PatchMatch

Sanketh Vedula
sanketh@cs.technion.ac.il

Agenda

- PatchMatch
  - Key ideas
  - Implementation
- Applications
- Image retargeting
  - Seam carving
  - Bi-directional similarity

PatchMatch

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

Connelly Barnes[1] Eli Shechtman[2,3] Adam Finkelstein[1] Dan B Goldman[2]
[1]Princeton University [2]Adobe Systems [3]University of Washington

(a) original (b) hole+constraints (c) hole filled (d) constraints (e) constrained retarget (f) reshuffle

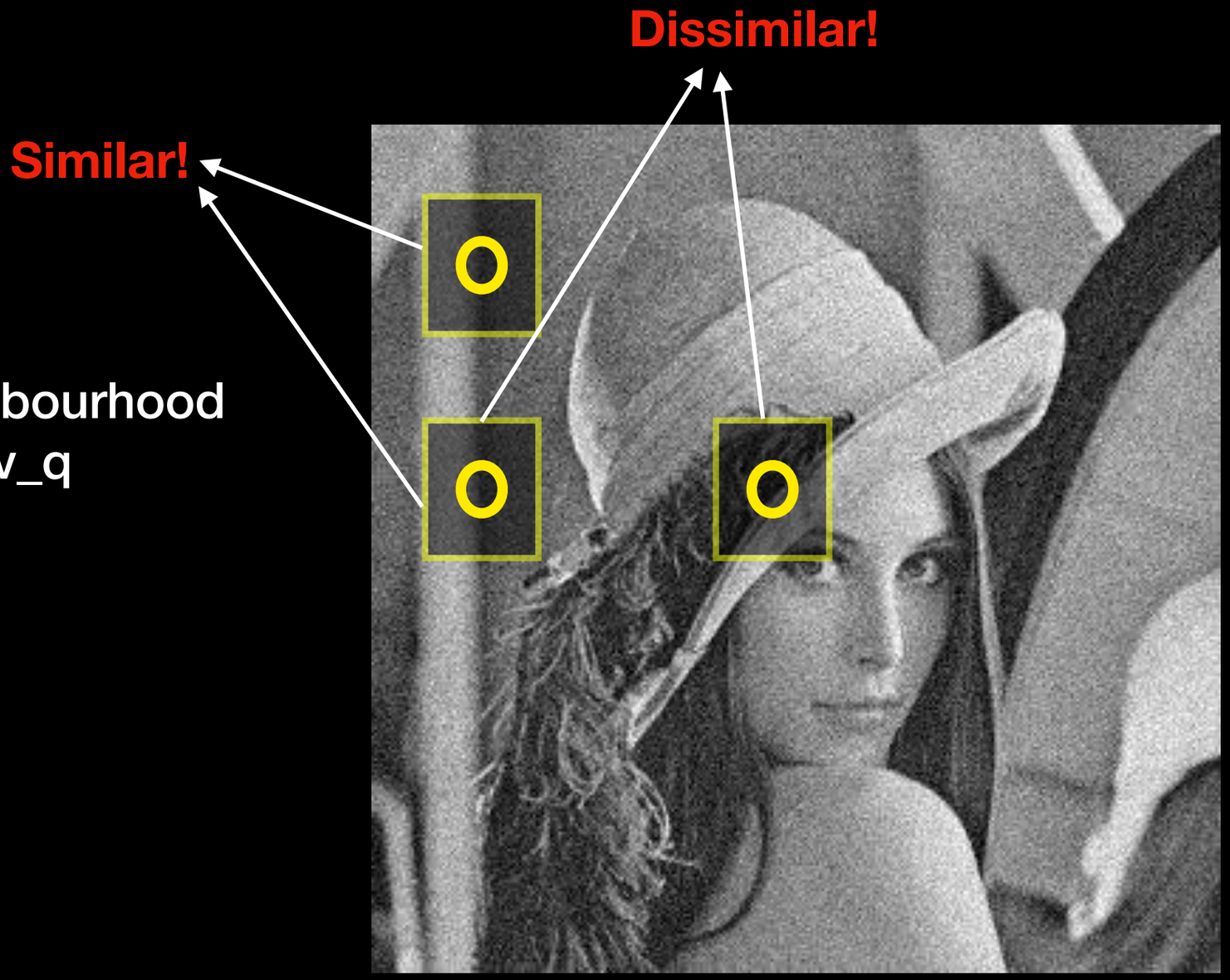
- Find dense (pixel-level) correspondence between two images
- Why do we care?
- Many applications — denoising, depth reconstruction from stereo, retargeting, inpainting and many more.

