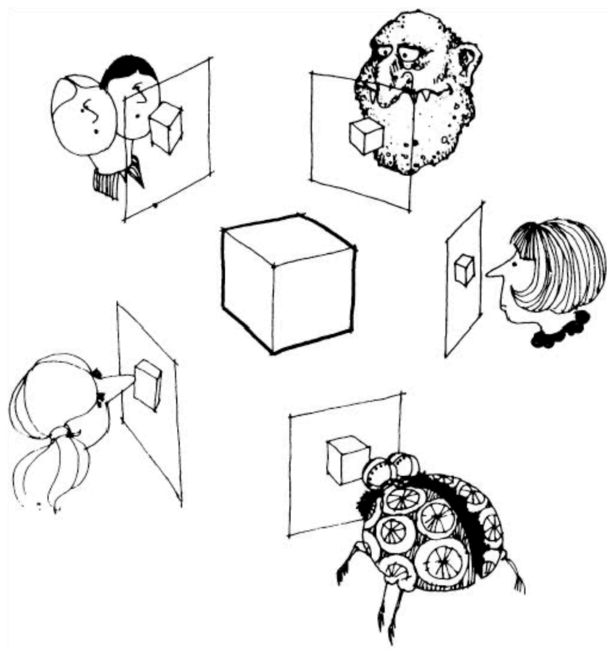
PatchMatch in Multi-View Stereo

Yiming Xie

2020.6.21



Many slides adapted from E. Dunn, S. Shen, Y. Furukawa, M. Pollefeys, and others

ETH 3D Benchmark(High Res.)

1	Method	all	high-res multi-view	indoor	outdoor
2	DeepPCF-MVS	80.84	88.10	88.56	86.73
3	DeepC-MVS_fast	79.62	86.82	86.47	87.85
4	DeepC-MVS	79.50	86.80	86.53	87.61
5	3Dnovator+	77.84	85.48	84.47	88.51
6	MG-MVS		83.41	83.45	83.28
7	3Dnovator	76.31	83.38	82.31	86.59
8	CLD-MVS		82.31	81.65	84.29
9	AP-MVS		82.00	81.11	84.69
10	MAR-MVS		81.84	80.70	85.27
11	ACMP	74.13	81.51	80.57	84.36
12	ACMM	73.20	80.78	79.84	83.58
13	AdaColmap		80.58	79.42	84.06
14	PCF-MVS	73.52	80.38	78.84	85.01
15	OpenMVS	72.83	79.77	78.33	84.09
16	TAPA-MVS	73.13	79.15	77.94	82.79
17	PLC	70.83	78.05	76.37	83.08
18	LS3D		76.95	74.82	83.37
19	F-COLMAP		76.38	74.33	82.50
20	LTVRE_ROB	69.57	76.25	74.54	81.41
21	ACMH+	68.96	76.01	74.01	82.03
22	ACMH	67.68	75.89	73.93	81.77
23	MSDG		73.36	70.99	80.49
24	COLMAP_ROB	66.92	73.01	70.41	80.81
25	OpenMVS_ROB	64.09	70.56	68.19	77.65
26	CMPMVS	51.72	70.19	68.16	76.28
27	LF4IMVS		64.02	62.19	69.50
28	PVSNet	57.27	61.67	59.27	68.85
29	FPMVS		53.68	51.64	59.81
30	ANet_high		50.57	46.10	63.99
31	Gipuma		45.18	41.86	55.16
32	PMVS	37.38	44.16	40.28	55.82
33	wuykxyi23d		39.76	33.18	59.51
34	PNet_i23d		38.26	35.42	46.79
35	MVE	26.22	30.37	25.89	43.81
36	wuyk23d		16.82	15.94	19.46
37	DeepMVS_CX				27.40
38	ITE_SJBPF				78.22

PatchMatch

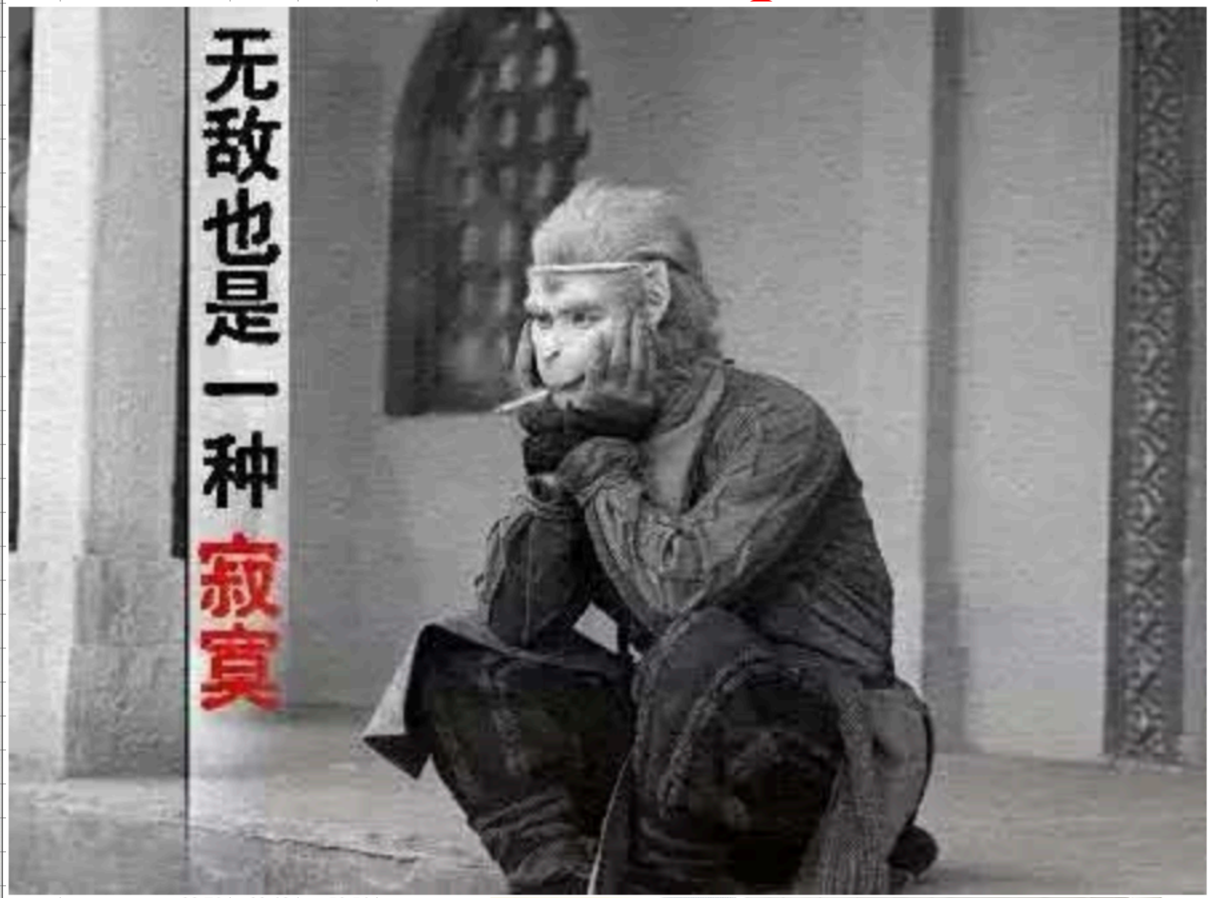
Unknown

Others

2020.6.20

ETH 3D Benchmark(High Res.)

	Method	all	high-res multi-view	indoor	outdoor
2	DeepPCF-MVS	80.84	88.10	88.56	86.73
3	DeepC-MVS_fast	79.62	86.82	86.47	87.85
4	DeepC-MVS	79.50	86.80	86.53	87.61
5	3Dnovator+	77.84	85.48	84.47	88.51
6	MG-MVS		83.41	83.45	83.28
7	3Dnovator				
8	CLD-MVS				
9	AP-MVS				
10	MAR-MVS				
11	ACMP				
12	ACMM				
13	AdaColmap				
14	PCF-MVS				
15	OpenMVS				
16	TAPA-MVS				
17	PLC				
18	LS3D				
19	F-COLMAP				
20	LTVRE_ROB				
21	ACMH+				
22	ACMH				
23	MSDG				
24	COLMAP_ROB				
25	OpenMVS_ROB				
26	CMPMVS				
27	LF4IMVS				
28	PVSNet				
29	FPMVS				
30	ANet_high				
31	Gipuma				
32	PMVS				
33	wuykxyi23d		39.76	33.18	59.51
34	PNet_i23d		38.26	35.42	46.79
35	MVE	26.22	30.37	25.89	43.81
36	wuyk23d		16.82	15.94	19.46
37	DeepMVS_CX				27.40
38	ITE_SJBPF				78.22



Topics

- Introduction
- PatchMatch
- PatchMatch Stereo
- View Selection

Introduction

Why Does it Matter?

UAV

Robotics

Augmented Reality

- **Goal: Sensing 3D Geometry**

Why Does it Matter?

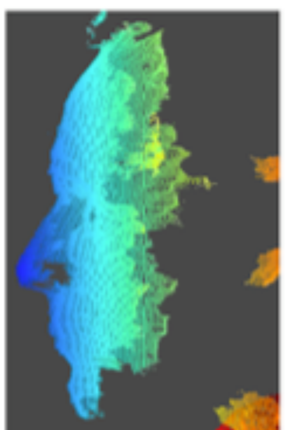
UAV

Robotics

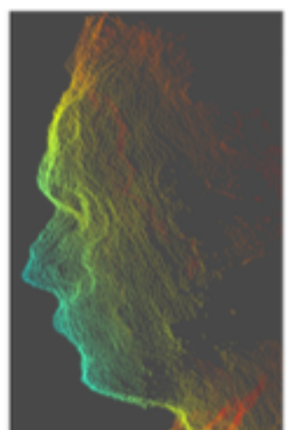
Augmented Reality

- **Goal: Sensing 3D Geometry**

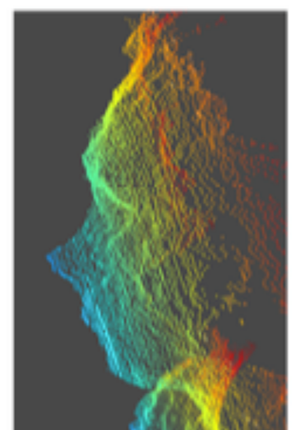
Depth Sensor



300mm



500mm



700mm

2500mm

Why Does it Matter?

UAV

Robotics

Augmented Reality

- **Goal: Sensing 3D Geometry**

- Among all, image-based methods provide a fast way of capturing accurate 3D content at a fraction of the cost of other approaches.

What is MVS

- Multi-view stereo (MVS): use stereo correspondence as their main cue and use more than two images to extract geometry from photographs.

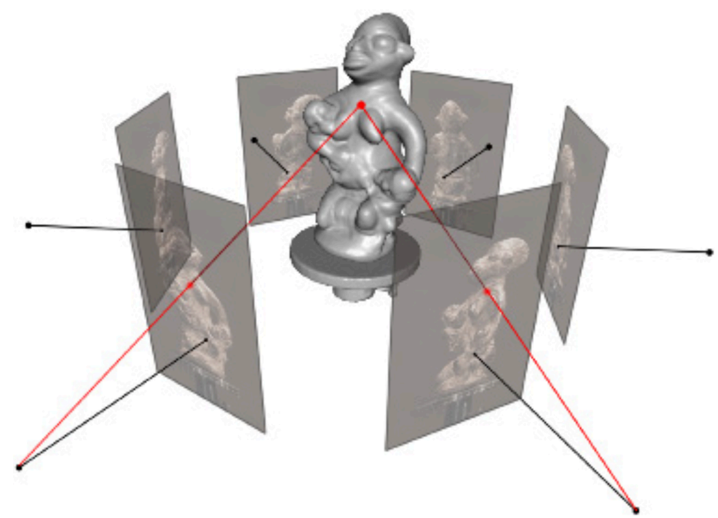
What is MVS

- Multi-view stereo (MVS): use stereo correspondence as their main cue and use more than two images to extract geometry from photographs.
    - Lambertian textured surfaces.
    - Known camera parameters.

What is MVS

- Multi-view stereo (MVS): use stereo correspondence as their main cue and use more than two images to extract geometry from photographs.
  - Lambertian textured surfaces.
  - Known camera parameters.

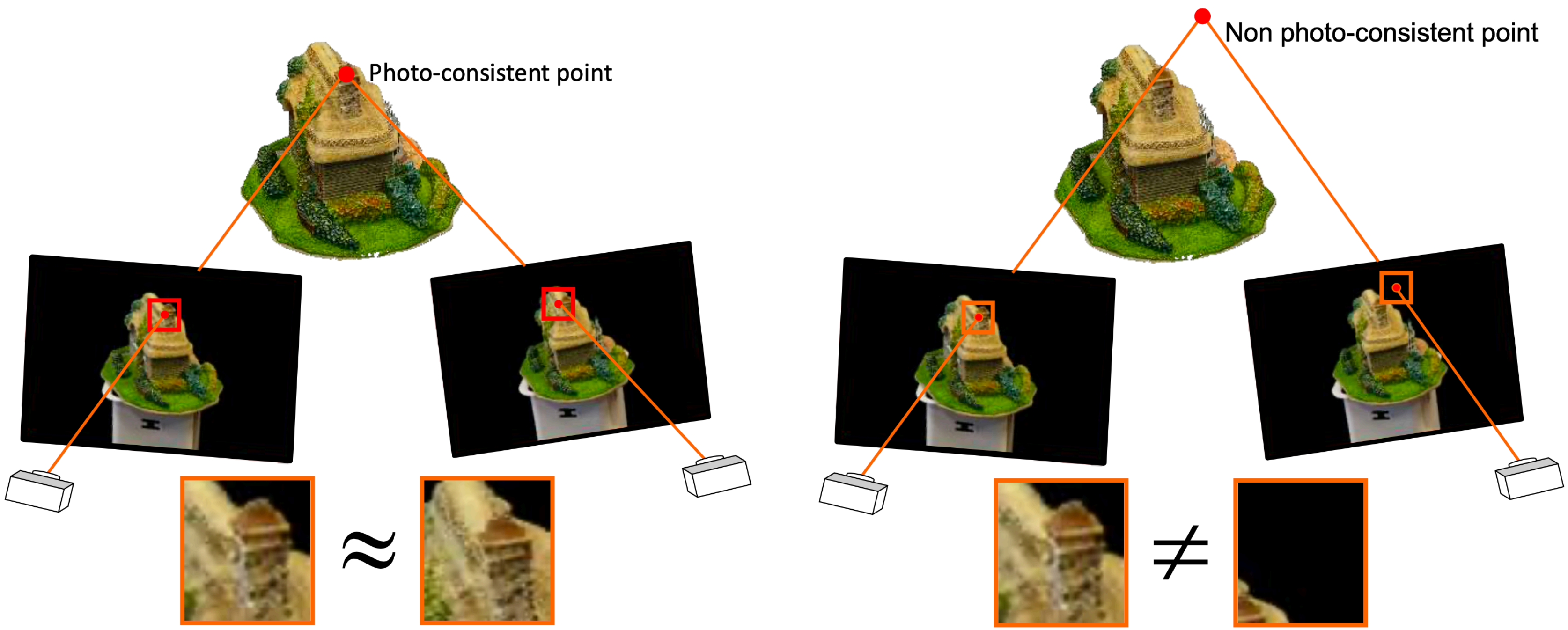
- Input: multiple images with calibrated cameras
- Output: dense 3d representation



Credit: Y. Furukawa

Multi-view stereo: Basic idea

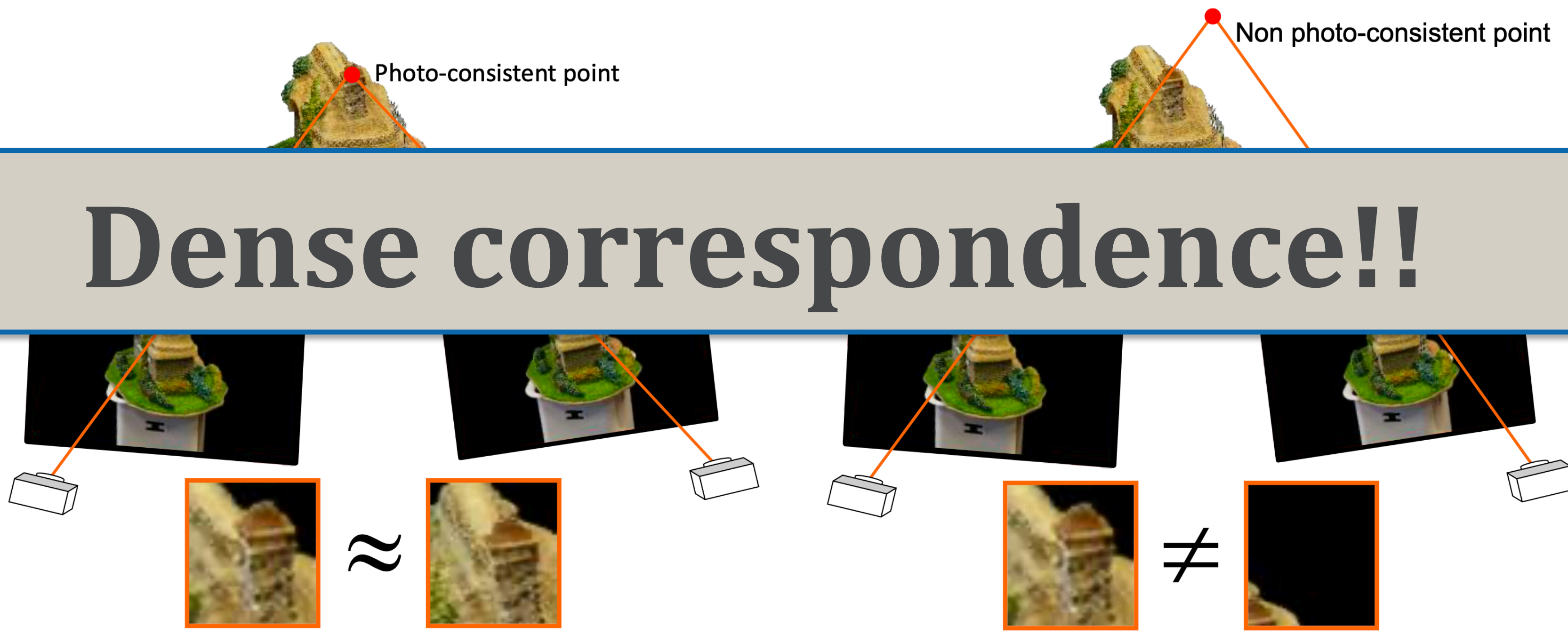
- Look for points in space that have photo-consistency.



Credit: Shuhan Shen

Multi-view stereo: Basic idea

- Look for points in space that have photo-consistency.



Dense correspondence!!

Credit: Shuhan Shen

Summary

- **Why?**
- capture accurate 3D geometry, and image-based method is cheap.

- **What?**
- use stereo correspondence as their main cue and use more than two images to extract geometry from photographs.

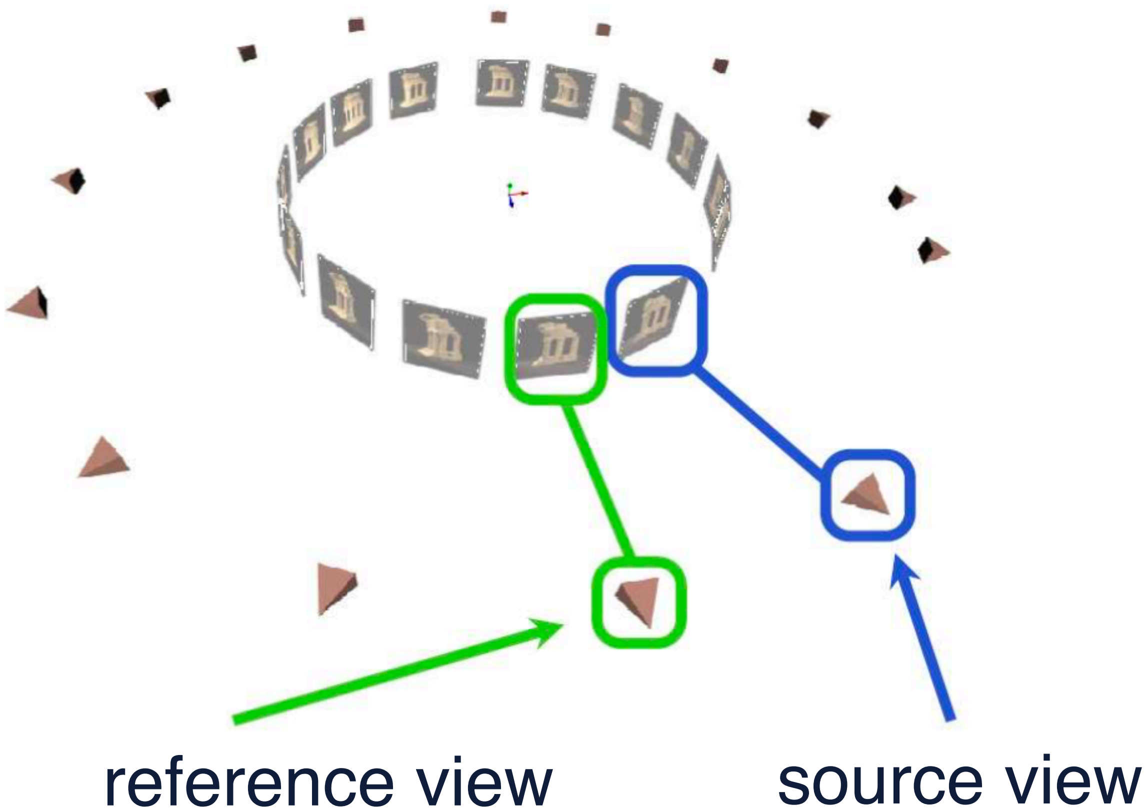
- **How?**
- Look for points in space that have photo-consistency.

