine

$n(q)$: refine

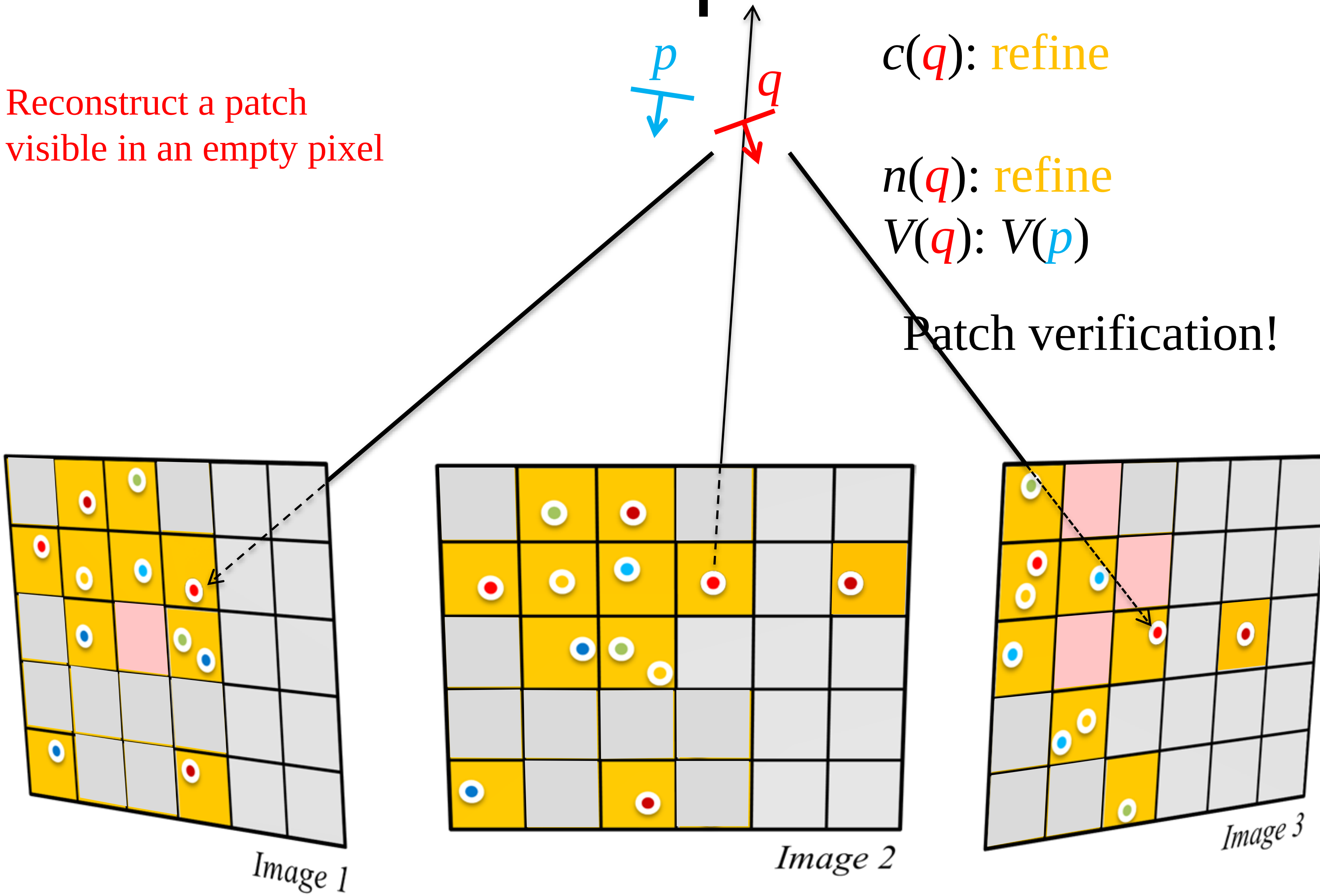
$V(q)$: $V(p)$

Patch verification!