Previously: matching patches

1. **Naive search**

for every pixel p

- define a small neighbourhood
- define vectors v_p, v_q



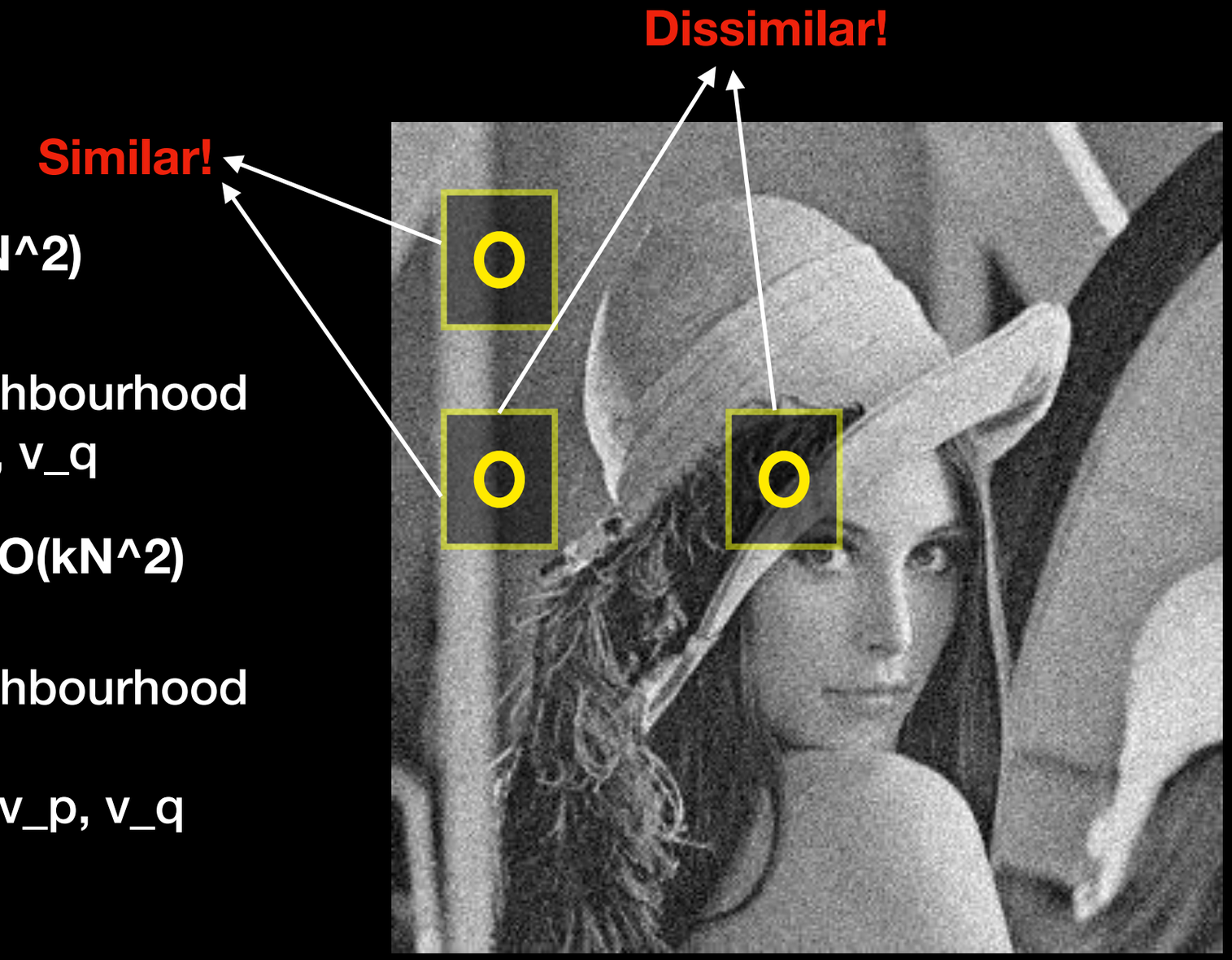
Previously: matching patches

1. **Naive search O(mN^2)**

   for every pixel p
   - define a small neighbourhood
   - define vectors v_p, v_q

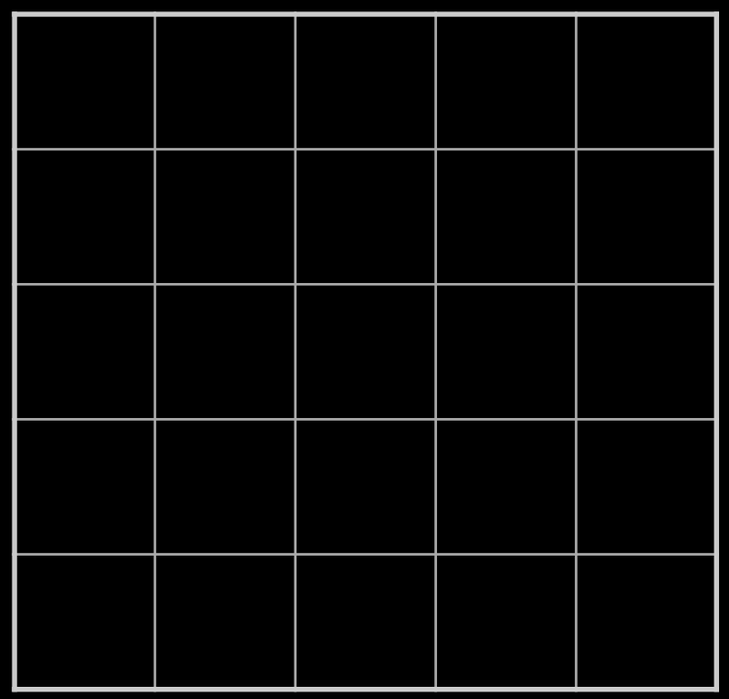
2. **Nearest neighbour O(kN^2)**

   for every pixel p
   - define a small neighbourhood
   - Take a PCA of it
   - Truncated vectors v_p, v_q
   - Match them