Depth-map Merging Based Approaches

Depth-map Merging Based Approaches

Depth-map Merging Based Approaches

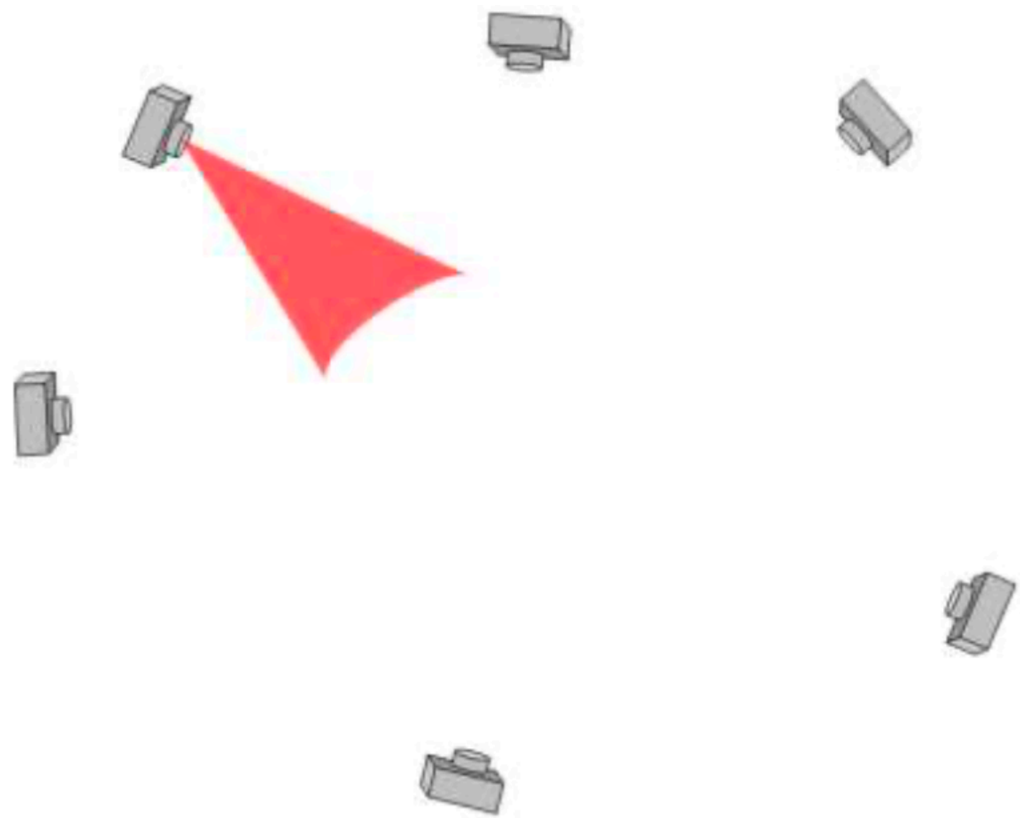
Step 1: Source view selection



Credit: Shuhan Shen

Depth-map Merging Based Approaches

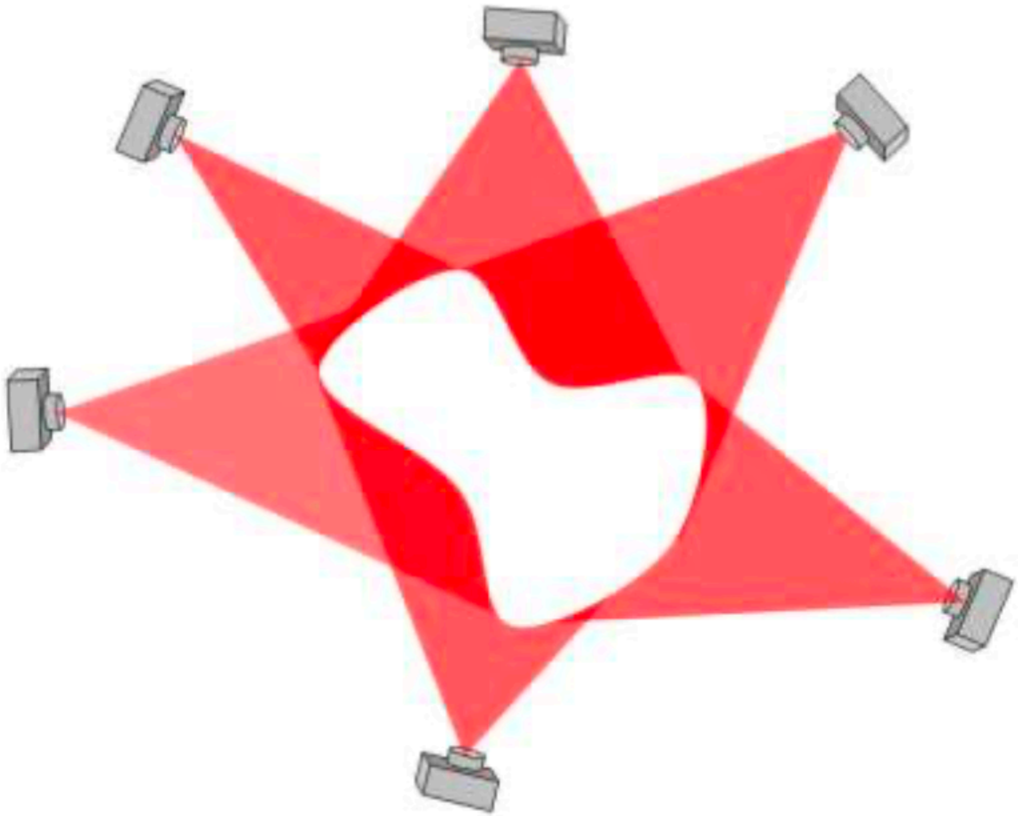
Step 2: Depth-map computation



Credit: Shuhan Shen

Depth-map Merging Based Approaches

Step 3: Depth-map merging



Credit: Shuhan Shen

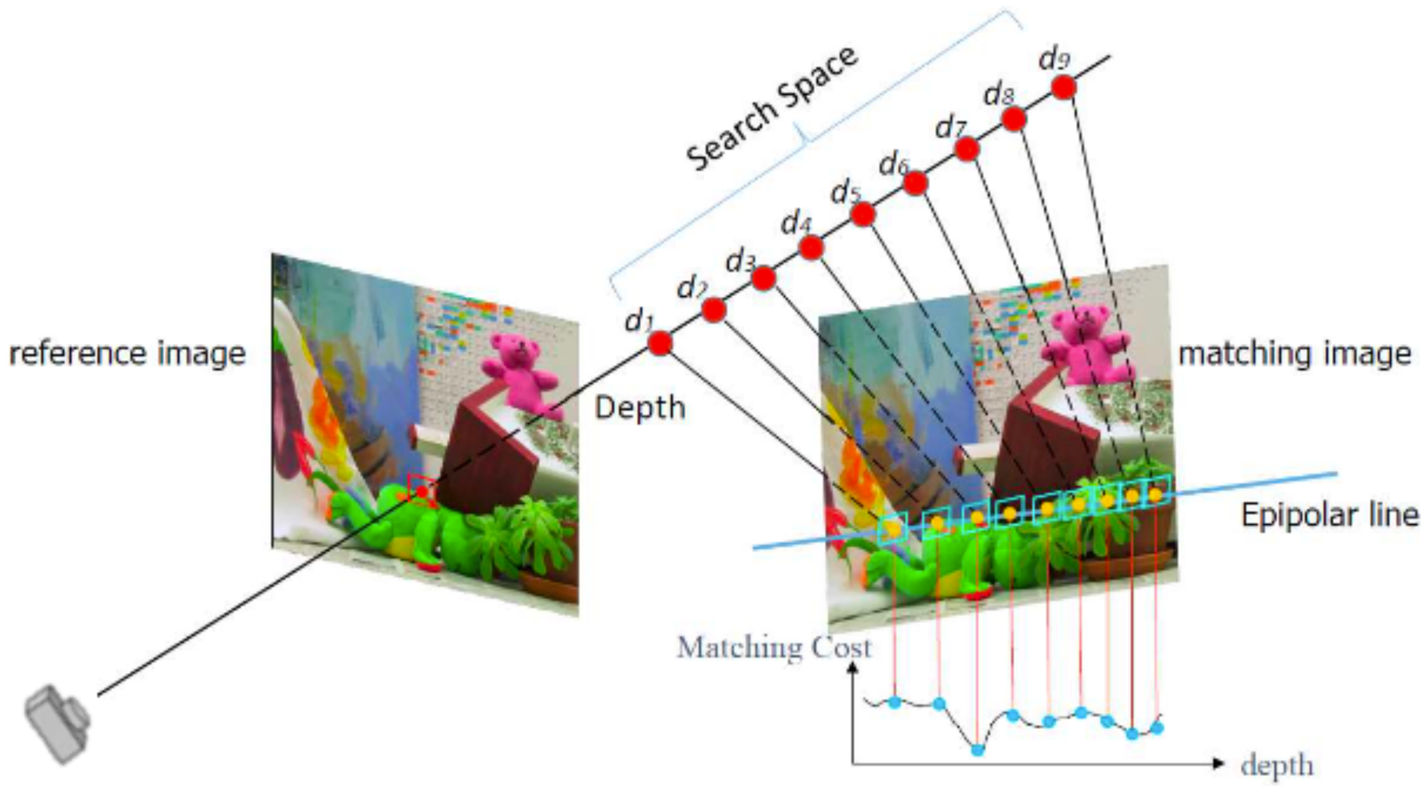
- Step 1: Source view selection
- Step 2: Depth-map computation
- Step 3: Depth-map merging

Key steps:
1.How to chose source images
2.How to compute depth map

How to compute depth map

Compute Depth Map: Basic idea

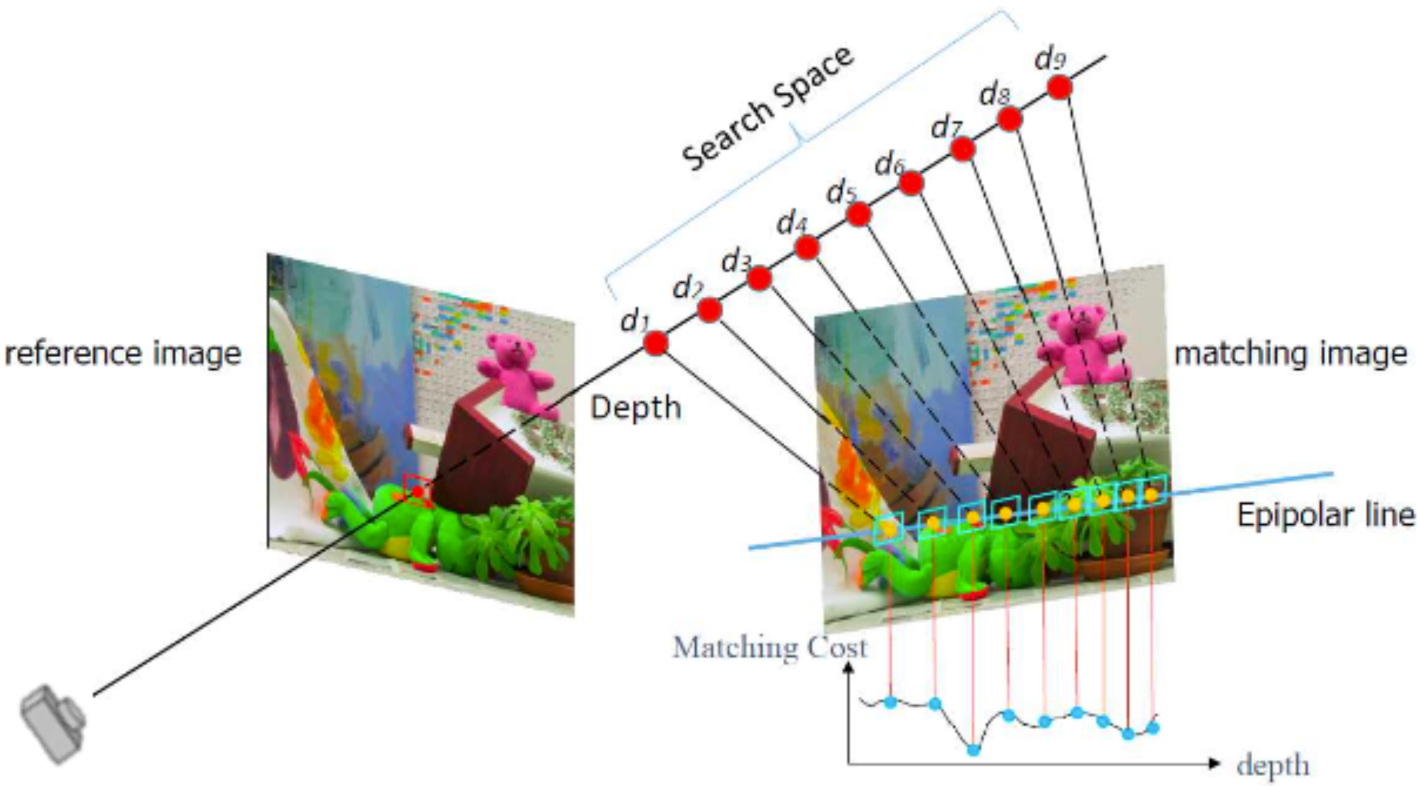
- Look for points in space that have photo-consistency.



Credit: E. Dunn

Compute Depth Map: Basic idea

- Look for points in space that have photo-consistency.



Credit: E. Dunn

✗ **limit on high resolution images**

PatchMatch

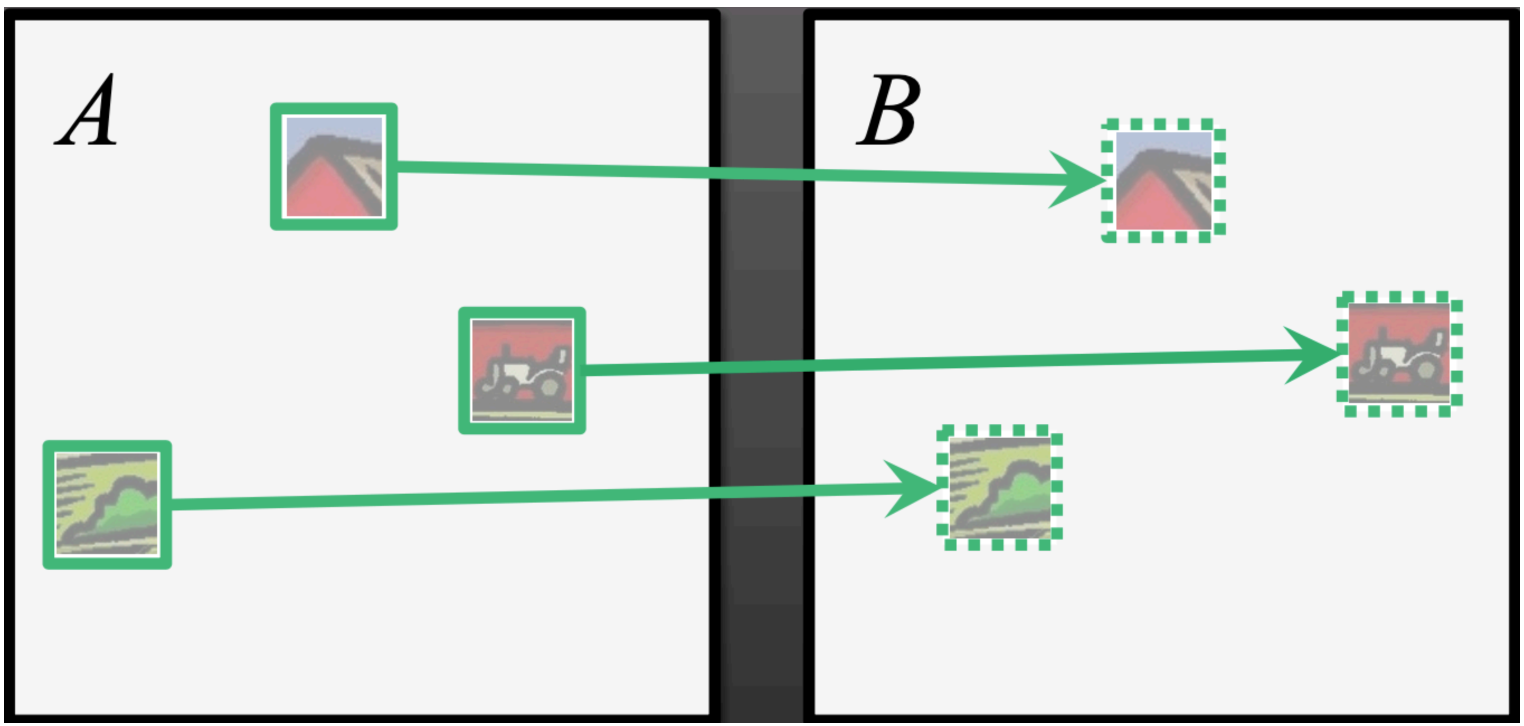
PatchMatch

- **A <u>randomized</u> algorithm for rapidly finding correspondences between image patches**

C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing”.** *SIGGRAPH 2009*

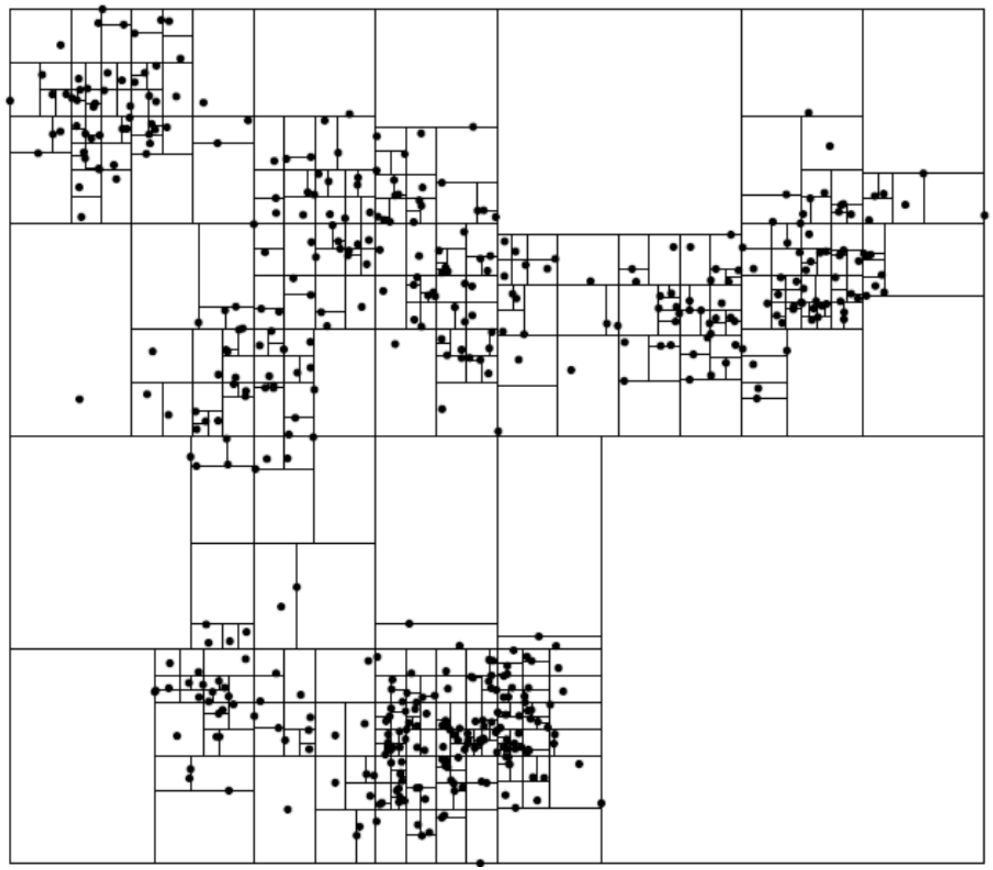
PatchMatch

- **Problem definition:**
- **Given images A and B, for each overlapping patch in image A, compute the offset to the nearest neighbor patch in image B**



C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Previous Work



Time:
$O(nlogn)$

Kd-tree with PCA

Credit: Hertzmann

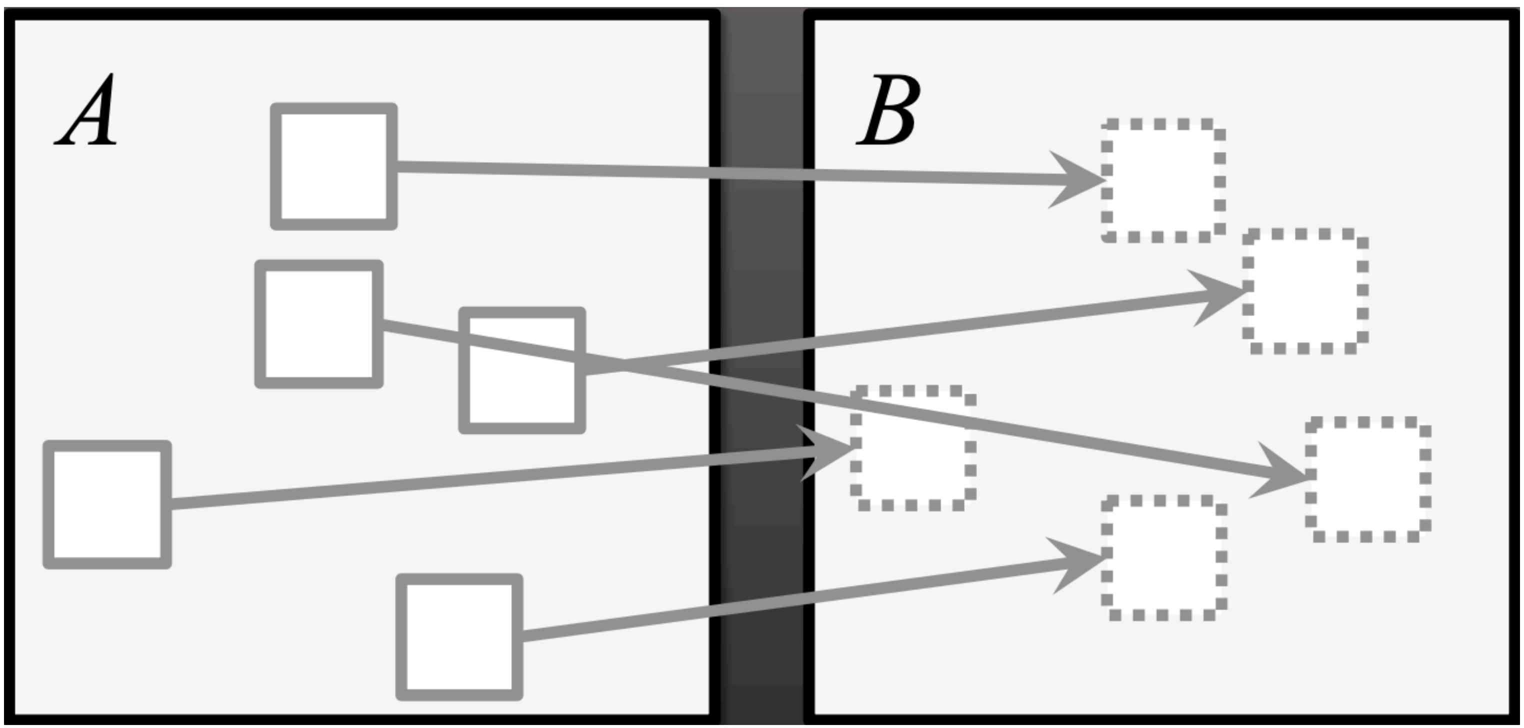
Key observation one

- **Law of large numbers: a non-trivial fraction of a large field of random offset assignments are likely to be good guesses**