Patch expansion

Reconstruct a patch
visible in an empty pixel

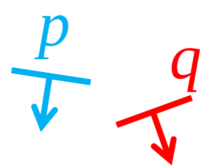


$c(q)$: refine

$n(q)$: refine
$V(q)$: $V(p)$

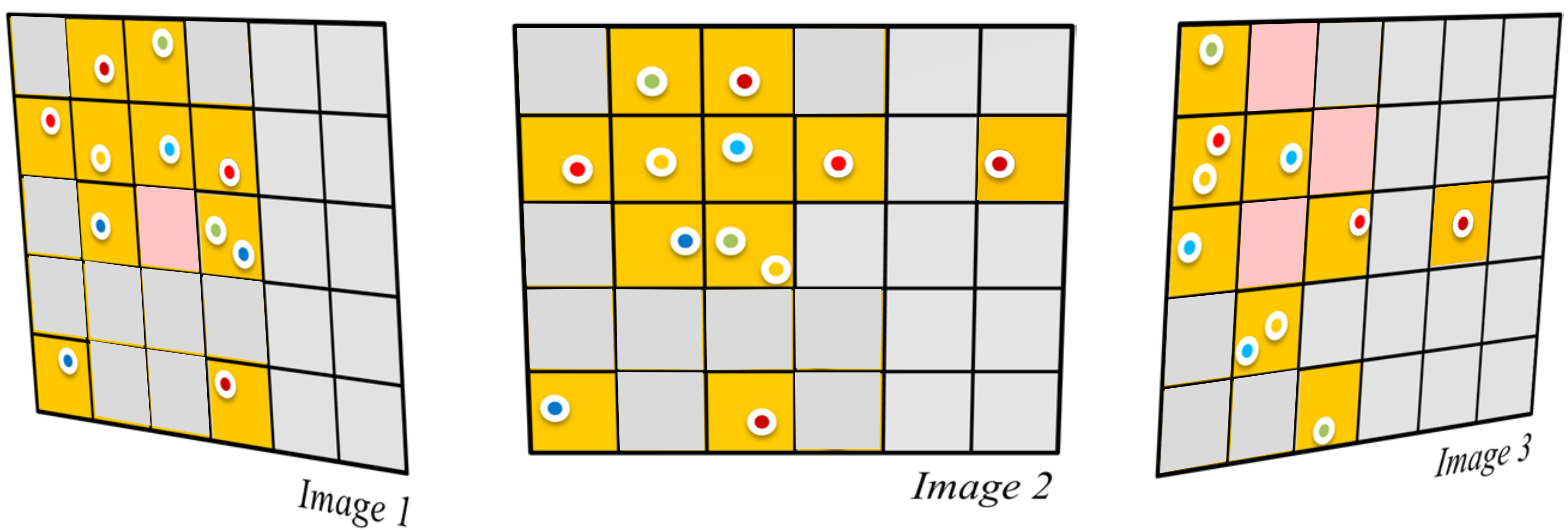
Patch verification!