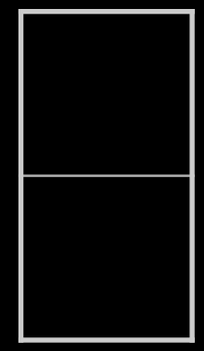


Key ideas

Patches **vs.** **Patch offsets**



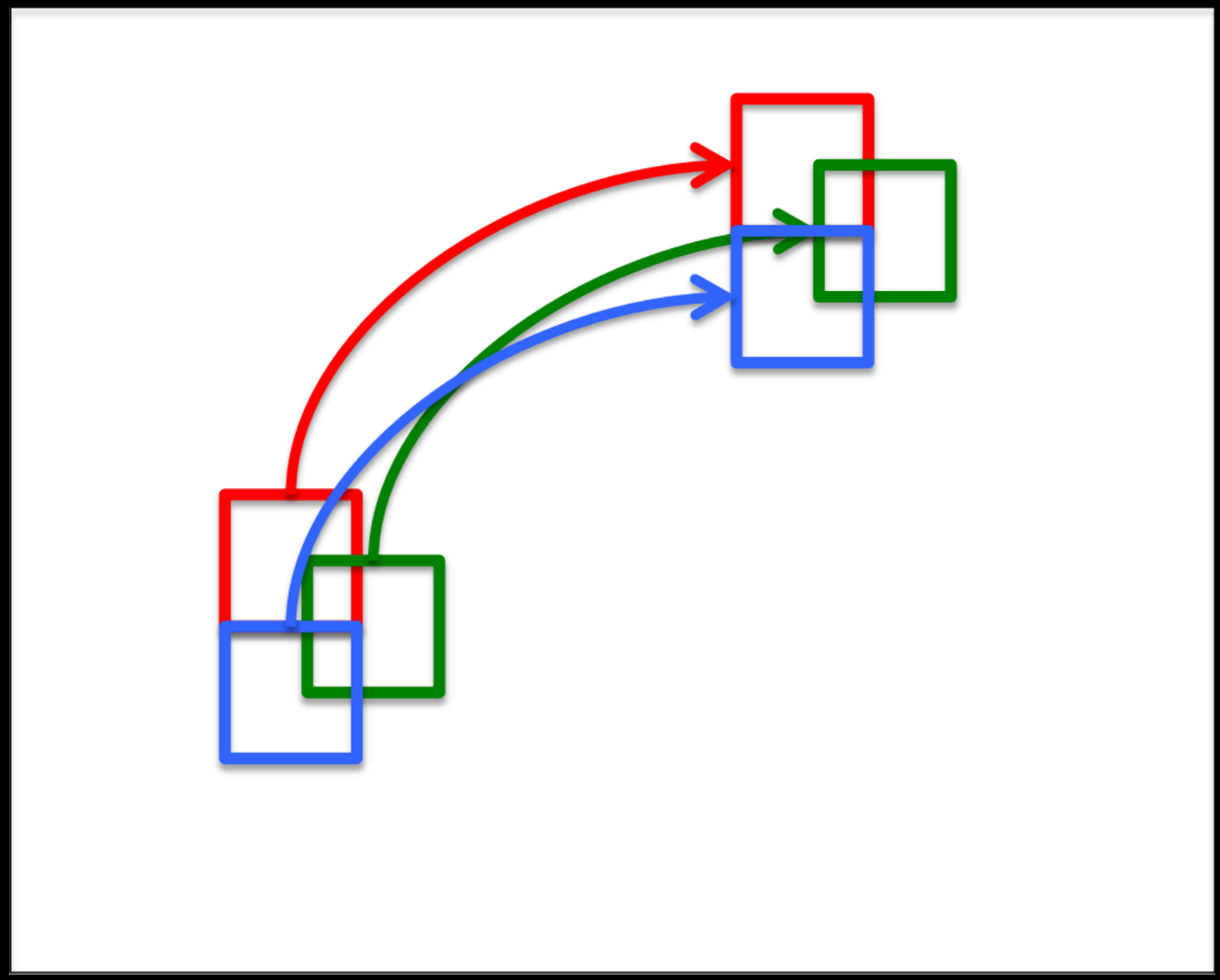
Patches of size 25



Patch offsets of size 2

Key ideas