C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing”.** *SIGGRAPH 2009*

Step 1: Initialization

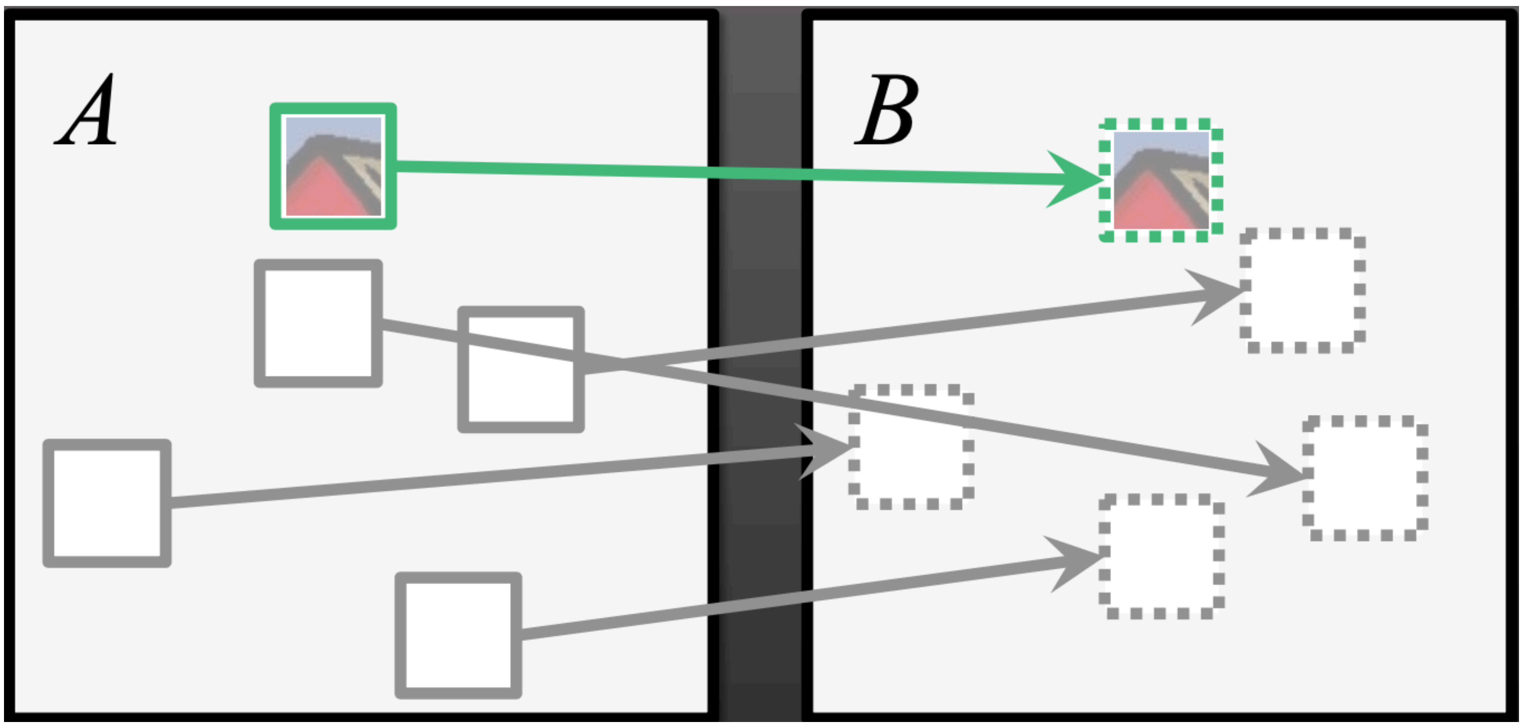
- **Initialization with random values(or derived from prior information)**
- $f(x,y) = random\ value$



C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 1: Initialization

- **Initialization with random values(or derived from prior information)**
- $f(x,y) = random\ value$



C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Key observation two: spatial coherence

- **High coherence of nearest neighbors in natural images**
- **Nearest neighbor of patch at (x,y) should be a strong hint for where to find nearest neighbor of patch at (x+1,y)**

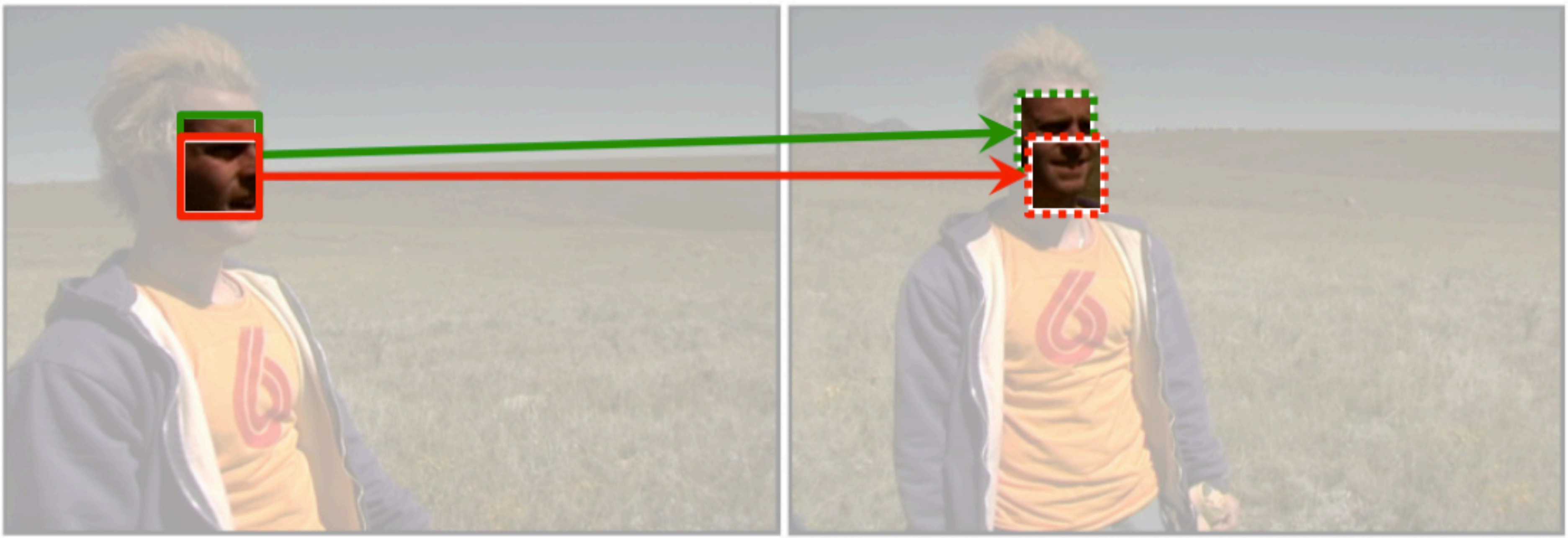
C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing".** *SIGGRAPH 2009*

Key observation two: spatial coherence

Credit: C. Barnes

C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing**”. *SIGGRAPH 2009*

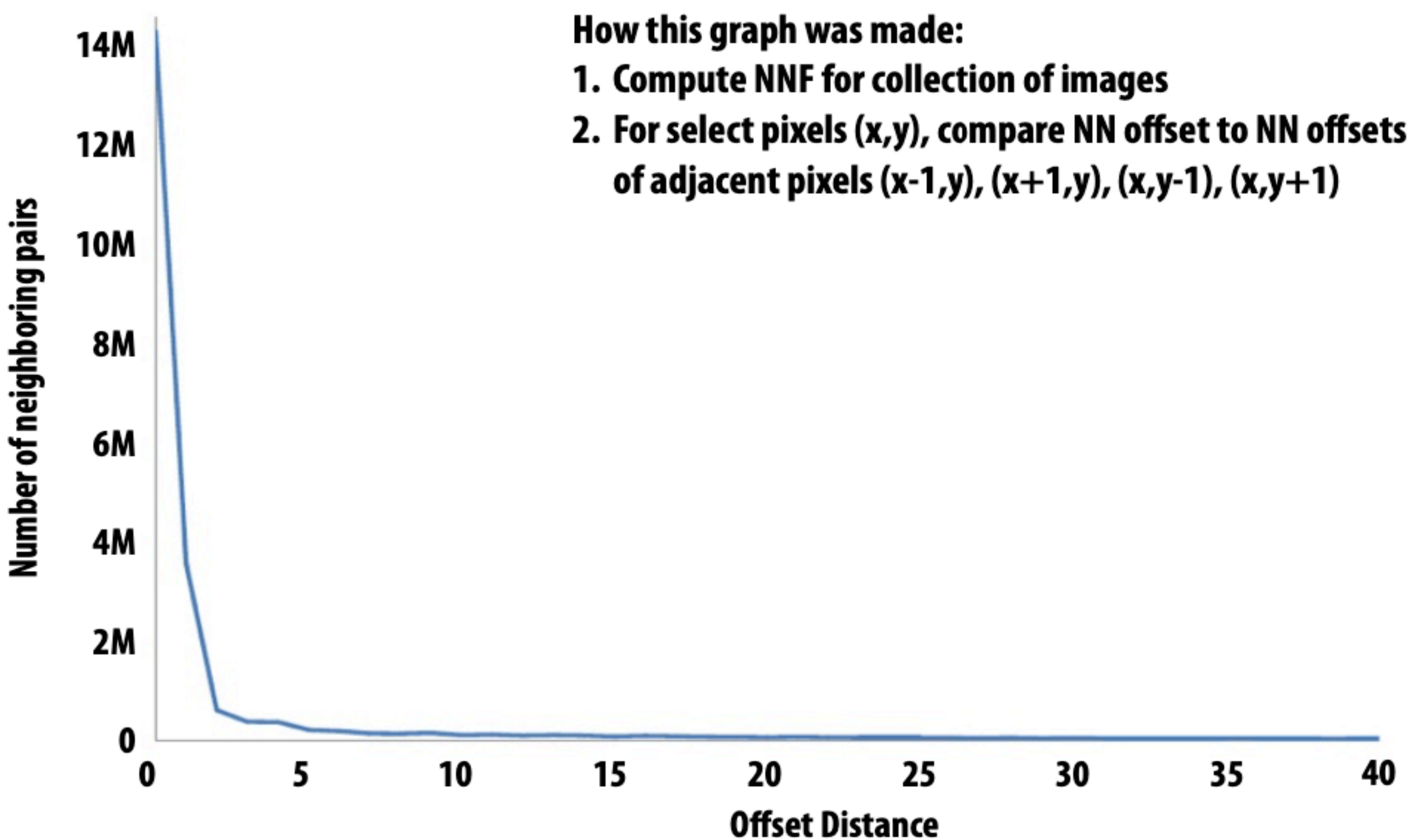
Key observation two: spatial coherence



Credit: C. Barnes

C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing”.** *SIGGRAPH 2009*

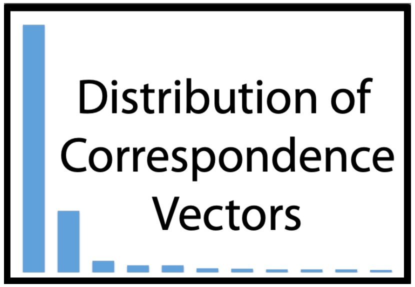
Use Statistics

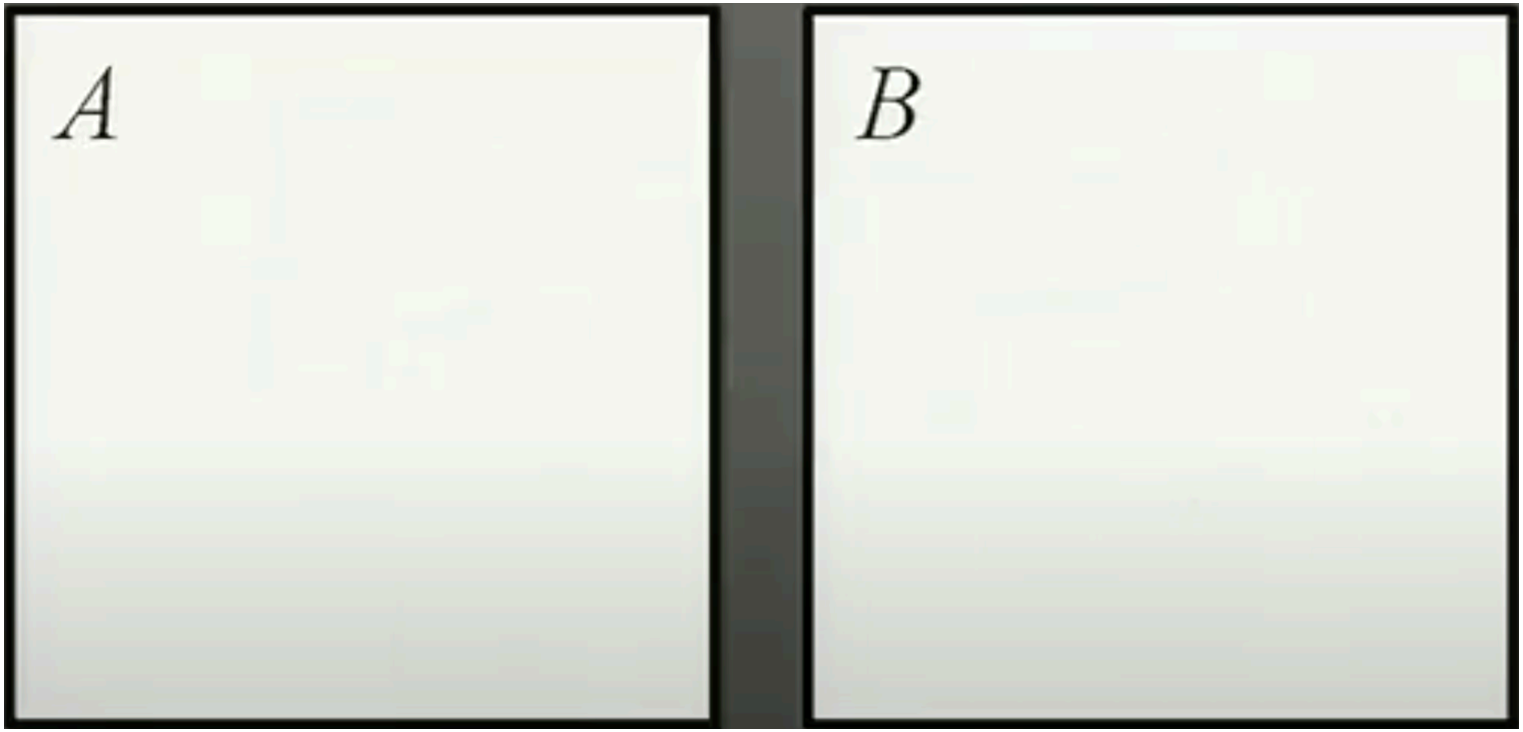


C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor(or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

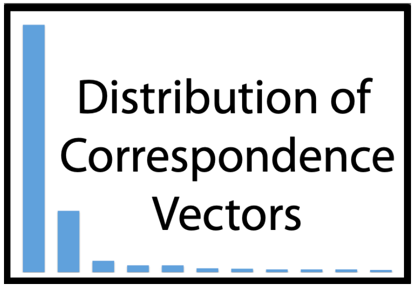


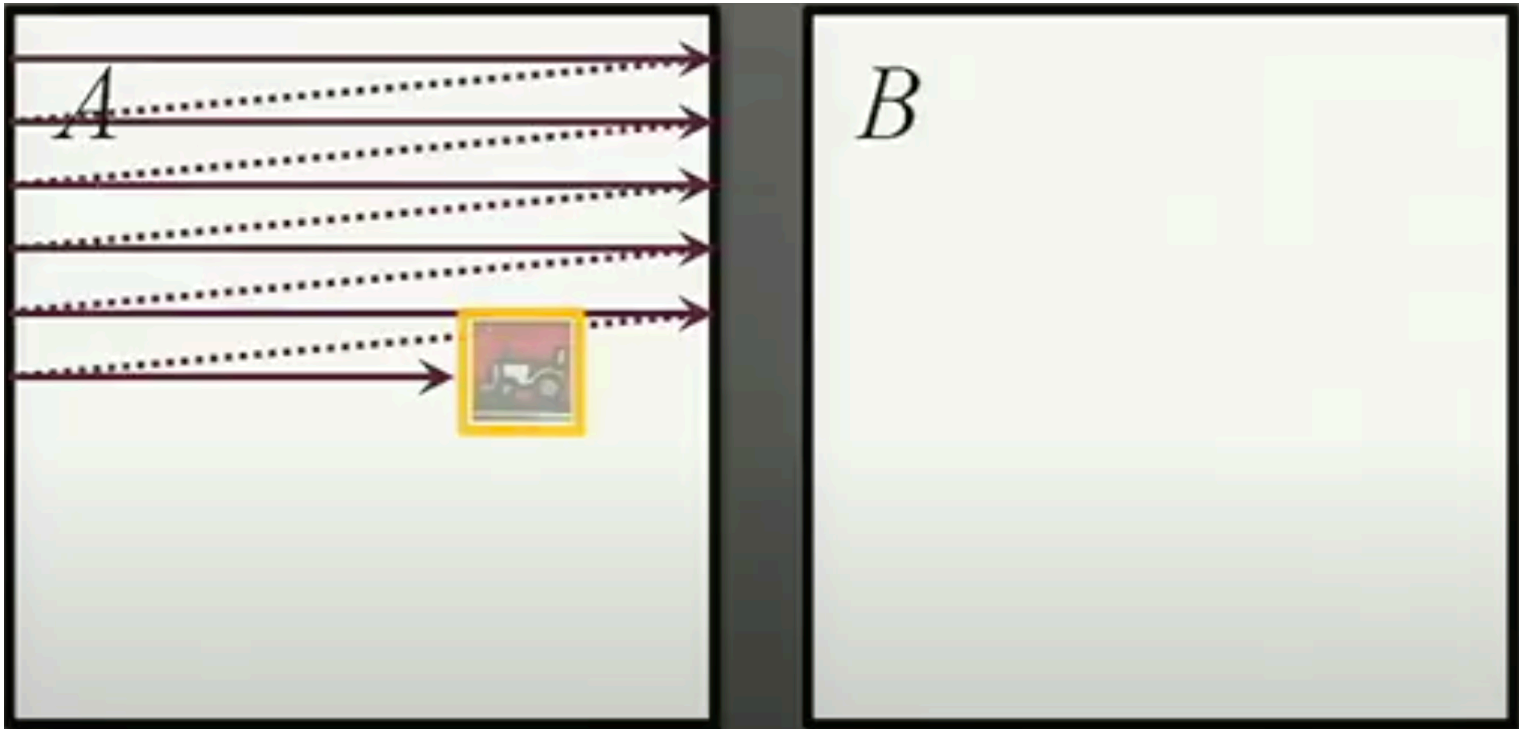


C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing**”. *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor(or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