Patch expansion



Repeat

- for every patch
- for every neighboring empty pixel



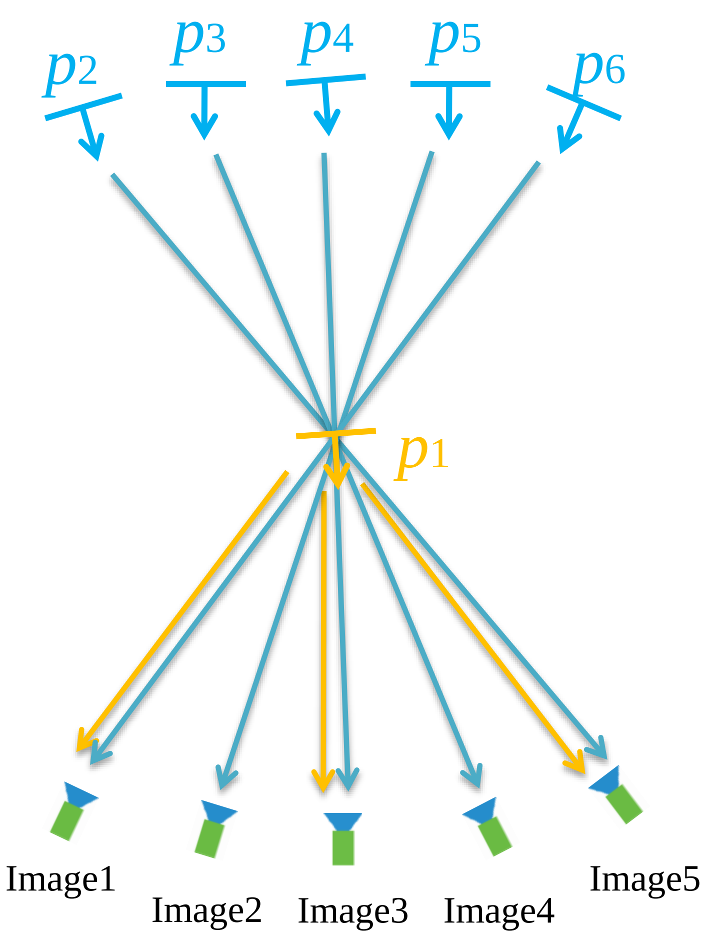
Patch filtering

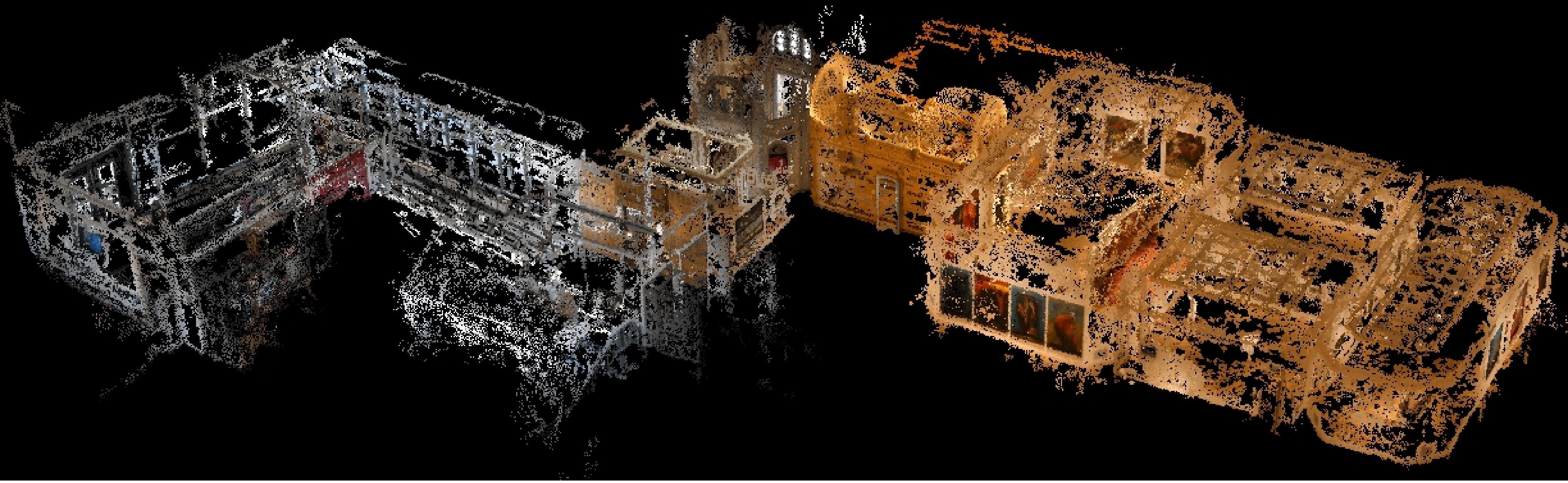
- Visibility consistency

Filter out p_1 if

$$|V(p_1)|N(p_1) < \sum_{i=2}^{6} N(p_i)$$

When p_1 is an outlier, both $V(p_1)$ and $N(p_1)$ are expected to be small





Limitations of MVS

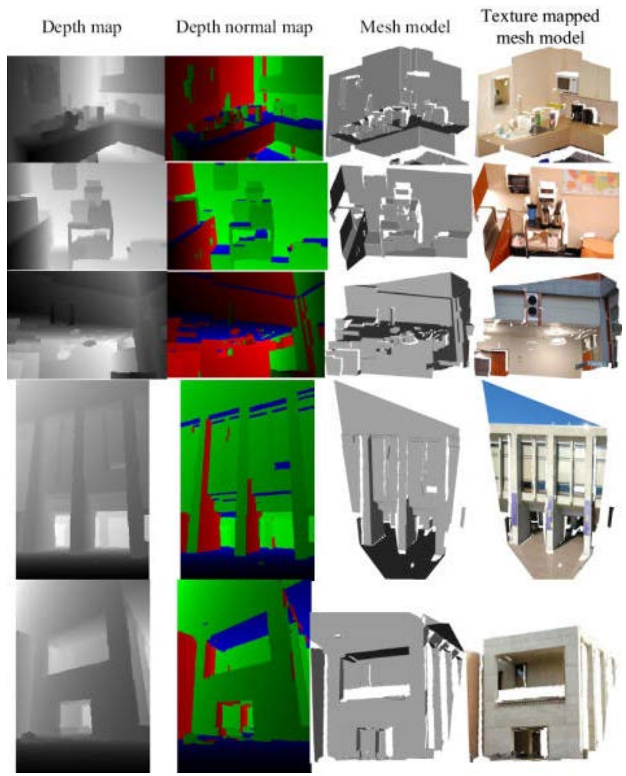
- Works well for various objects and scenes
- Surfaces must be Lambertian and well-textured