Coherent matches with neighbours



Key ideas

M — total number of pixels

Probability of correct random guess: 1/M

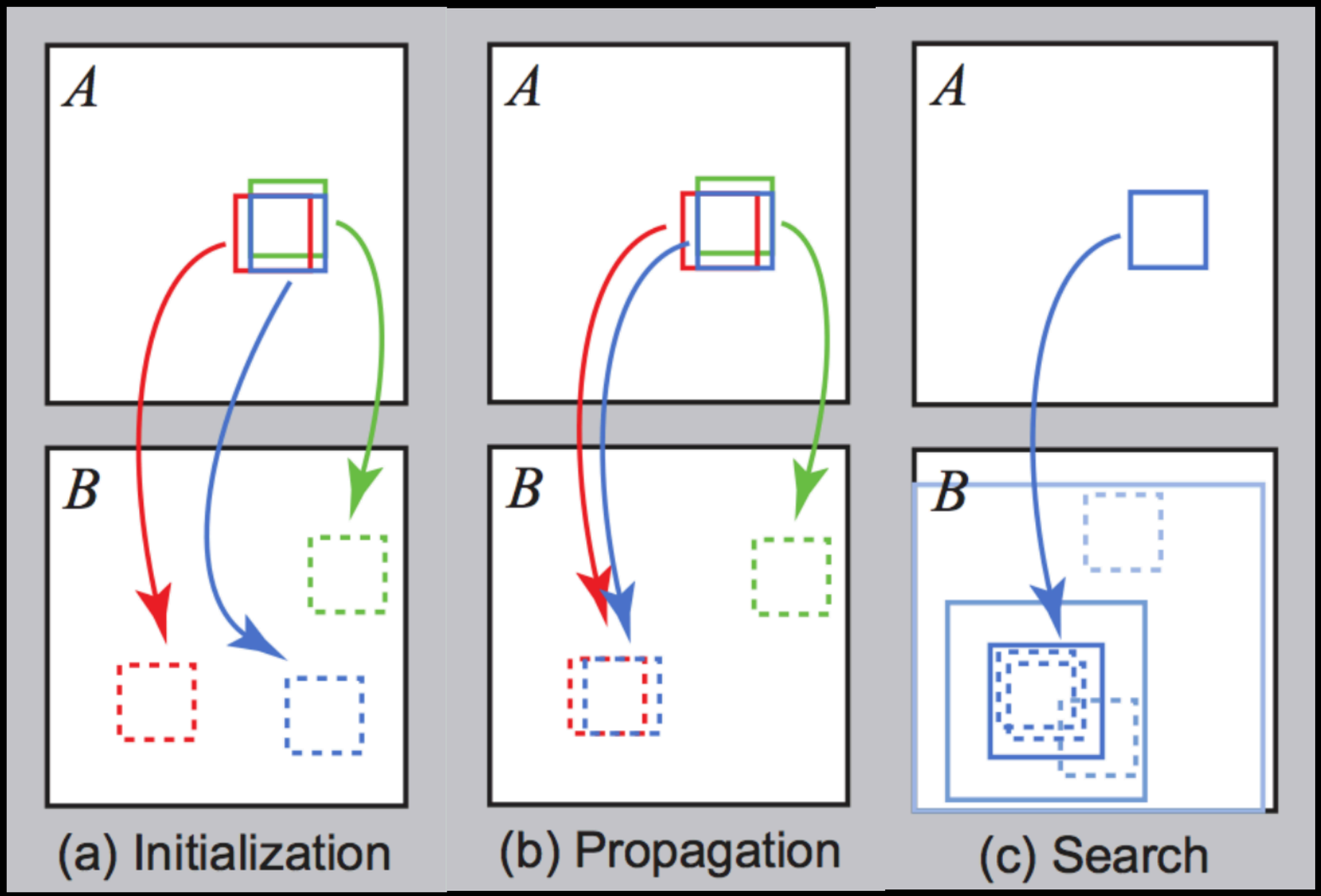
Probability of incorrect random guess: 1 - 1/M