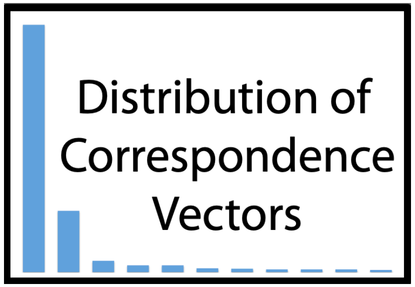


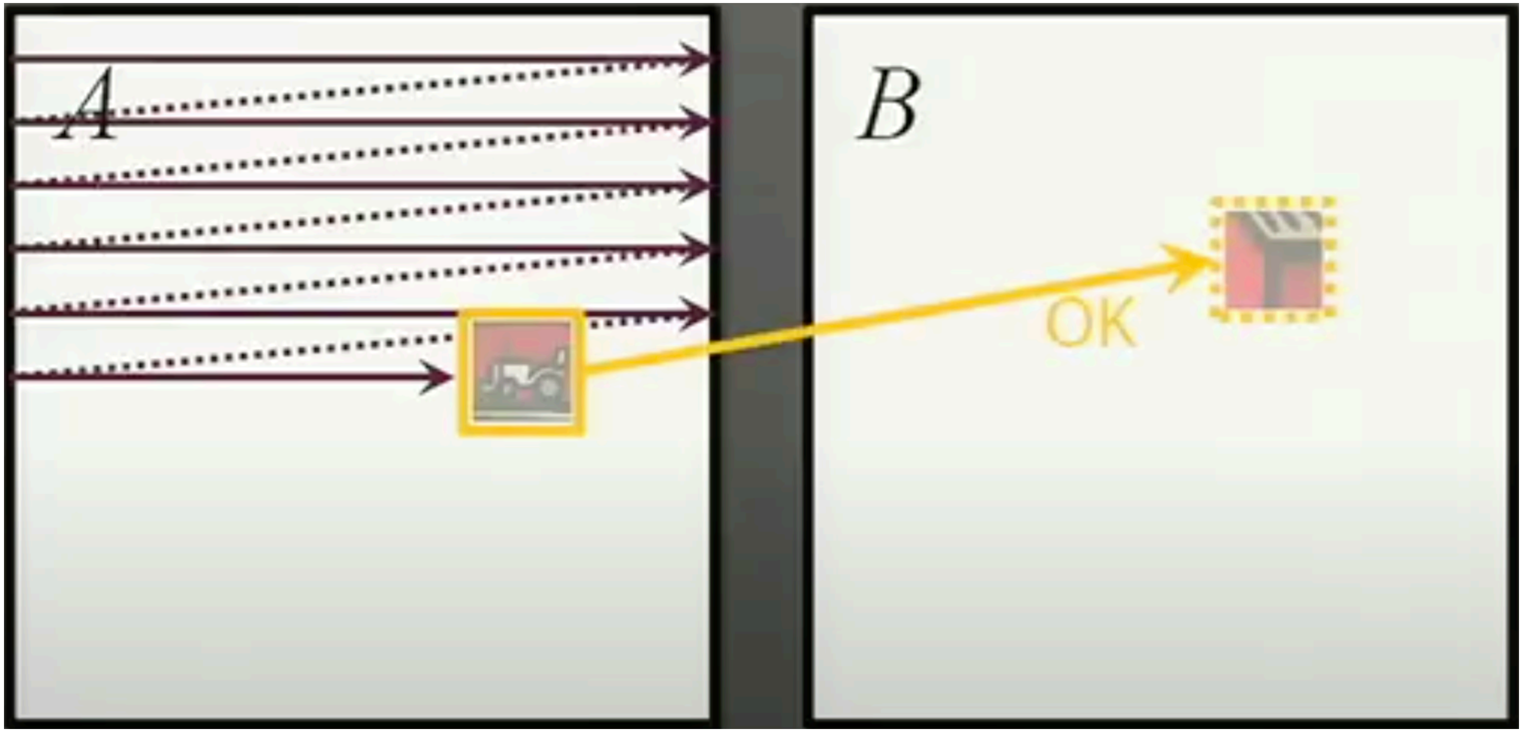


C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

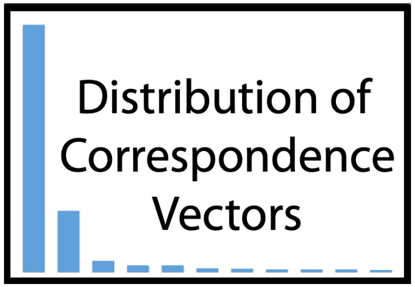


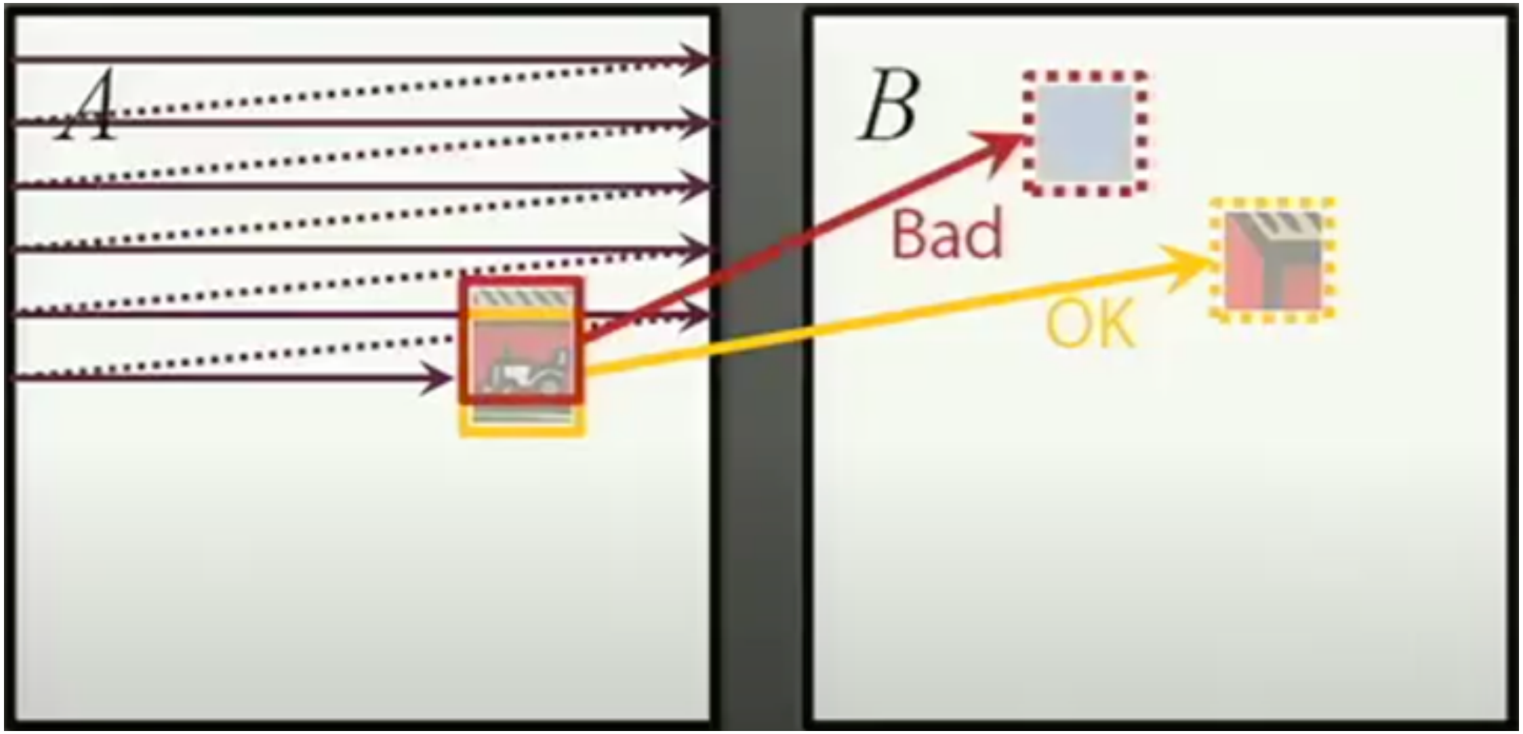


C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

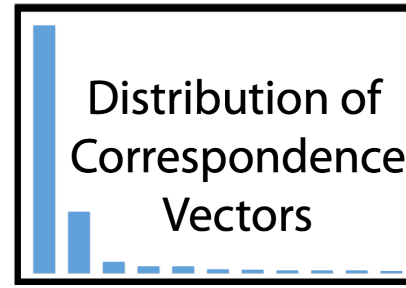


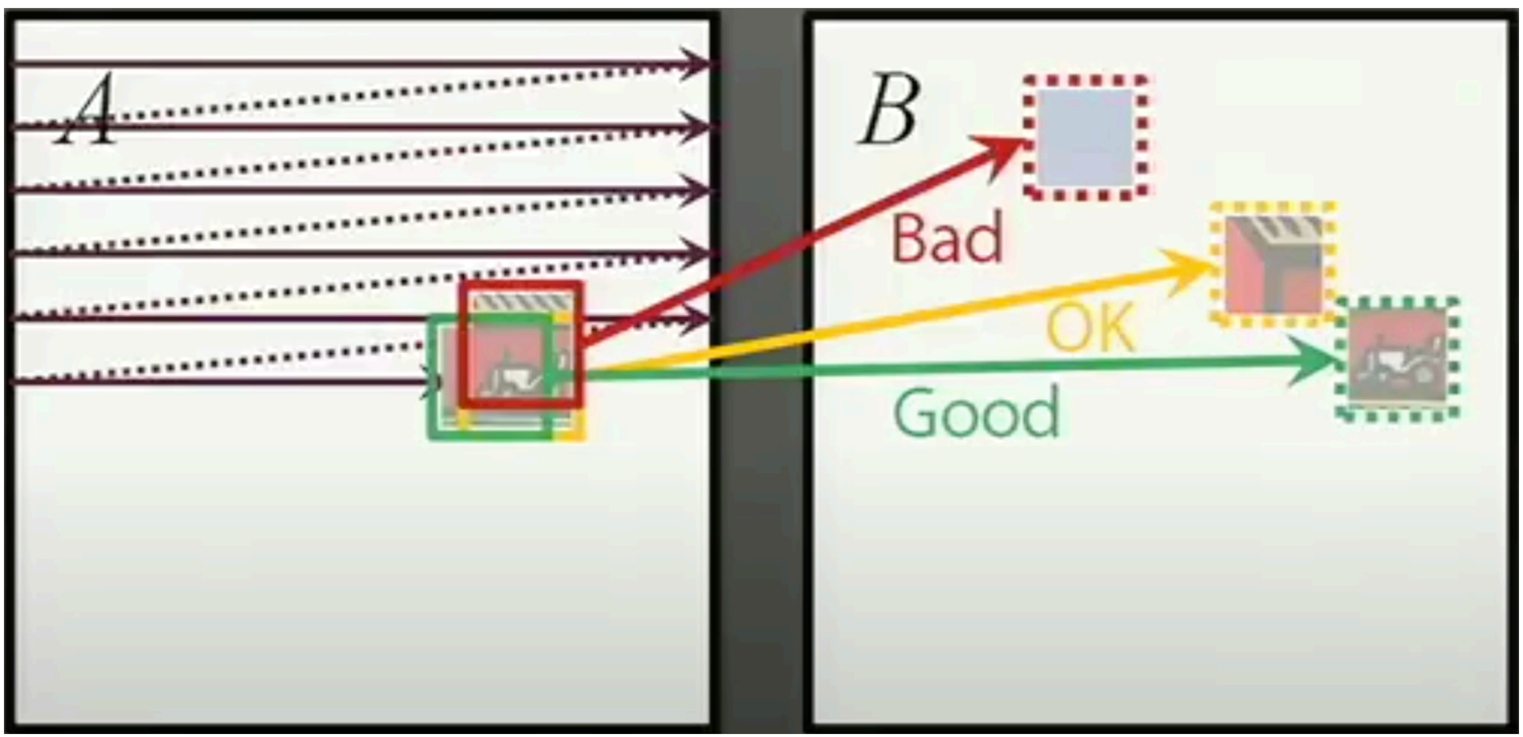


C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing”.** *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

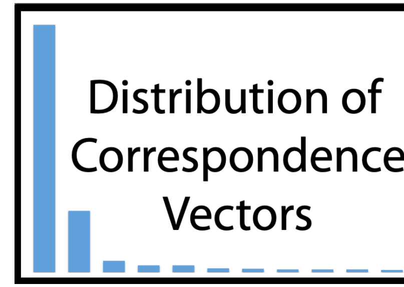


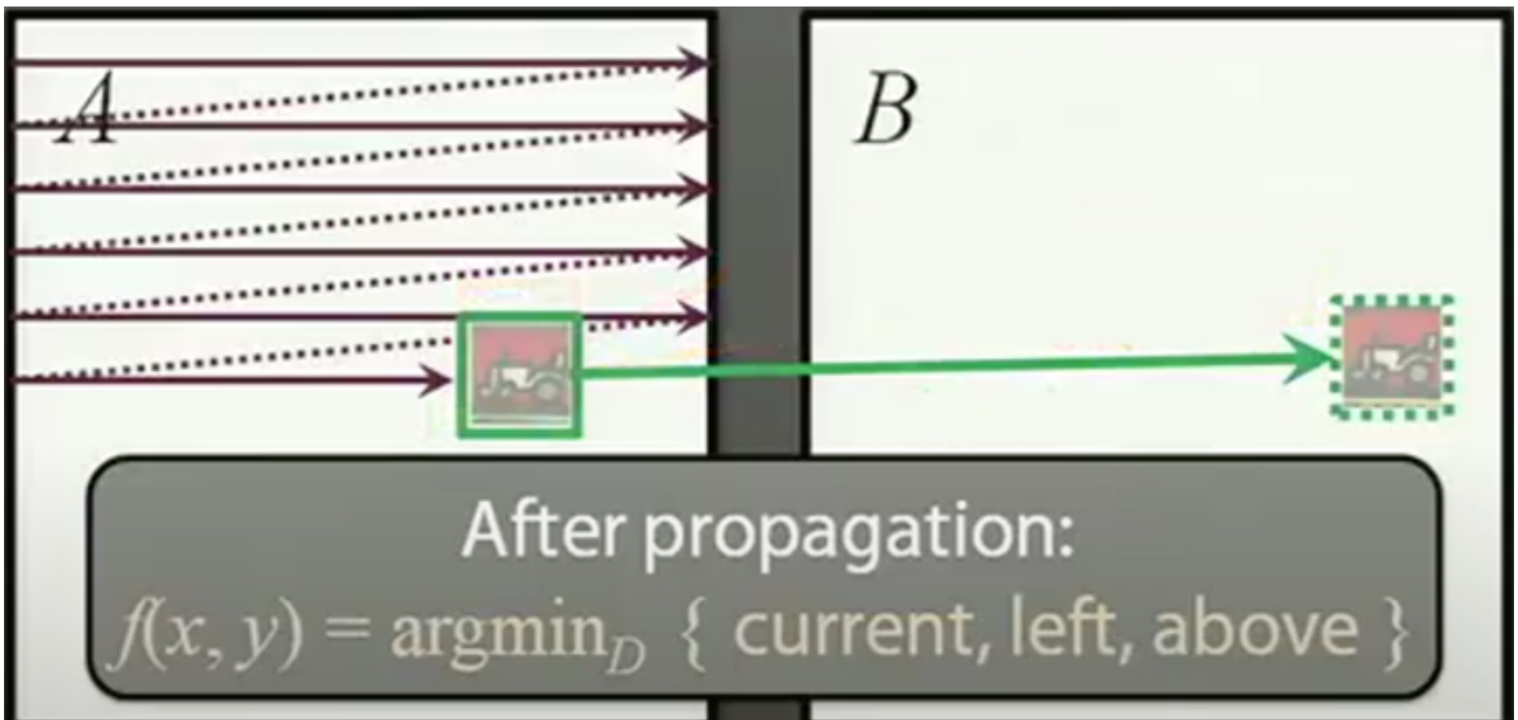


C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing”.** *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

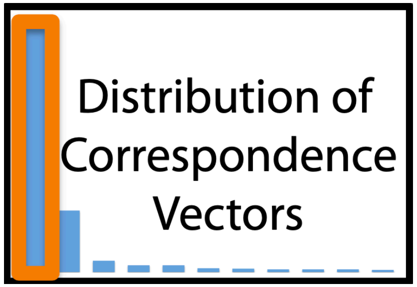


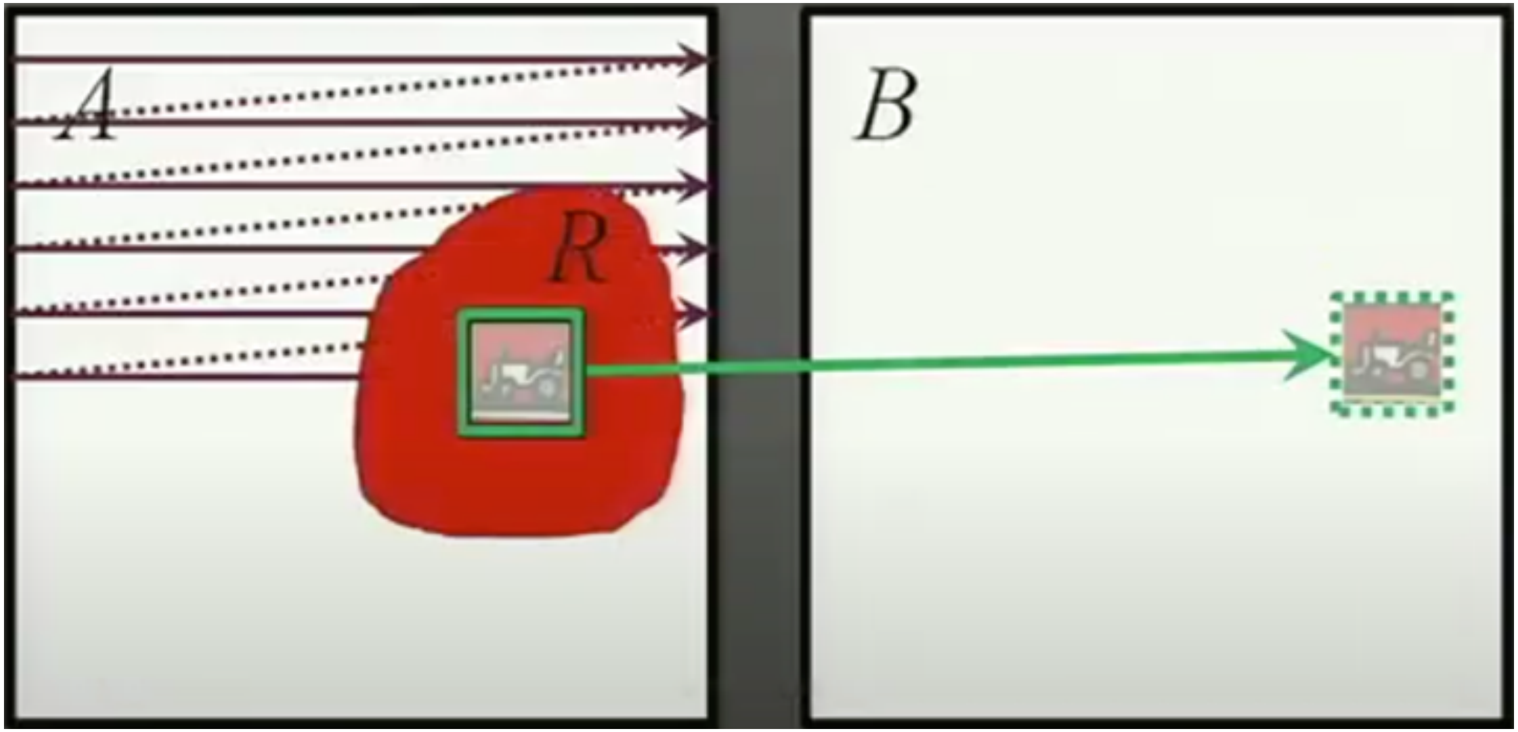


C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x, y) = argmin_d(f(x, y), f(x-1, y), f(x, y-1))$

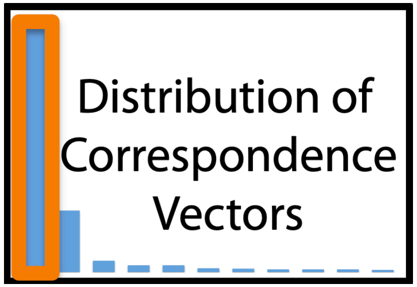


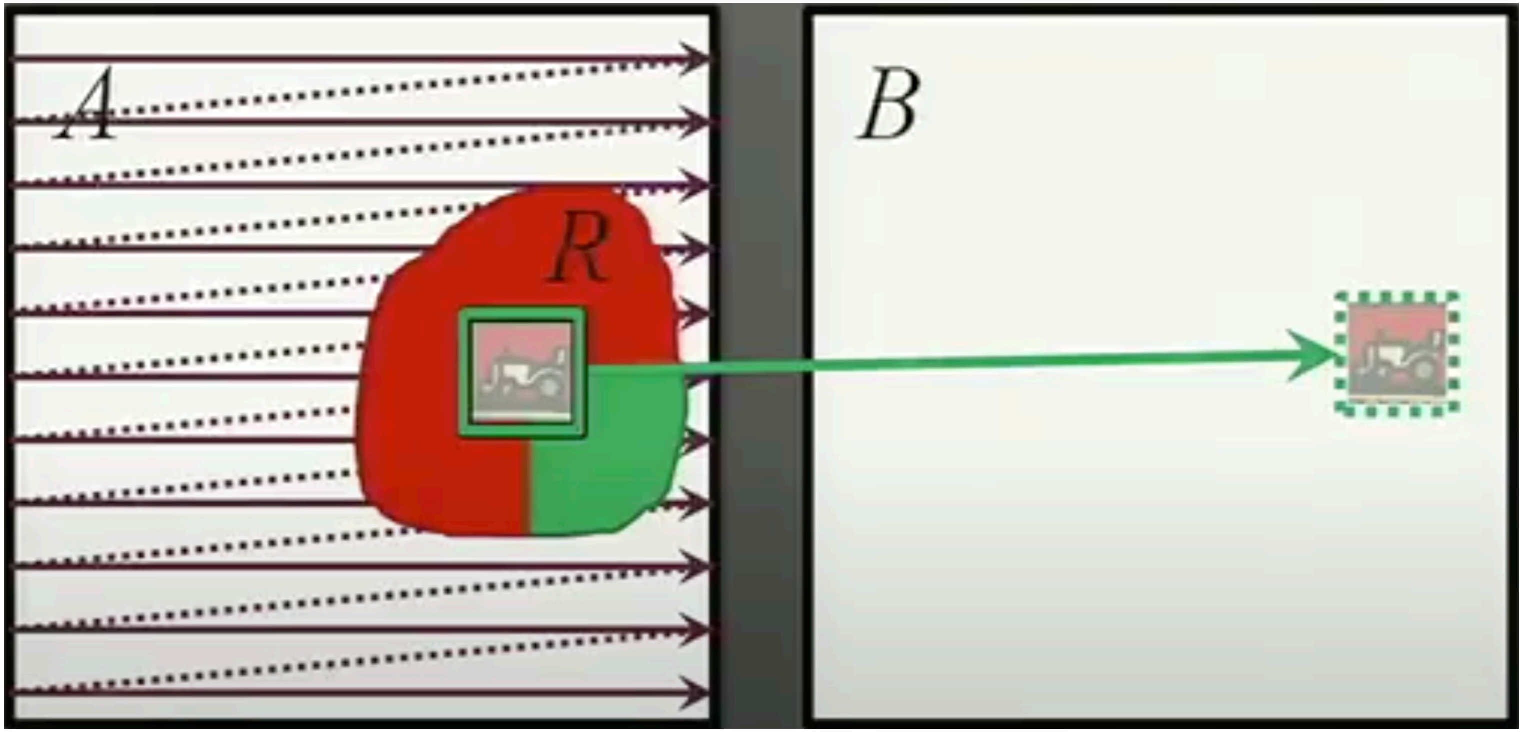


C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x-1,y), f(x,y-1))$

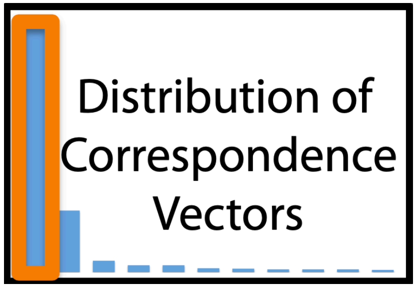


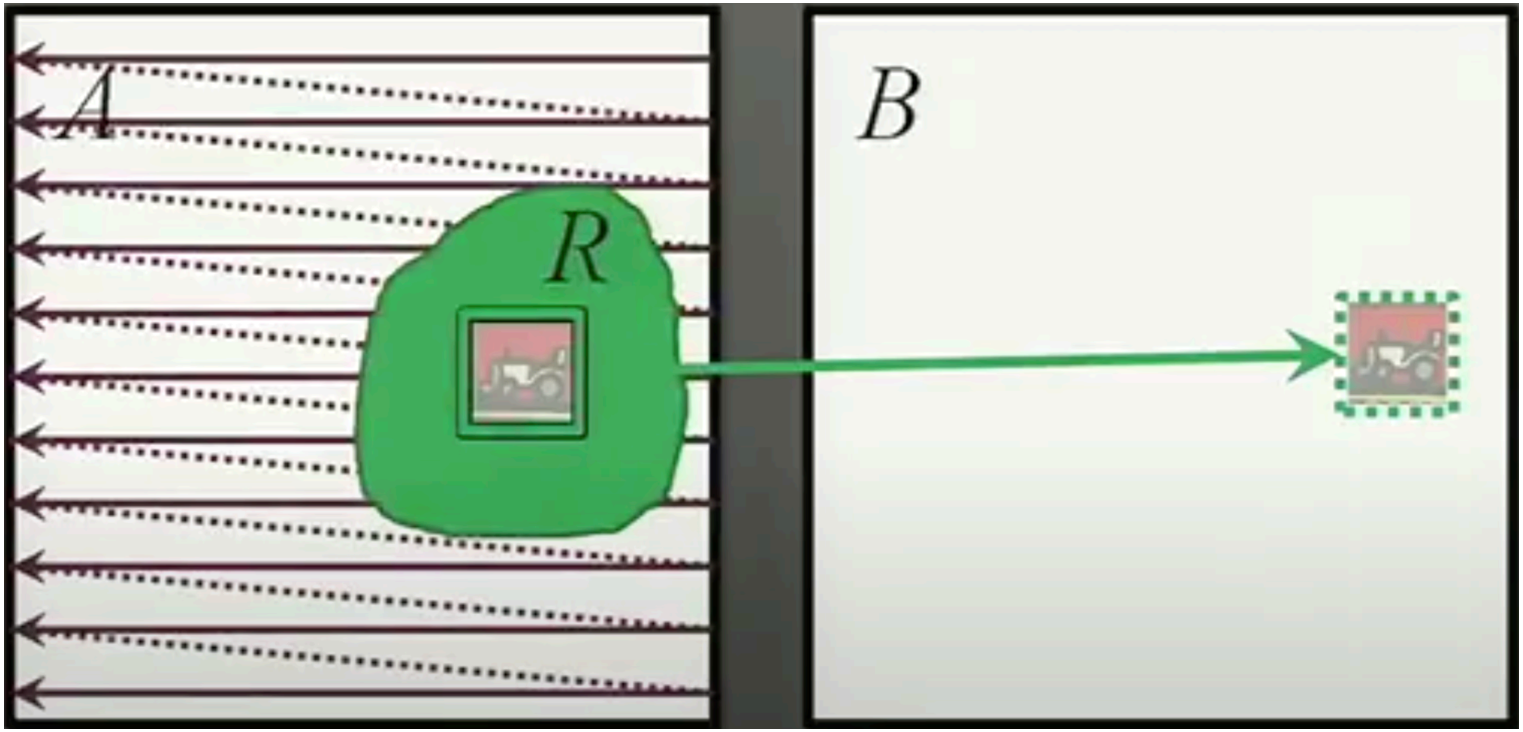


C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 2: Propagation

- **Try to improve offset estimate by exploiting spatial coherence with left and top neighbor (or right, bottom)**
- $f(x,y) = argmin_d(f(x,y), f(x+1,y), f(x,y+1))$