- Problematic for architectural scenes

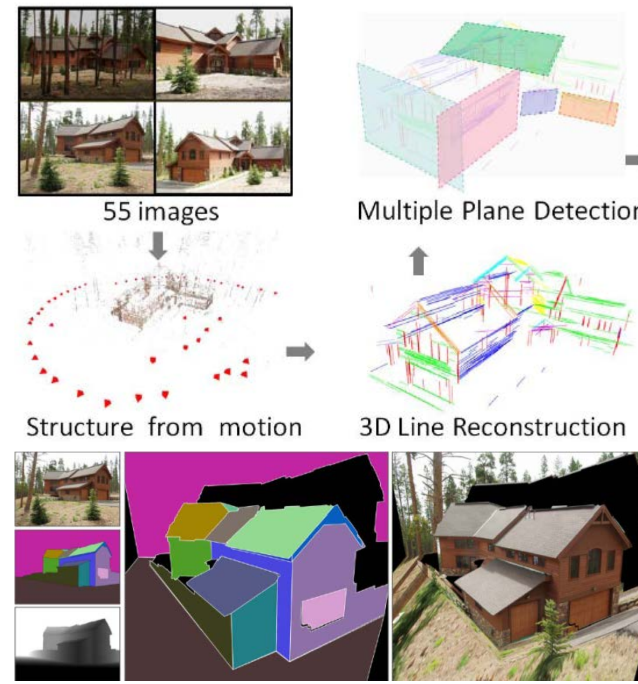
  

Recent Literature

[Zebedin et al., ECCV 2008]



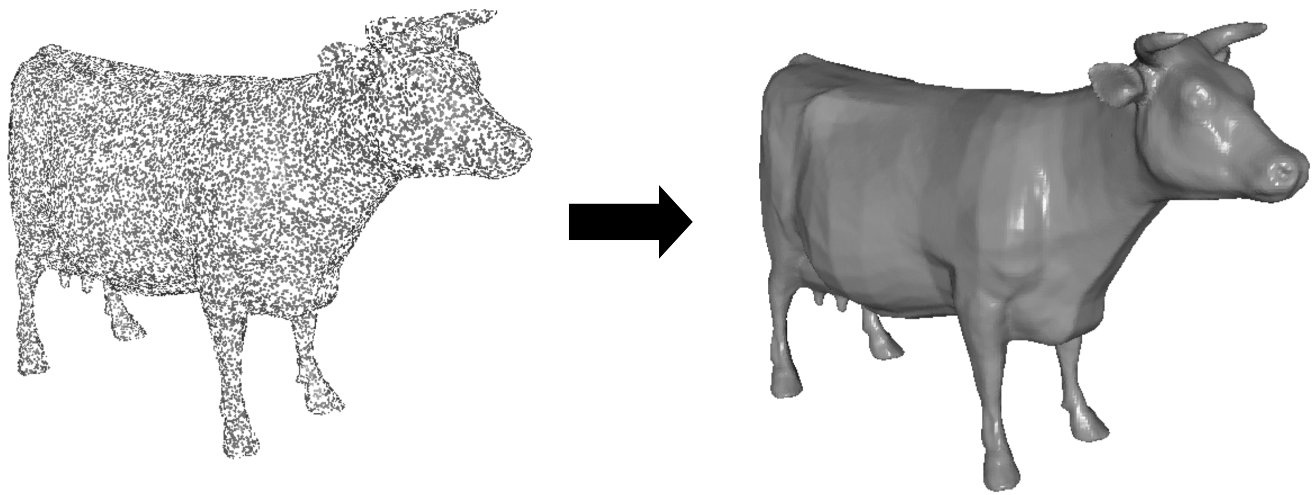
[Furukawa et al., CVPR 2009]



[Sinha et al., ICCV 2009]

Poisson Surface Reconstruction

- Input: points with oriented normals (pointing outwards)
- Output: dense, connected, triangle mesh



http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version8.0/