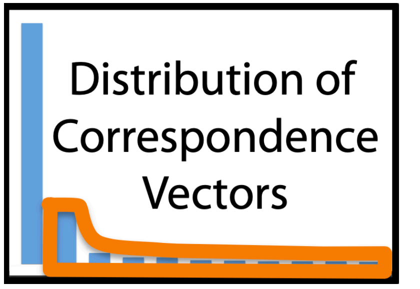




C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Step 3: Random Search

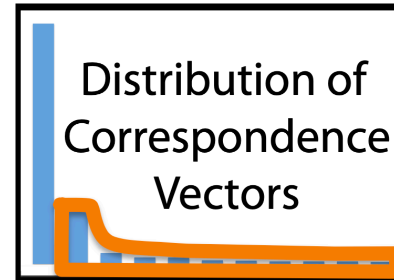
- **Avoiding local minima**
- **Random search in the neighborhood of the best offset found so far.**
- $f(x, y) = argmin_d\{candidate\ correspondence\}$

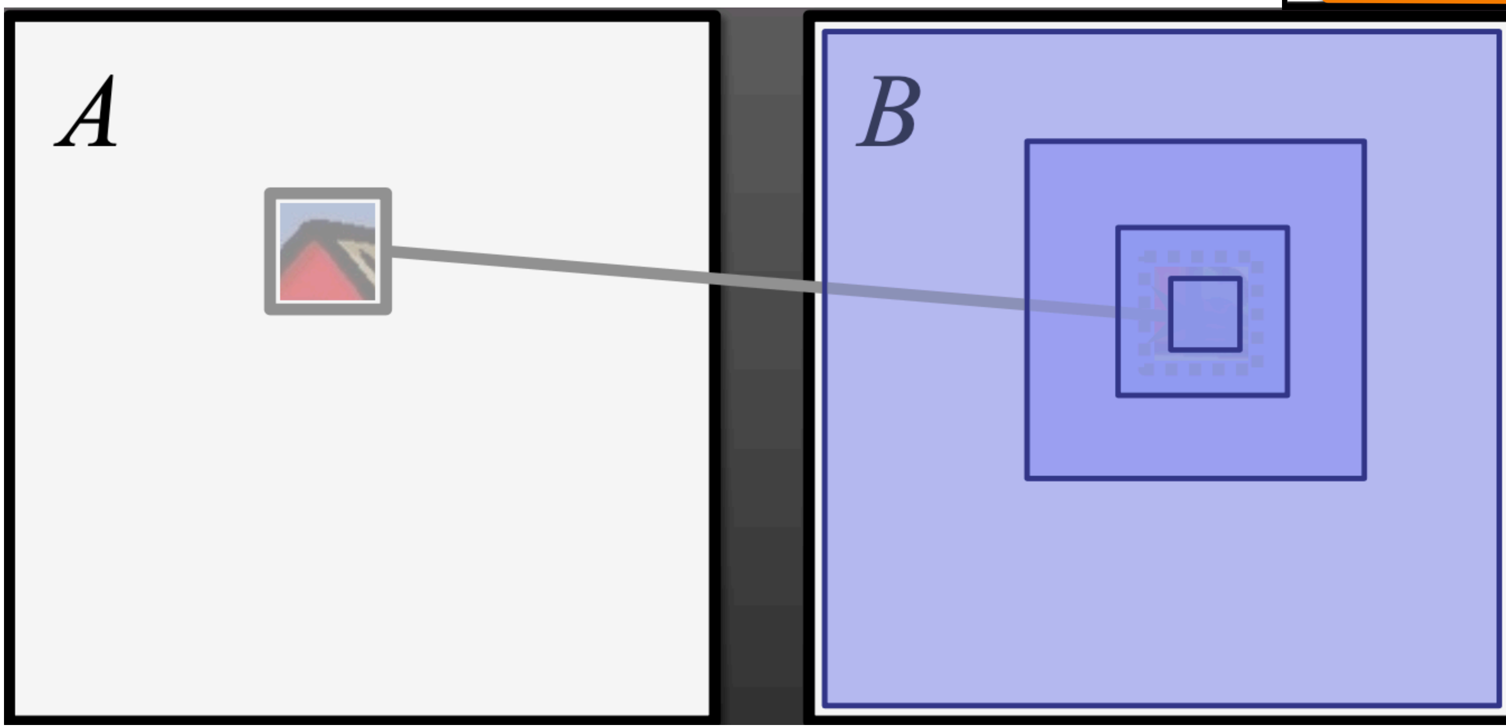


C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing”.** *SIGGRAPH 2009*

Step 3: Random Search

- **random search in the neighborhood of the best offset found so far.**
- $f(x, y) = argmin_d\{candidate\ correspondence\}$

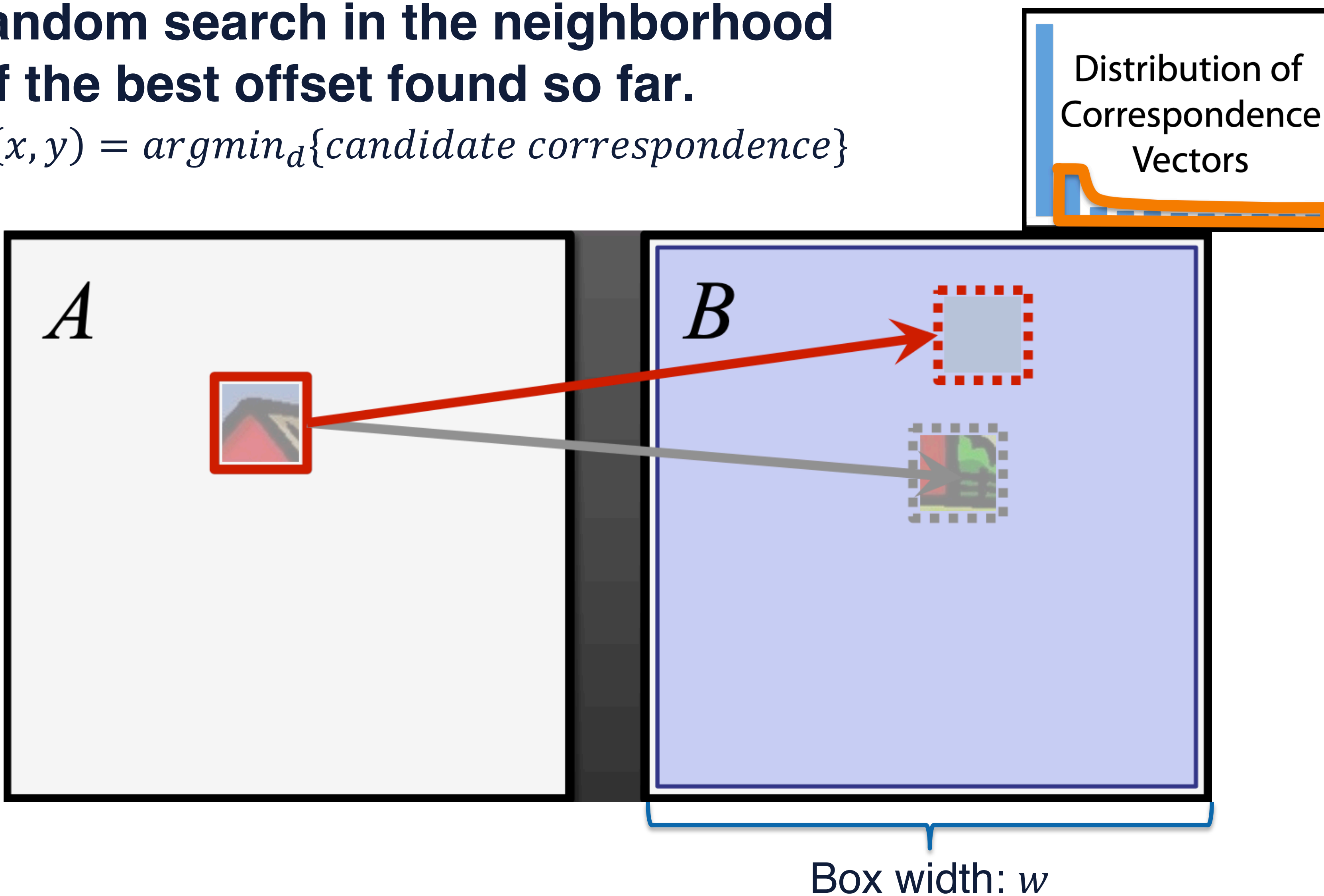




C. Barnes, et. al. “**PatchMatch: A randomized correspondence algorithm for structural image editing**”. *SIGGRAPH 2009*

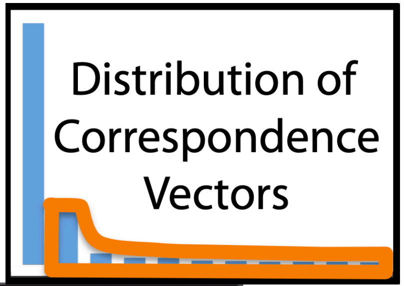
Step 3: Random Search

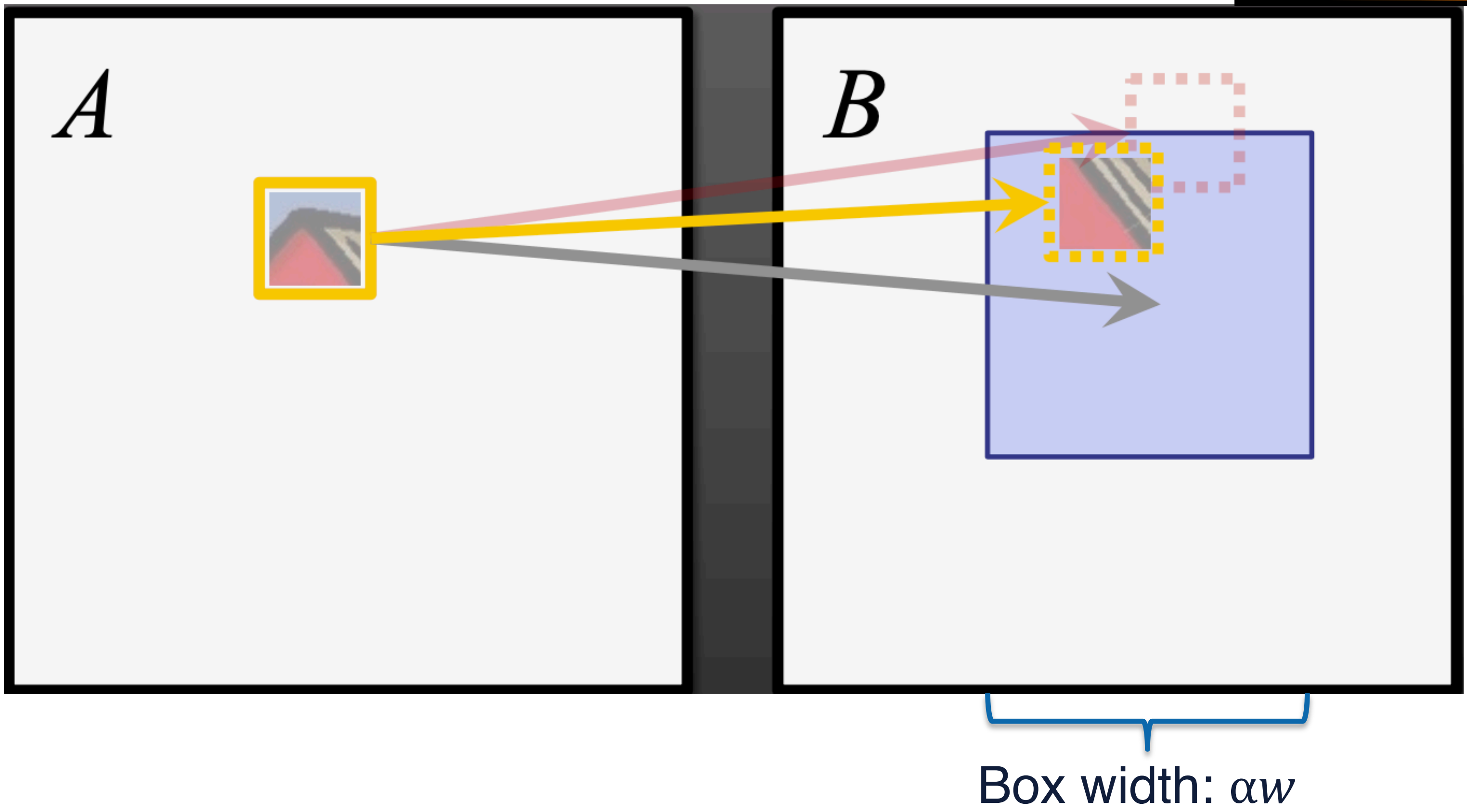
- **random search in the neighborhood of the best offset found so far.**
- $f(x,y) = argmin_d\{candidate\ correspondence\}$



Step 3: Random Search

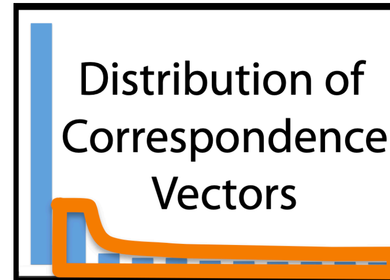
- **random search in the neighborhood of the best offset found so far.**
- $f(x, y) = argmin_d\{candidate\ correspondence\}$

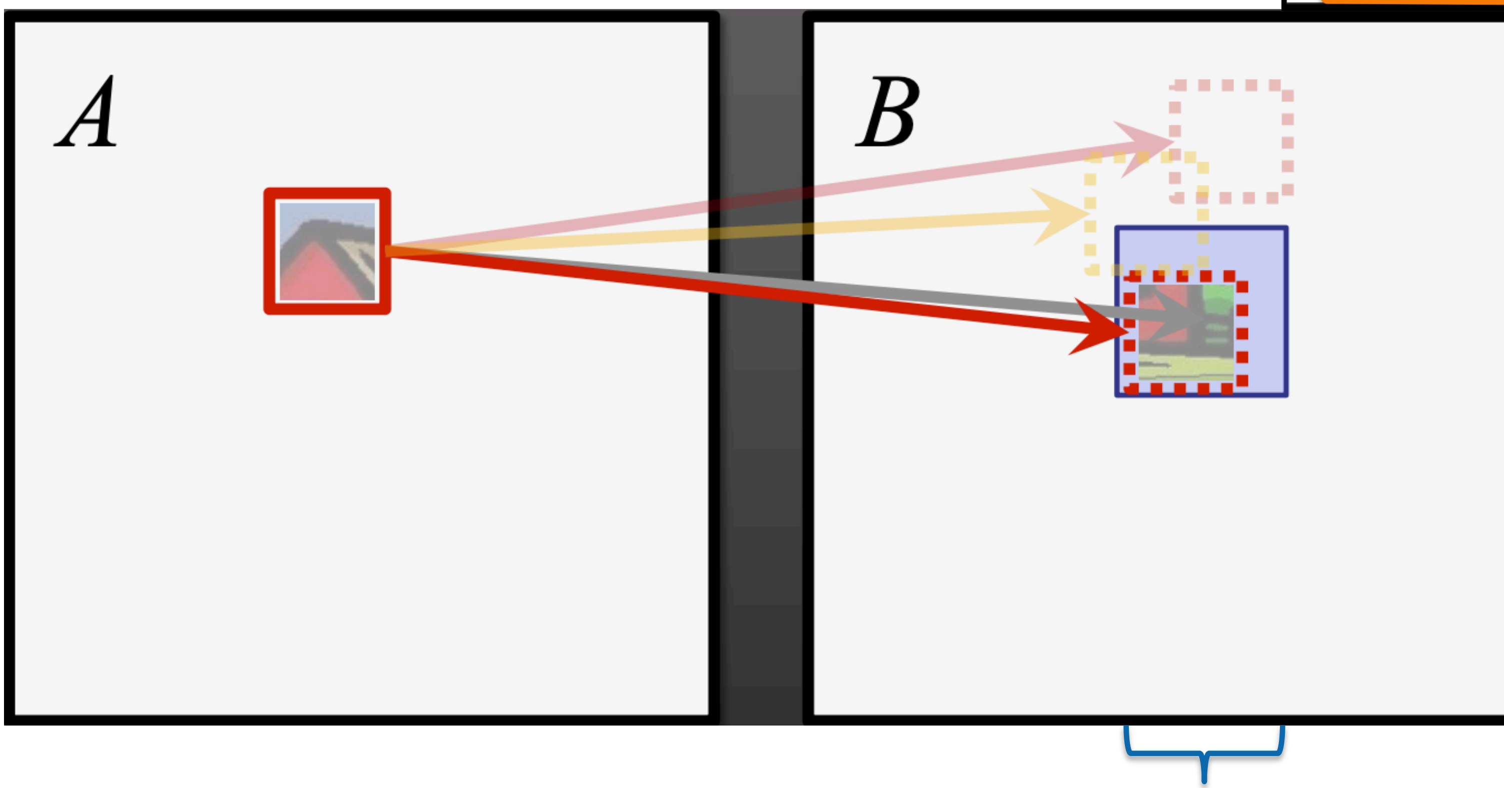




Step 3: Random Search

- **random search in the neighborhood of the best offset found so far.**
- $f(x, y) = argmin_d\{candidate\ correspondence\}$

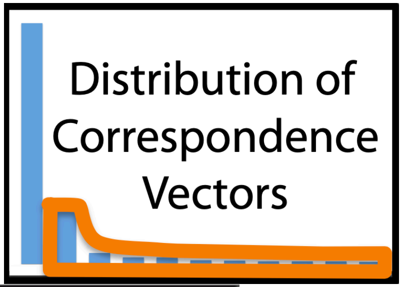


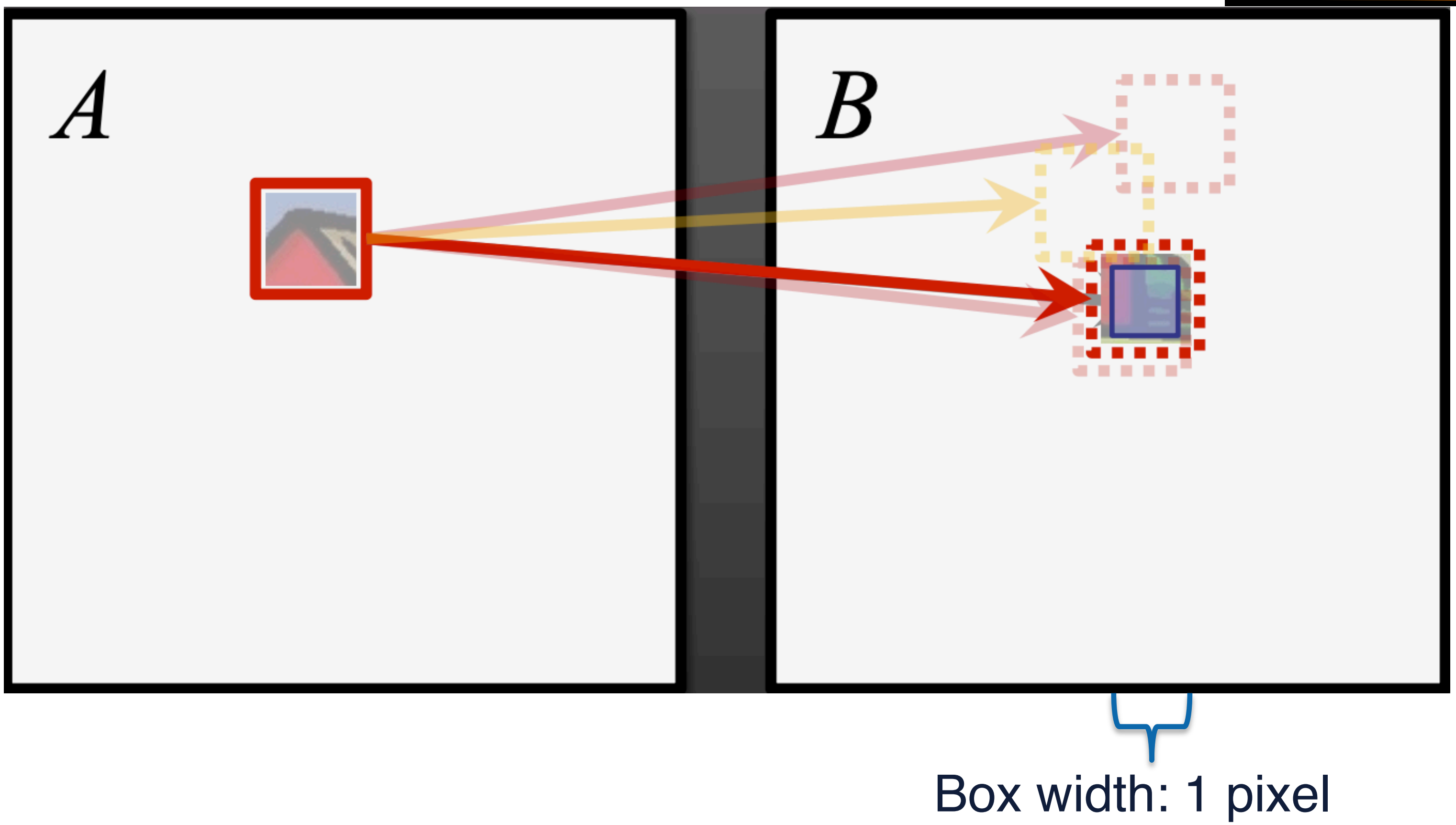


Box width: $\alpha^2 w$

Step 3: Random Search

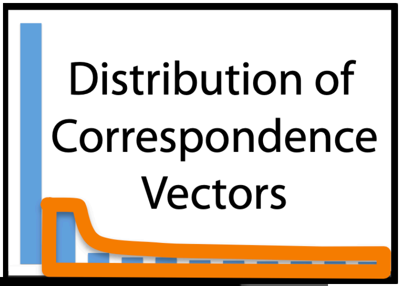
- **random search in the neighborhood of the best offset found so far.**
- $f(x,y) = argmin_d\{candidate\ correspondence\}$

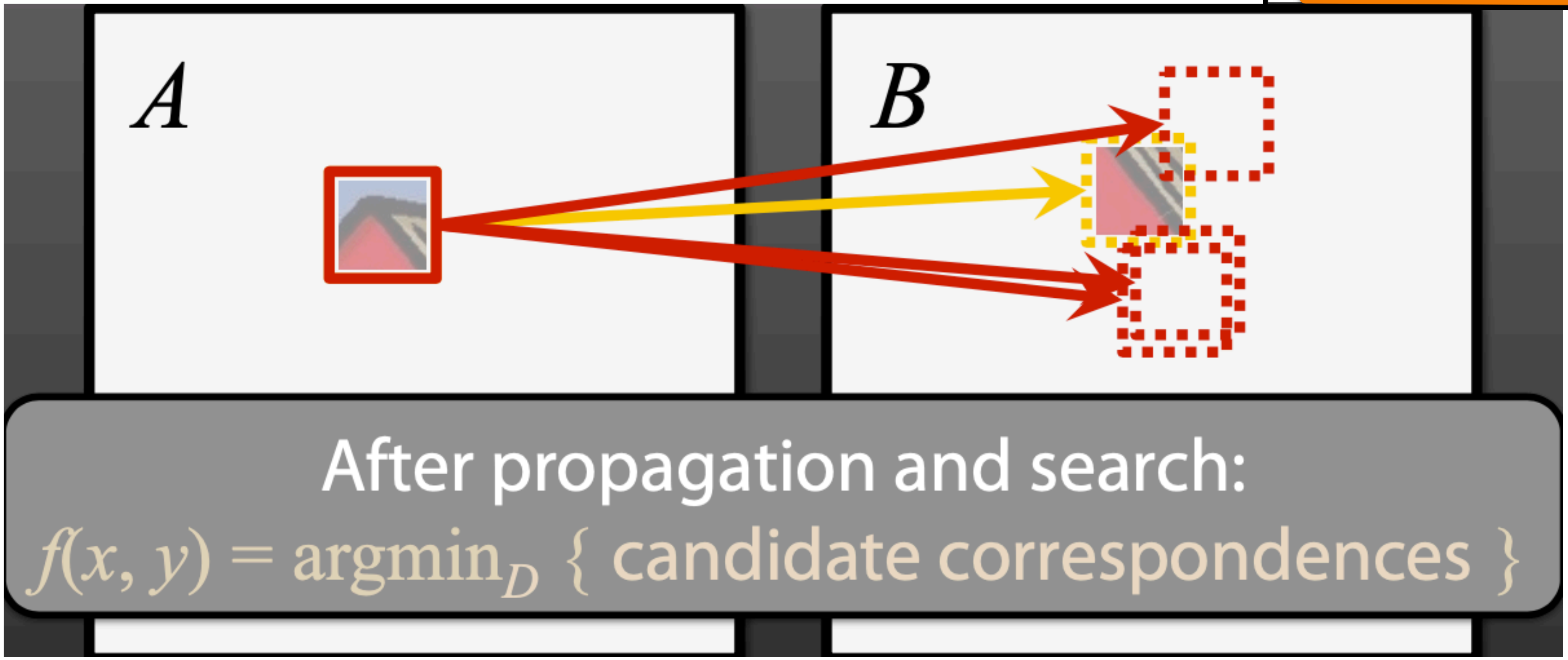




Step 3: Random Search

- **random search in the neighborhood of the best offset found so far.**
- $f(x, y) = argmin_d\{candidate\ correspondence\}$





C. Barnes, et. al. "**PatchMatch: A randomized correspondence algorithm for structural image editing**". *SIGGRAPH 2009*

Summary

- PatchMatch:
    - Step 1: Initialization
    - Step 2: Propagation
    - Step 3: Random Search

Repeat until converged

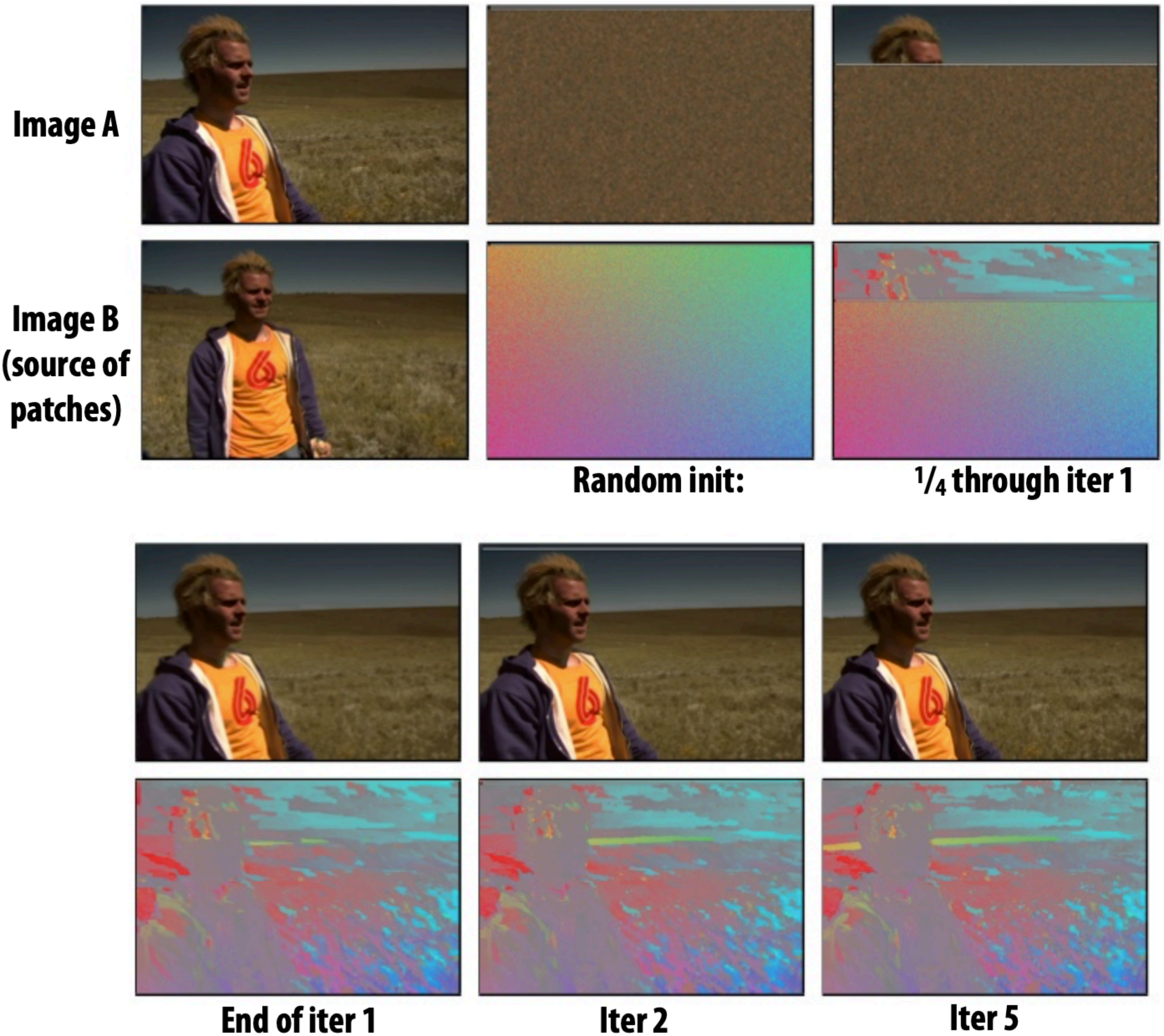
Summary

- PatchMatch:
    - Step 1: Initialization
    - Step 2: Propagation
    - Step 3: Random Search

Repeat until converged

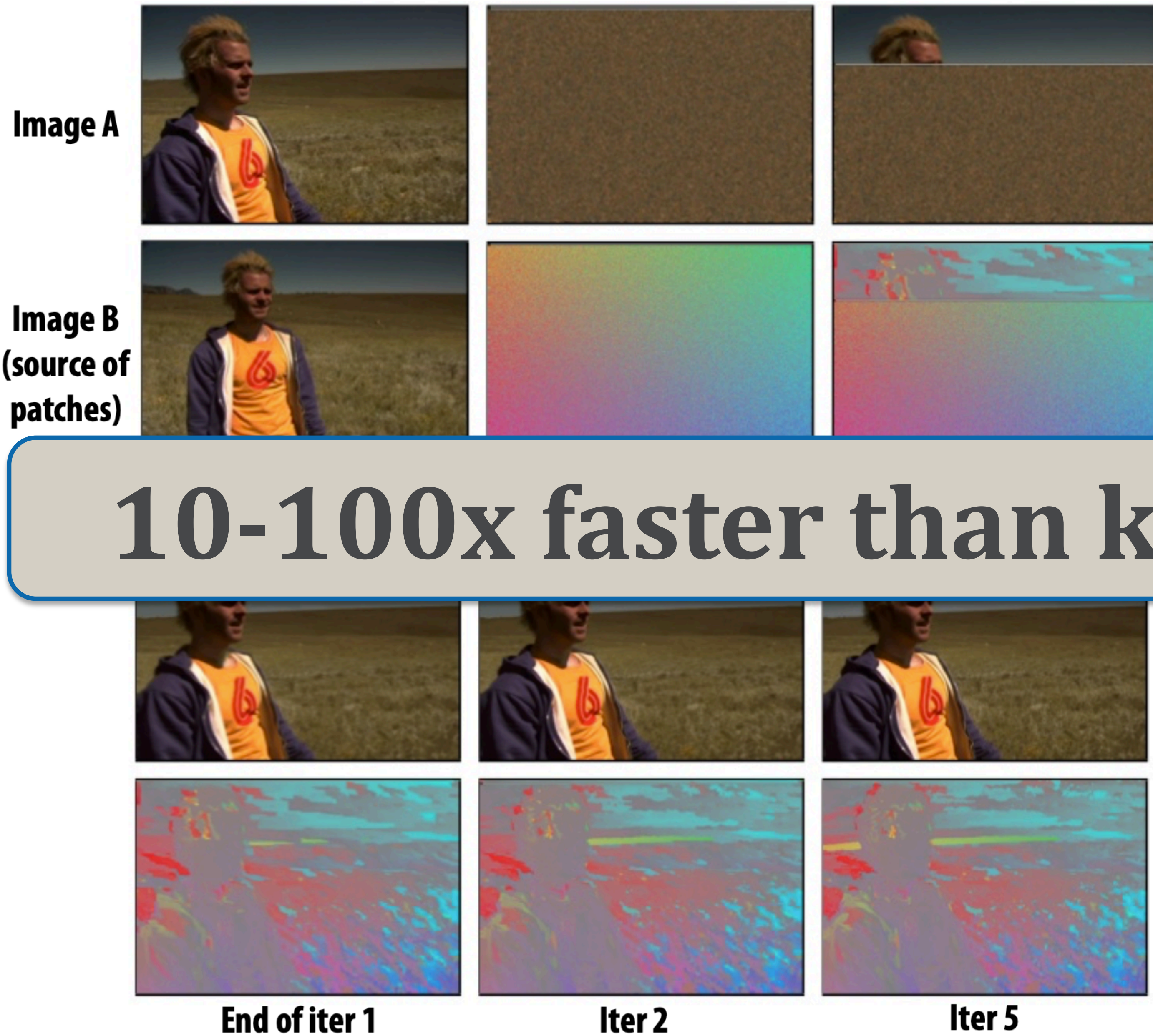
- key insights:
    - some good patch matches can be found via random sampling.
    - natural coherence in the imagery allows us to propagate such matches quickly to surrounding areas.

Experiment: Reconstruct A using patches from B



Credit: Barnes

Experiment: Reconstruct A using patches from B



10-100x faster than kd-tree!

Credit: Barnes

Why does it work?

- Assume source and target images have equal size (M pixels) and that random initialization is used.

- The odds of any one location being assigned the best offset: 1 / M

- But for M pixels:
  - The odds of at least one offset being correctly assigned are quite good:

$$1 - (1 - \frac{1}{M})^M$$

E.g. M=10e5, this is (1– 0.367)

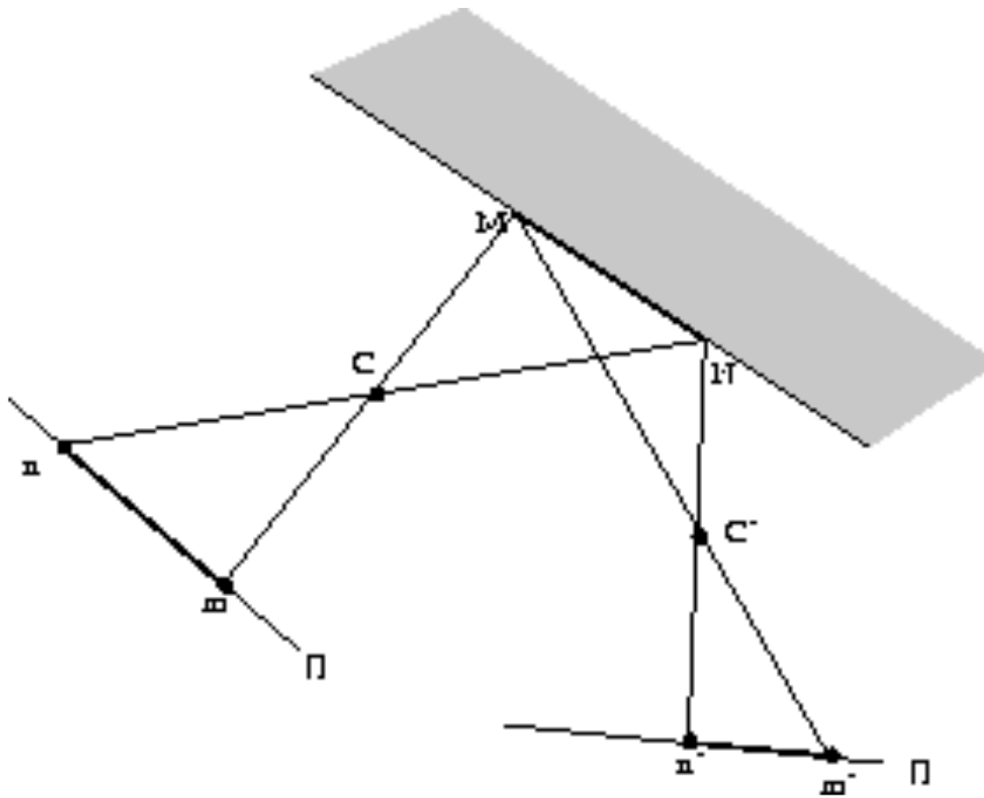
If top C nearest neighbors are enough, the odds will be $1 - (1 - \frac{C}{M})^M$

C. Barnes, et. al. "**Patchmatch: A randomized correspondence algorithm for structural image editing".** *SIGGRAPH 2009*

PatchMatch Stereo

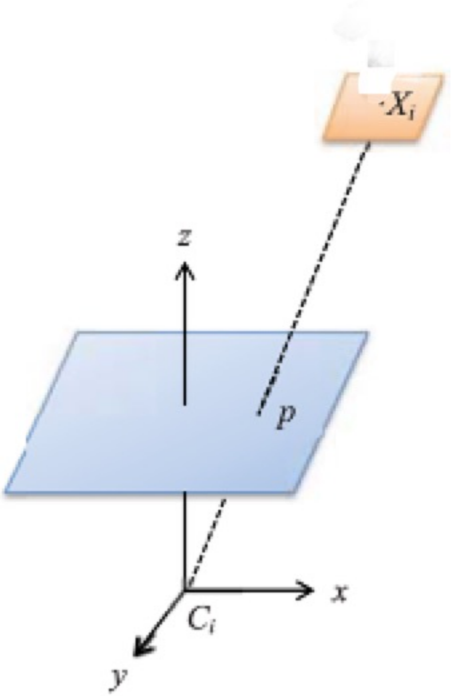
PatchMatch Stereo

Figure 6: Foreshortening due to the change of viewing position and direction.



PatchMatch Stereo

- **Extend to find an approximate nearest neighbor according to a plane.**
- **Offset -> depth**

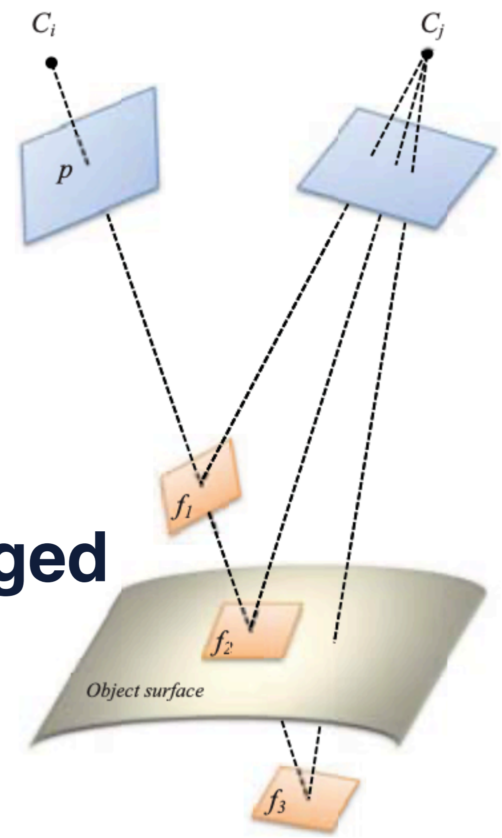


M. Bleyer, et. al. "**Patchmatch stereo - stereo matching with slanted support windows**", *BMVC 2011*

PatchMatch Stereo

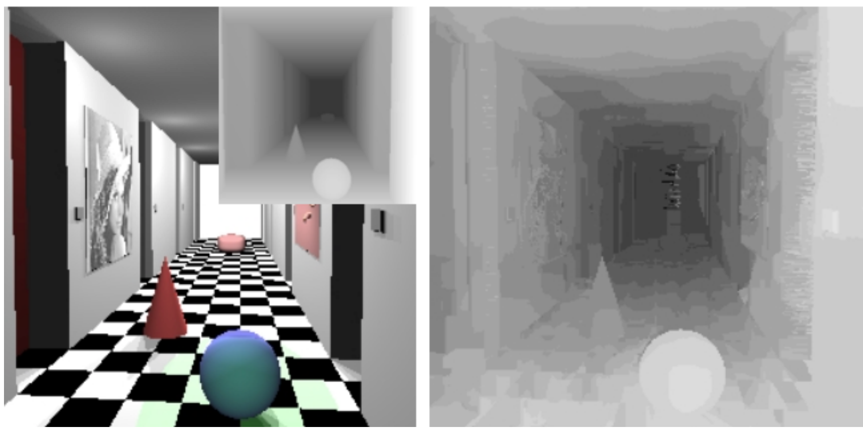
- Step 1: Initialization
- Step 2: Propagation
- Step 3: Random Search

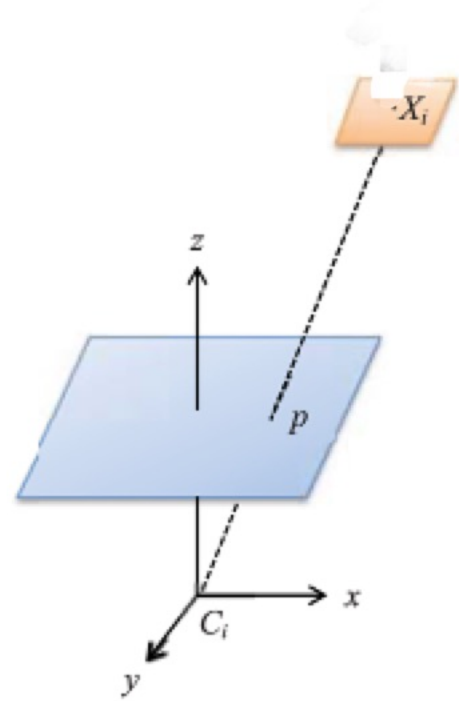
Repeat until converged



M. Bleyer, et. al. "**Patchmatch stereo - stereo matching with slanted support windows**", *BMVC 2011*

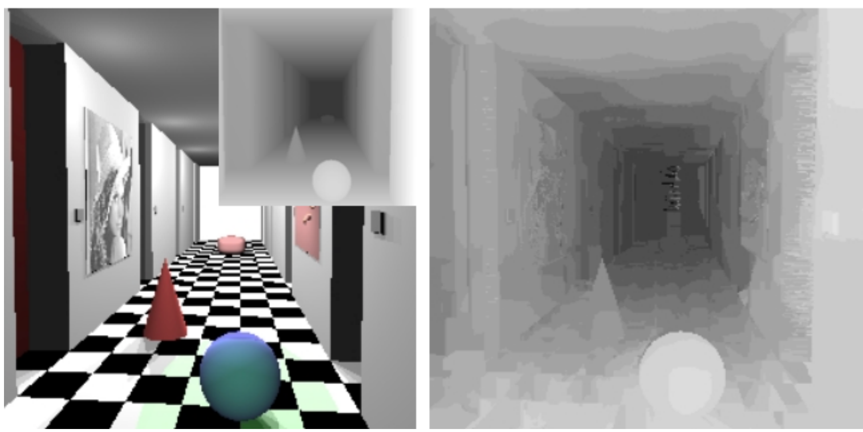
PatchMatch Stereo

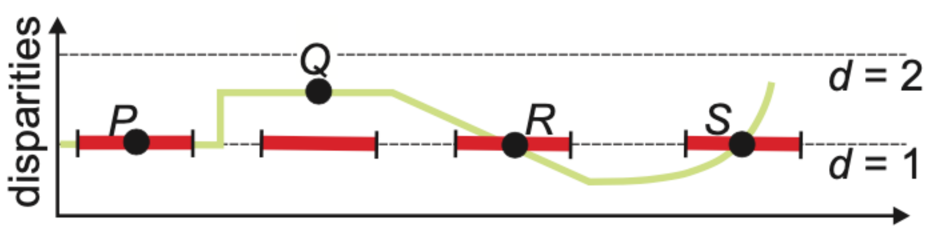




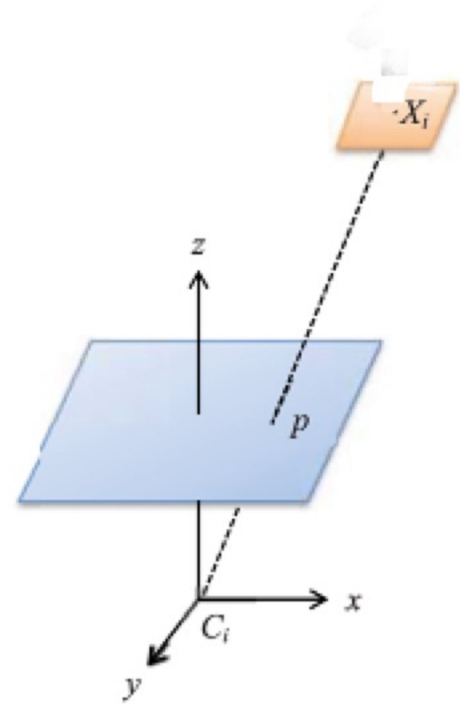
M. Bleyer, et. al. “**Patchmatch stereo - stereo matching with slanted support windows**”, *BMVC 2011*

PatchMatch Stereo



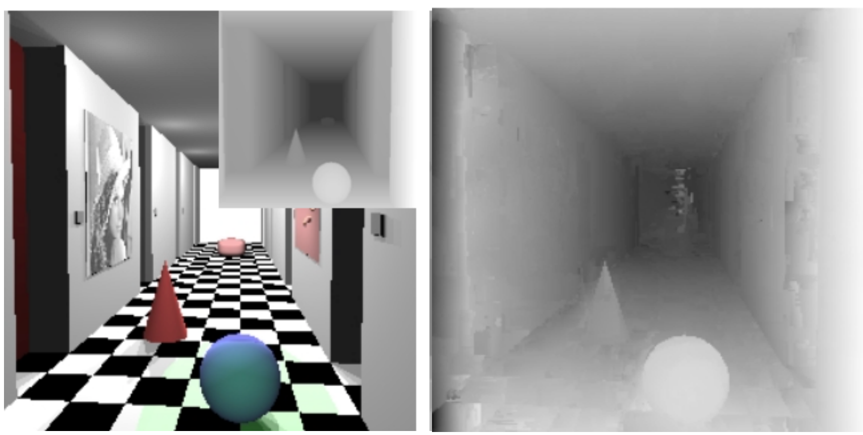


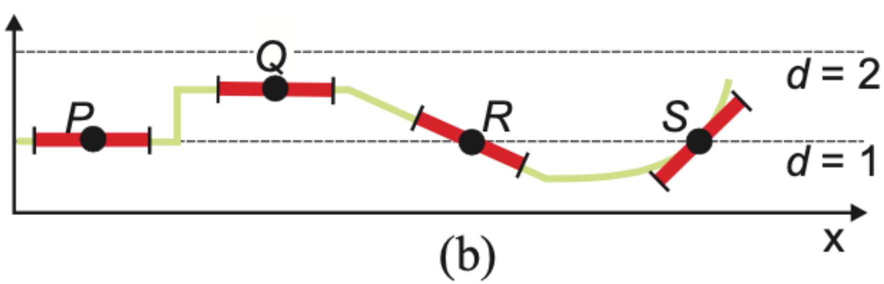
(a)

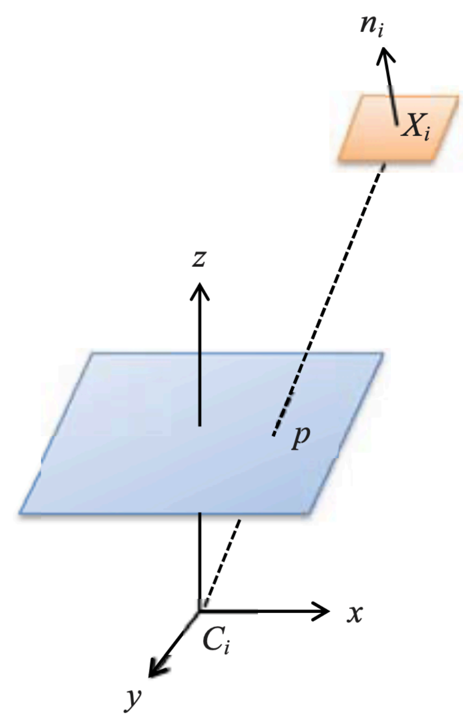


M. Bleyer, et. al. “**Patchmatch stereo - stereo matching with slanted support windows**”, *BMVC 2011*

PatchMatch Stereo



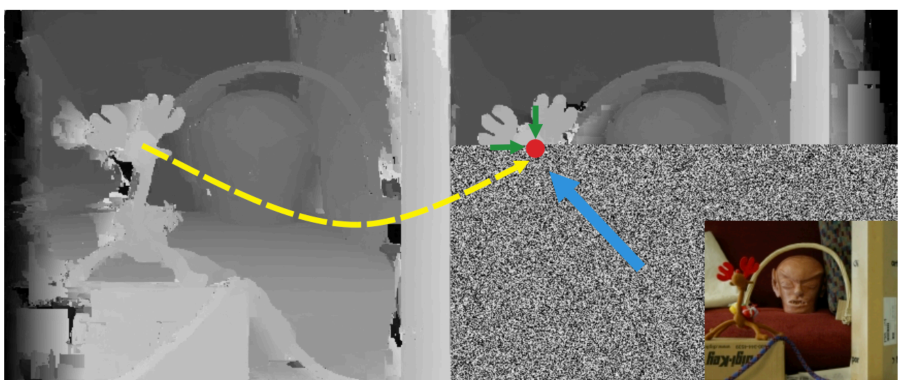




M. Bleyer, et. al. “**Patchmatch stereo - stereo matching with slanted support windows**”, *BMVC 2011*

PatchMatch Stereo

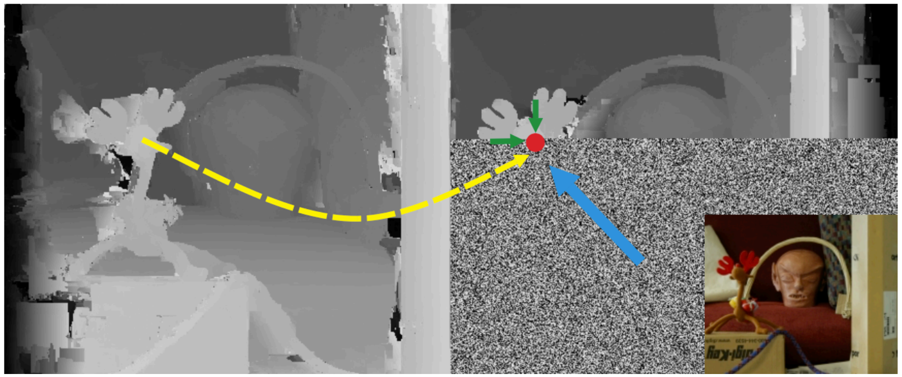
- For Each Pixel
    - Assign Random Depth and Normal
- For N Iterations
    - For Each Pixel
        - Propagate Depth and Normal From Neighbor
        - Sample New Random Depth and Normal
        - Update Depth

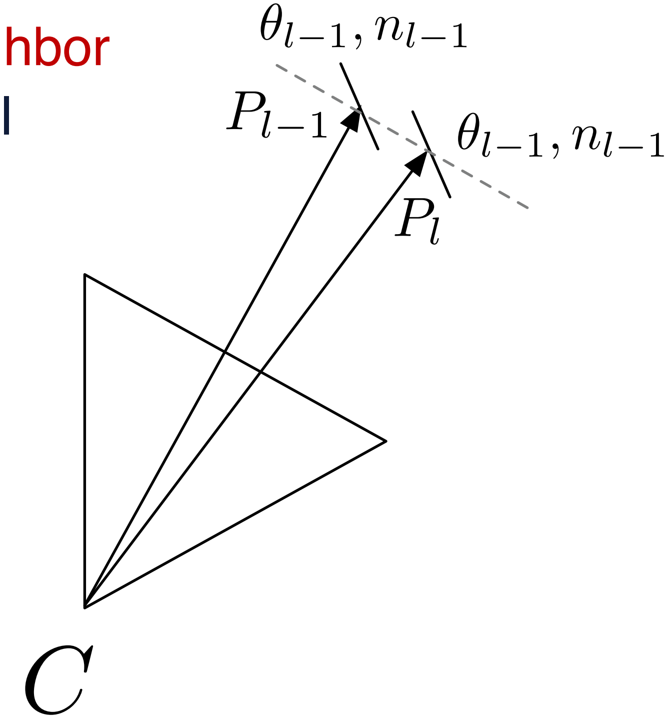


M. Bleyer, et. al. "**Patchmatch stereo - stereo matching with slanted support windows**", *BMVC 2011*

PatchMatch Stereo

- For Each Pixel
    - Assign Random Depth and Normal
- For N Iterations
    - For Each Pixel
        - Propagate Depth and Normal From Neighbor
        - Sample New Random Depth and Normal
        - Update Depth

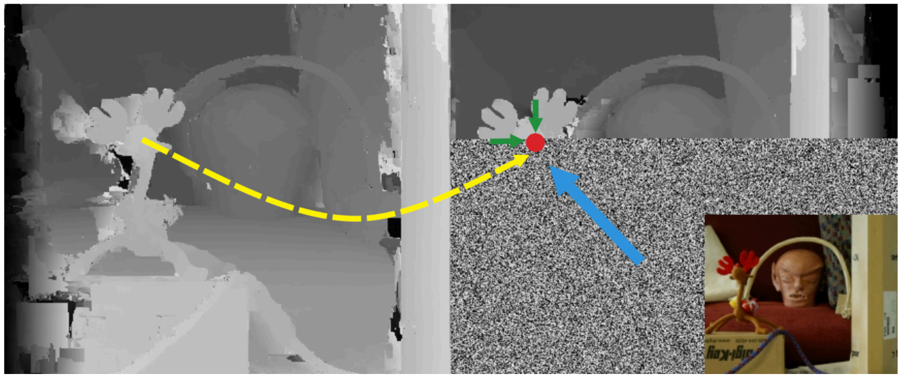


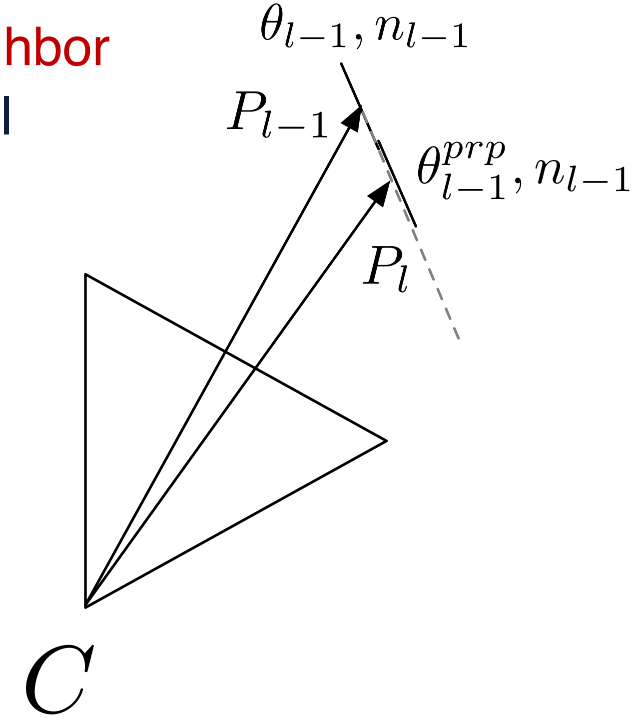


M. Bleyer, et. al. "**Patchmatch stereo - stereo matching with slanted support windows**", *BMVC 2011*

PatchMatch Stereo

- For Each Pixel
  - Assign Random Depth and Normal
- For N Iterations
  - For Each Pixel
    - Propagate Depth and Normal From Neighbor
    - Sample New Random Depth and Normal
    - Update Depth





J. L. Schönberger, et. al. "**Pixelwise View Selection for Unstructured Multi-View Stereo**", *ECCV 2016*

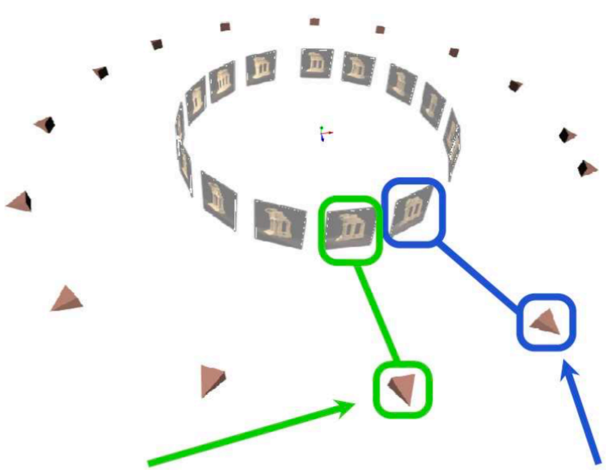
Summary

- Problem Definition:
  - Finding a “good” slanted support plane at each pixel.
- The difference with vanilla PatchMatch
  - (offset) -> (depth, normal)

View Selection

- Step 1: Source view selection
- Step 2: Depth-map computation
- Step 3: Depth-map merging

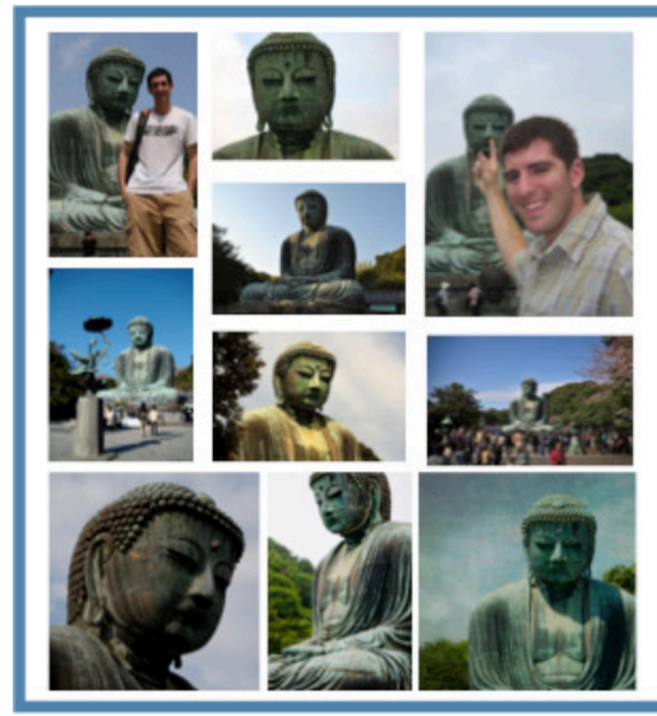
Key steps:
1.How to chose source images
2.How to compute depth map



View Selection

- *How to robustly integrate photo-consistency measurements from multiple views?*

Reference image

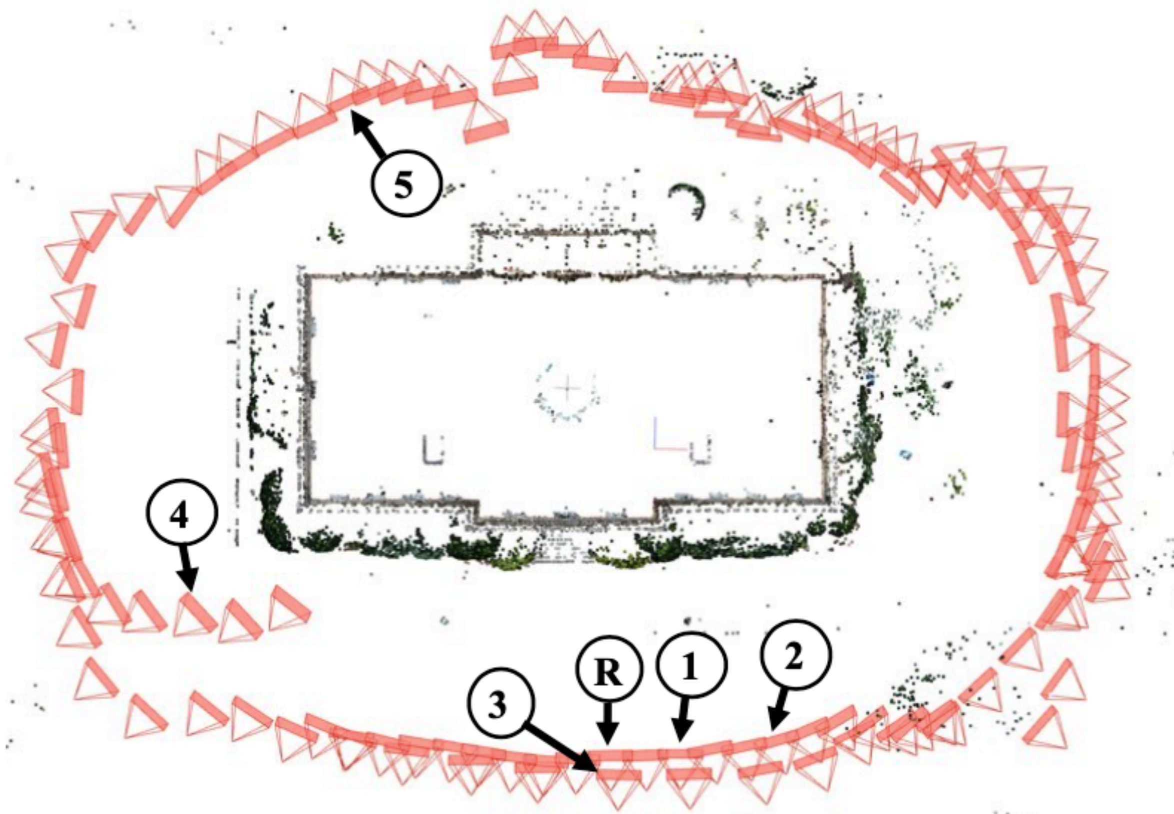


211 Source images (only 10 are shown)

Credit: E. Dunn

View Selection

- **Coarse visibility estimation via pose clustering**

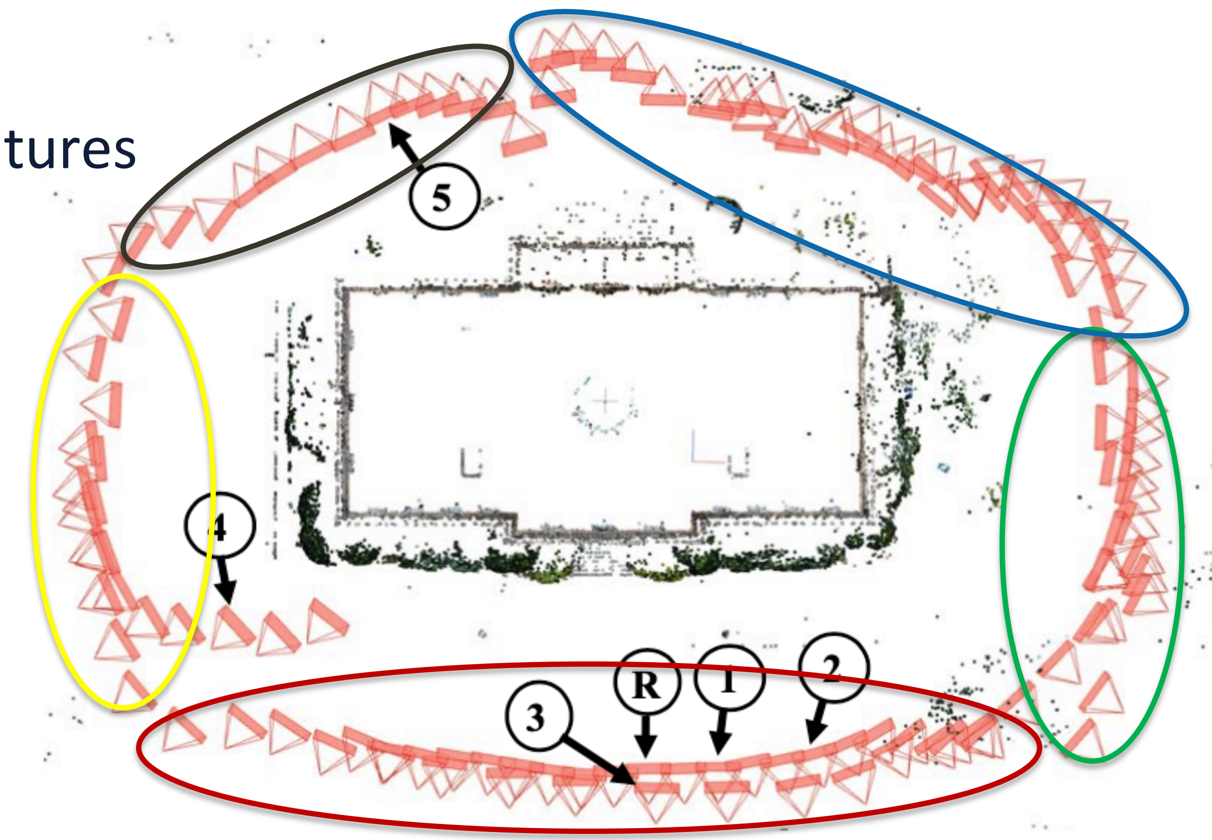


Y. Furukawa. “**Towards Internet-scale Multi-view Stereo**”, *CVPR 2010*

Credit: Schonberger

View Selection

- **Coarse visibility estimation via pose clustering**
  - Camera Proximity
  - Shared Sparse Features



Y. Furukawa. "**Towards Internet-scale Multi-view Stereo**", *CVPR 2010*

Credit: Schonberger

View Selection

- **Fine-scale visibility estimation**

- *Good* candidate source image ?

View Selection

- **Fine-scale visibility estimation**

- *Good* candidate source image ?

- Global
- -> a similar viewing direction as the target image
- -> a suitable baseline neither too short to degenerate the reconstruction accuracy nor too long to have less common coverage of the scene.

Shen. "**Accurate Multiple View 3D Reconstruction Using Patch Based Stereo for Large-Scale Scenes**", *TIP 2013*

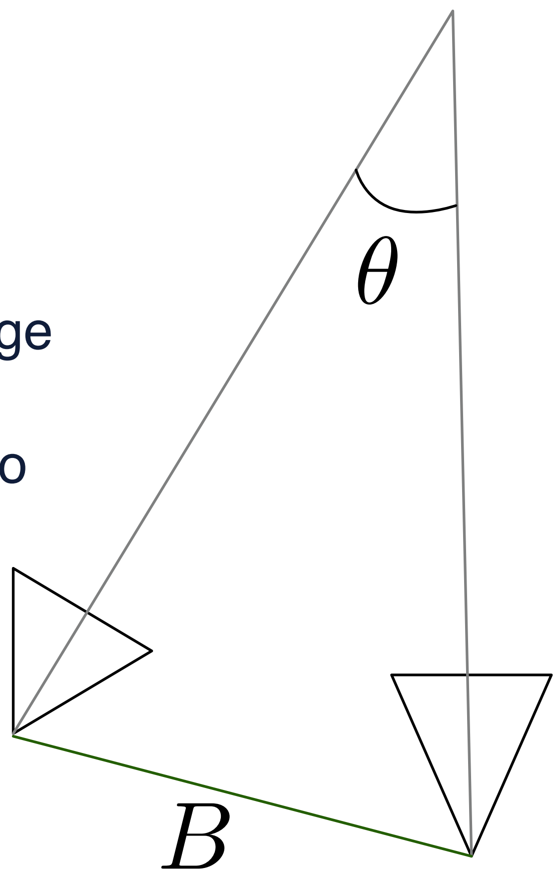
View Selection

- **Fine-scale visibility estimation**

- *Good* candidate source image ?

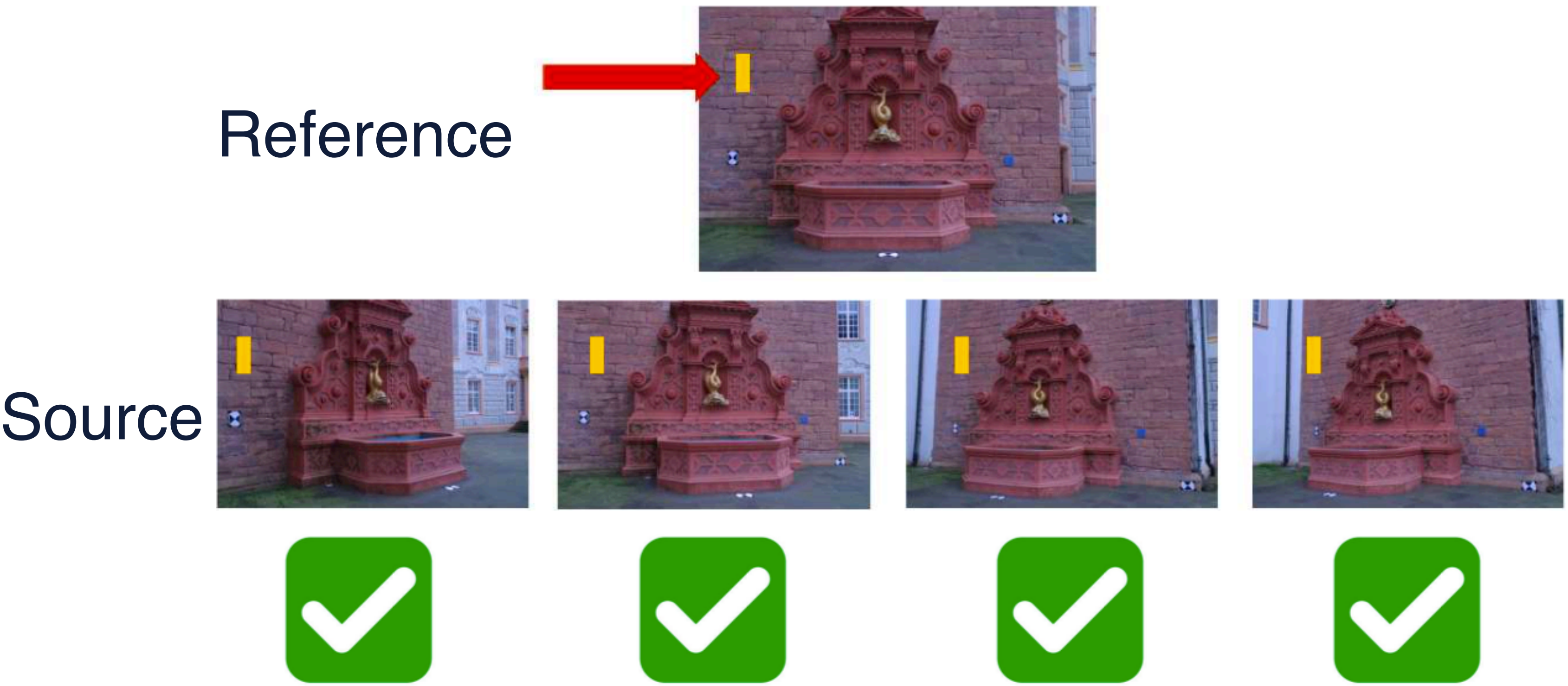
- Global
- -> a similar viewing direction as the target image
- -> a suitable baseline neither too short to degenerate the reconstruction accuracy nor too long to have less common coverage of the scene.

- $5° < \theta < 60°$
- $0.05d \leq B \leq 2d$



Shen. "**Accurate Multiple View 3D Reconstruction Using Patch Based Stereo for Large-Scale Scenes**", *TIP 2013*

Pixel-Level View Selection



Credit: E. Dunn

Pixel-Level View Selection

Reference

Source

Credit: E. Dunn

Pixel-Level View Selection

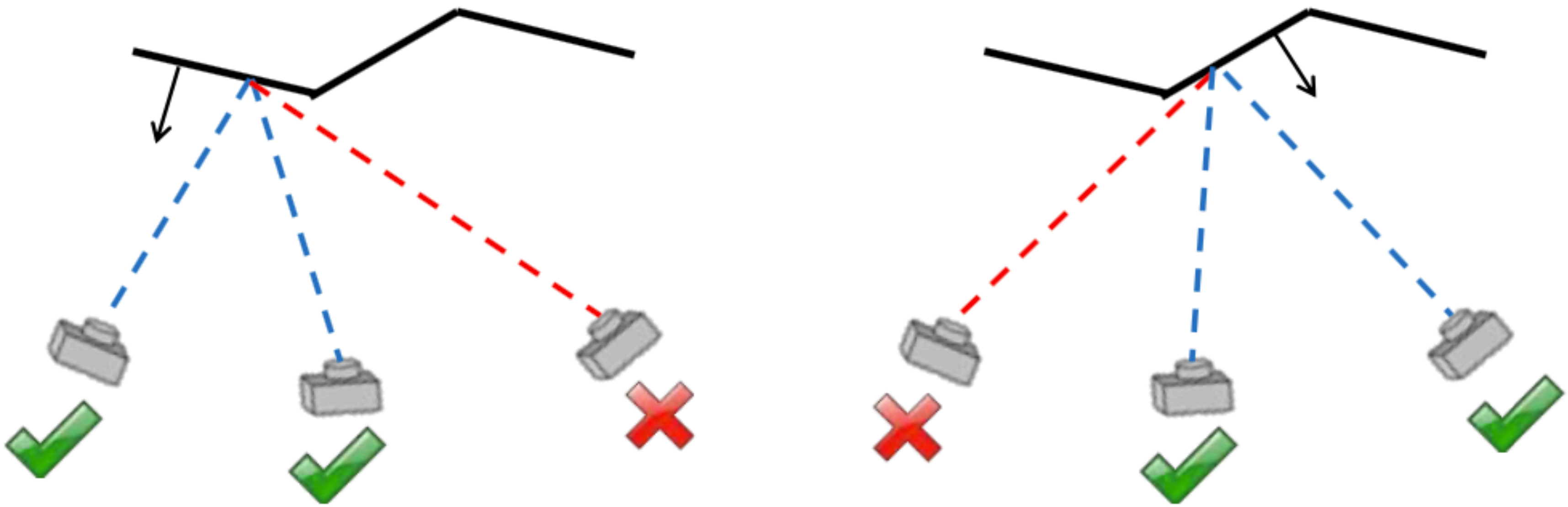
Reference

Source

Credit: E. Dunn

Image Selection vs Depth Estimation

- visibility requires scene structure and scene structure requires visibility
- **This is a chicken-and-egg problem**



Credit: S. Shen

Joint Pixel-Level View Selection and Depthmap Estimation

- **Maximum likelihood estimation (MLE):**
- **Likelihood function** $P(X, Z, \theta, N)$**:**
- **Images:**
  - $\boldsymbol{X = \{X^{ref}, X^{src}\}, X^{src} = \{X^m | m = 1 \dots M\}}$
- **Depth:**
  - $\boldsymbol{\theta = \{\theta_l | l = 1 \dots L\}}$
- **Normal:**
  - $\boldsymbol{N = \{n_l | l = 1 \dots L\}}$
- **Occlusion indicators:**
  - $\boldsymbol{Z = \{Z_l^m | l = 1 \dots L, m = 1 \dots M\}, Z_l^m \in \{0, 1\}}$

E. Zheng, et. al. “**Patchmatch based joint view selection and depthmap estimation**”, *CVPR 2014*

J. L. Schönberger, et. al. “**Pixelwise View Selection for Unstructured Multi-View Stereo**”, *ECCV 2016*

Joint Pixel-Level View Selection and Depthmap Estimation

- **Maximum likelihood estimation (MLE):**
- **Likelihood function** $P(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{N})$**:**

$$\prod_{l=1}^{L}\prod_{m=1}^{M}[P(Z_{l,t}^{m}|Z_{l-1,t}^{m}, Z_{l,t-1}^{m})P(X_{l}^{m}|Z_{l}^{m}, \theta_{l}, \boldsymbol{n}_{l})P(\theta_{l}, \boldsymbol{n}_{l}|\theta_{l}^{m}, \boldsymbol{n}_{l}^{m})]$$

- **Images:**
  - $\boldsymbol{X} = \{X^{ref}, \boldsymbol{X}^{src}\}, \boldsymbol{X}^{src} = \{\boldsymbol{X}^{\boldsymbol{m}} | \boldsymbol{m} = \mathbf{1} \ldots \boldsymbol{M}\}$
- **Depth:**
  - $\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\boldsymbol{l}} | \boldsymbol{l} = \mathbf{1} \ldots \boldsymbol{L}\}$
- **Normal:**
  - $\boldsymbol{N} = \{\boldsymbol{n}_{\boldsymbol{l}} | \boldsymbol{l} = \mathbf{1} \ldots \boldsymbol{L}\}$
- **Occlusion indicators:**
  - $\boldsymbol{Z} = \{\boldsymbol{Z}_{\boldsymbol{l}}^{\boldsymbol{m}} | \boldsymbol{l} = \mathbf{1} \ldots \boldsymbol{L}, \boldsymbol{m} = \mathbf{1} \ldots \boldsymbol{M}\}, \boldsymbol{Z}_{\boldsymbol{l}}^{\boldsymbol{m}} \in \{\mathbf{0}, \mathbf{1}\}$

E. Zheng, et. al. “**Patchmatch based joint view selection and depthmap estimation**”, *CVPR 2014*
J. L. Schönberger, et. al. “**Pixelwise View Selection for Unstructured Multi-View Stereo**”, *ECCV 2016*

Joint Pixel-Level View Selection and Depthmap Estimation

- **Maximum likelihood estimation (MLE):**
- **Likelihood function** $P(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{N})$**:**

$$\prod_{l=1}^{L}\prod_{m=1}^{M}[P(Z_{l,t}^m|Z_{l-1,t}^m, Z_{l,t-1}^m)\underbrace{P(X_l^m|Z_l^m, \theta_l, \boldsymbol{n}_l)}_{\textbf{\textit{Photometric prior}}}P(\theta_l, \boldsymbol{n}_l|\theta_l^m, \boldsymbol{n}_l^m)]$$

If $Z_l^m = 1$,

$P(X_l^m|Z_l^m, \theta_l, n_l) \propto \rho_l^m(\theta_l, n_l)$ (color similarity)

If $Z_l^m = 0$,

$P(X_l^m|Z_l^m, \theta_l, n_l) =$ uniform distribution

E. Zheng, et. al. "**Patchmatch based joint view selection and depthmap estimation**", *CVPR 2014*
J. L. Schönberger, et. al. "**Pixelwise View Selection for Unstructured Multi-View Stereo**", *ECCV 2016*

Joint Pixel-Level View Selection and Depthmap Estimation

- **Maximum likelihood estimation (MLE):**
- **Likelihood function** $P(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{N})$**:**

$$\prod_{l=1}^{L}\prod_{m=1}^{M}[\underbrace{P(Z_{l,t}^{m}|Z_{l-1,t}^{m}, Z_{l,t-1}^{m})}_{\text{Spatial-temporary smoothness}}P(X_{l}^{m}|Z_{l}^{m}, \theta_{l}, \boldsymbol{n}_{l})P(\theta_{l}, \boldsymbol{n}_{l}|\theta_{l}^{m}, \boldsymbol{n}_{l}^{m})]$$

$$P(Z_{l,t}^{m}|Z_{l-1,t}^{m}, Z_{l,t-1}^{m}) = P(Z_{l,t}^{m}|Z_{l-1,t}^{m})P(Z_{l,t}^{m}|Z_{l,t-1}^{m}).$$

$$P(Z_{l}^{m}|Z_{l-1}^{m}) = \begin{pmatrix} \gamma & 1-\gamma \\ 1-\gamma & \gamma \end{pmatrix}.$$

$$P(Z_{l,t}^{m}|Z_{l,t-1}^{m}) = \begin{pmatrix} \lambda_t & 1-\lambda_t \\ 1-\lambda_t & \lambda_t \end{pmatrix}$$

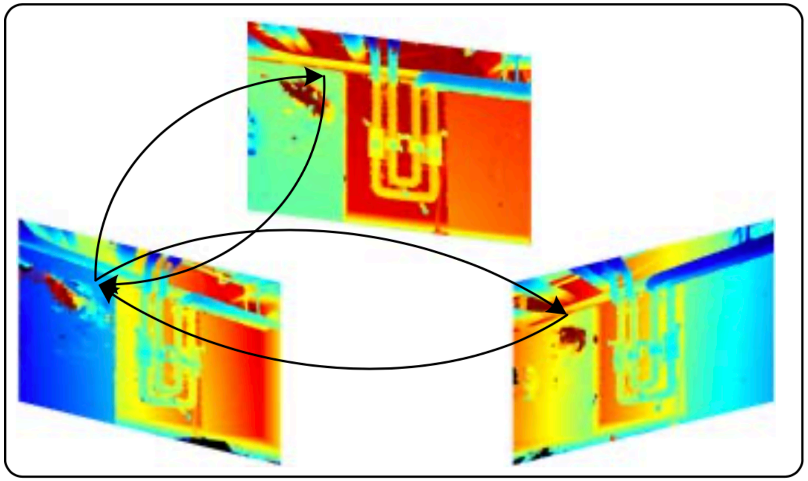
E. Zheng, et. al. "**Patchmatch based joint view selection and depthmap estimation**", *CVPR 2014*
J. L. Schönberger, et. al. "**Pixelwise View Selection for Unstructured Multi-View Stereo**", *ECCV 2016*

Joint Pixel-Level View Selection and Depthmap Estimation

- **Maximum likelihood estimation (MLE):**
- **Likelihood function** $P(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{N})$**:**

$$\prod_{l=1}^{L} \prod_{m=1}^{M} [P(Z_{l,t}^{m} | Z_{l-1,t}^{m}, Z_{l,t-1}^{m}) P(X_{l}^{m} | Z_{l}^{m}, \theta_{l}, \boldsymbol{n}_{l}) \underbrace{P(\theta_{l}, \boldsymbol{n}_{l} | \theta_{l}^{m}, \boldsymbol{n}_{l}^{m})]}_{\textbf{geometric consistency}}$$

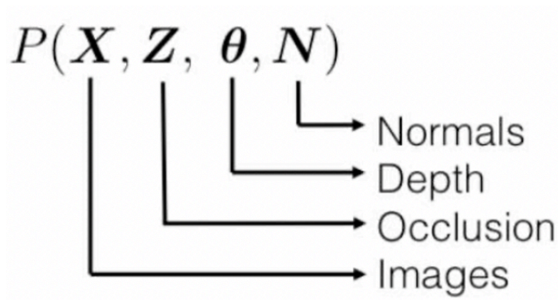
Forward-backward reprojection error



E. Zheng, et. al. "**Patchmatch based joint view selection and depthmap estimation**", *CVPR 2014*
J. L. Schönberger, et. al. "**Pixelwise View Selection for Unstructured Multi-View Stereo**", *ECCV 2016*

Joint Pixel-Level View Selection and Depthmap Estimation

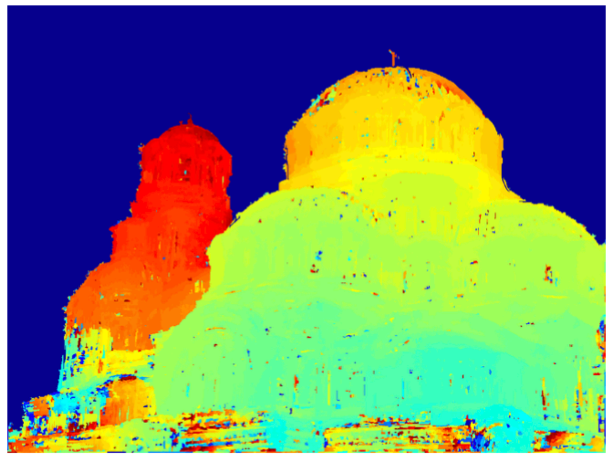
- **Joint likelihood Estimation:**



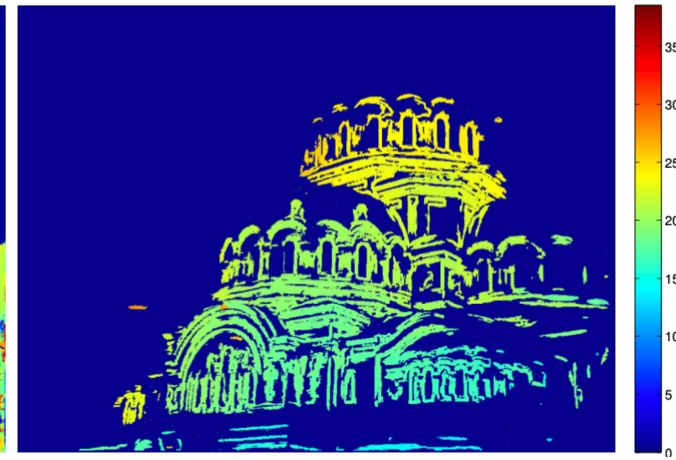
- Generalized Expectation Maximization
  - E-Step
    - Infer Z using variational inference
  - M-Step
    - Infer θ, N using PatchMatch sampling

E. Zheng, et. al. "**Patchmatch based joint view selection and depthmap estimation**", *CVPR 2014*
J. L. Schönberger, et. al. "**Pixelwise View Selection for Unstructured Multi-View Stereo**", *ECCV 2016*

Robustness of Pixel-Level Selection



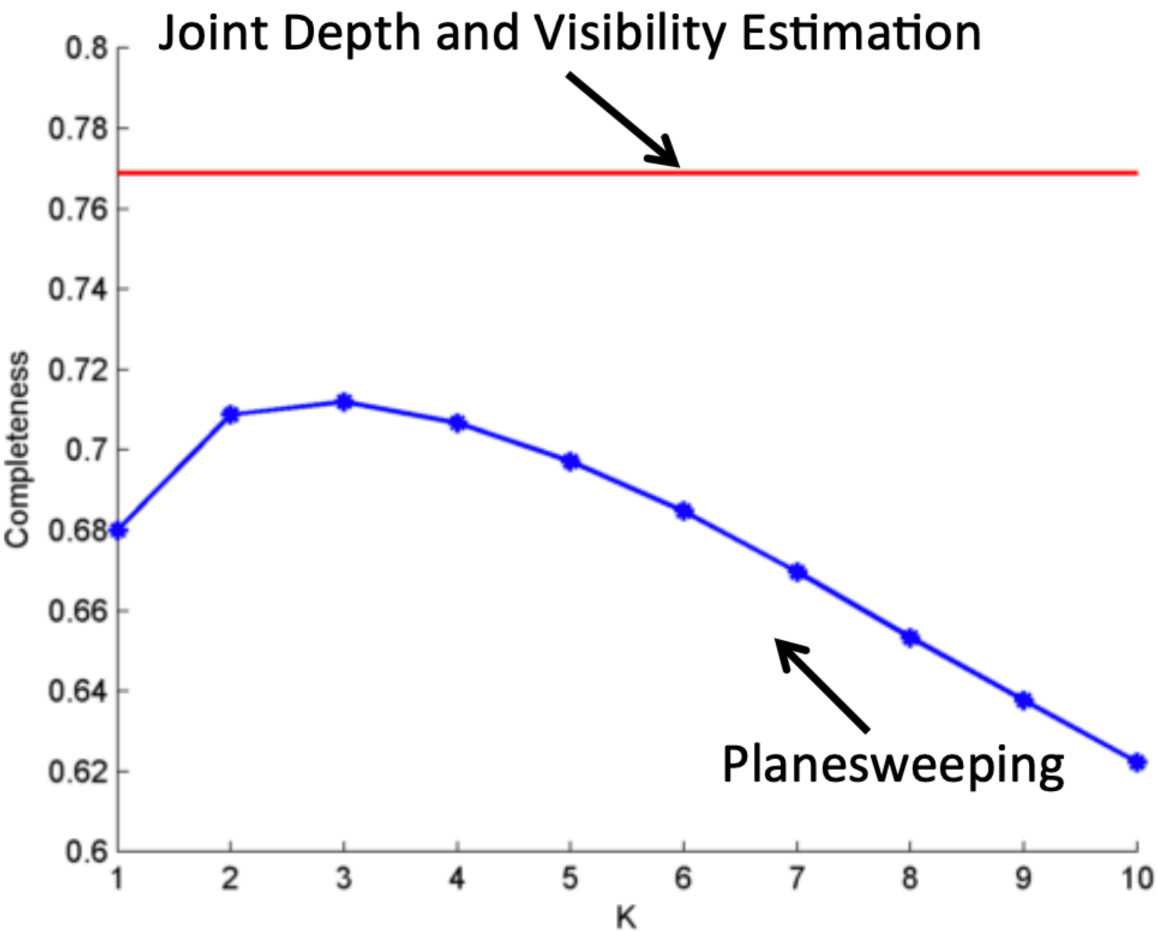
Pixel-level

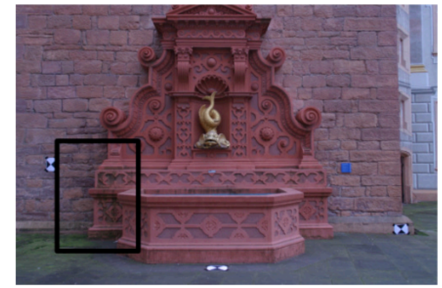


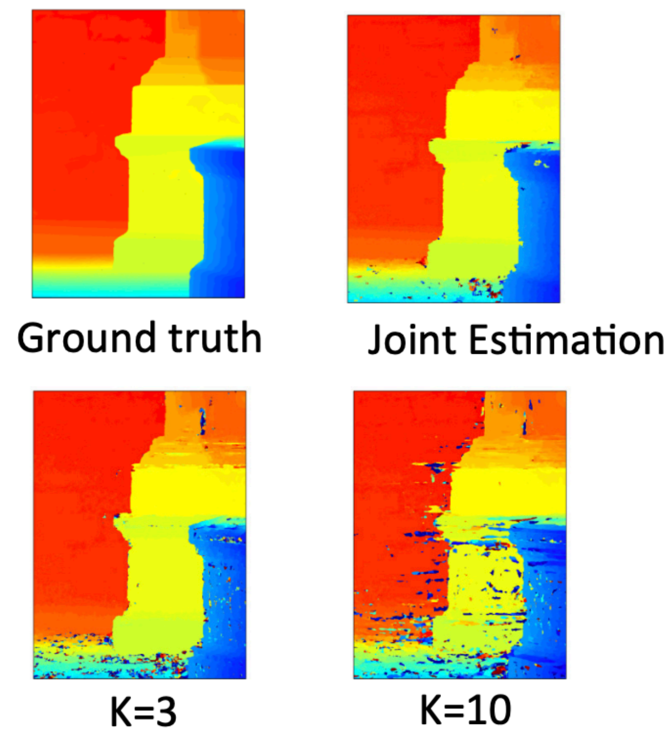
Baseline

E. Zheng, et. al. “**Patchmatch based joint view selection and depthmap estimation**”, *CVPR 2014*

Robustness of Pixel-Level Selection







Completeness: percentage of pixels with errors less than 2 cm

E. Zheng, et. al. "**Patchmatch based joint view selection and depthmap estimation**", *CVPR 2014*

Summary

- **PatchMatch**
    - A <u>randomized</u> algorithm for rapidly finding correspondences between image patches
    - Step
        - 1: Initialization
        - 2: Propagation
        - 3: Random Search
- **PatchMatch Stereo**
    - Finding a “good” slanted support plane at each pixel.
    - Difference from vanilla PatchMatch
        - (offset) -> (depth, normal)
- **View Selection**
    - Coarse visibility estimation
    - Fine-scale visibility estimation
    - Joint Pixel-Level View Selection and Depthmap Estimation
        - Pixel-Level occlusion indicator
        - chicken-and-egg -> Generalized Expectation maximization

Thanks