Probability of all pixels with incorrect random guess: (1-1/M)^M (~0.37)

=> Probability of at least one pixel with a correct guess: 1 - (1-1/M)^M

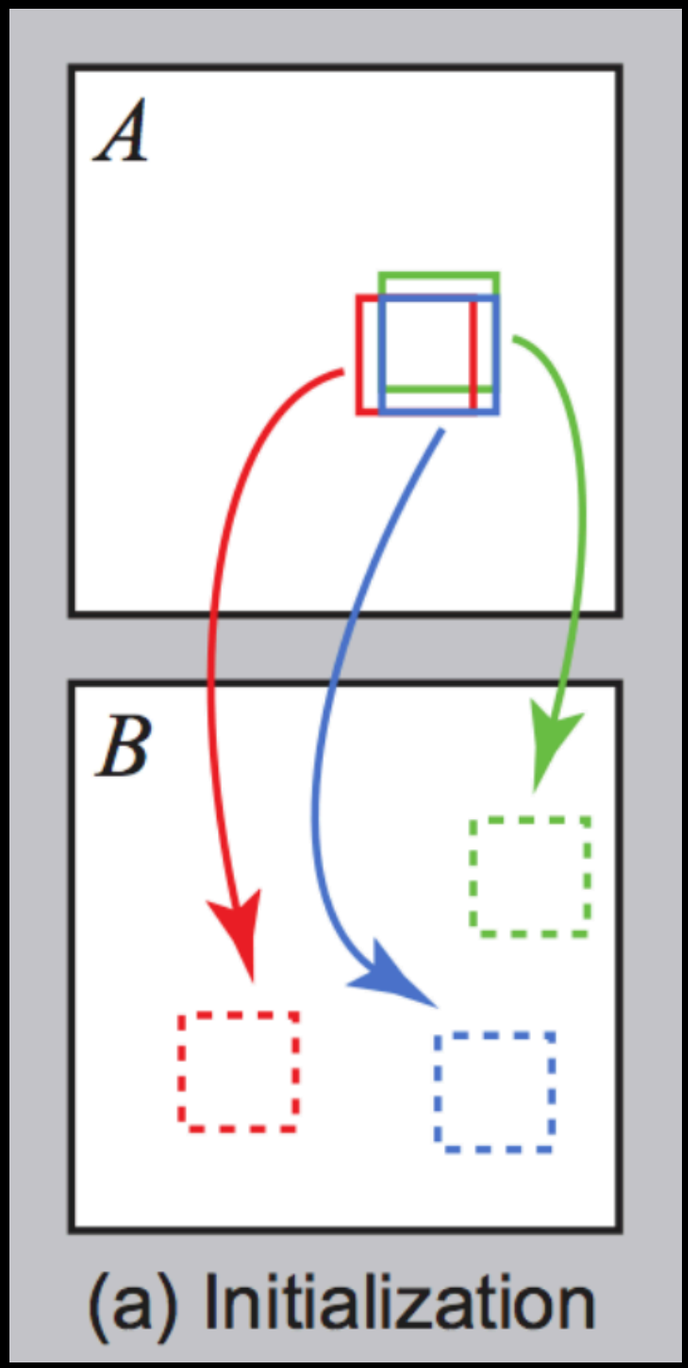
=> Probability of at least one pixel with a good enough guess: 1-(1-C/M)^M

Algorithm: 3 steps



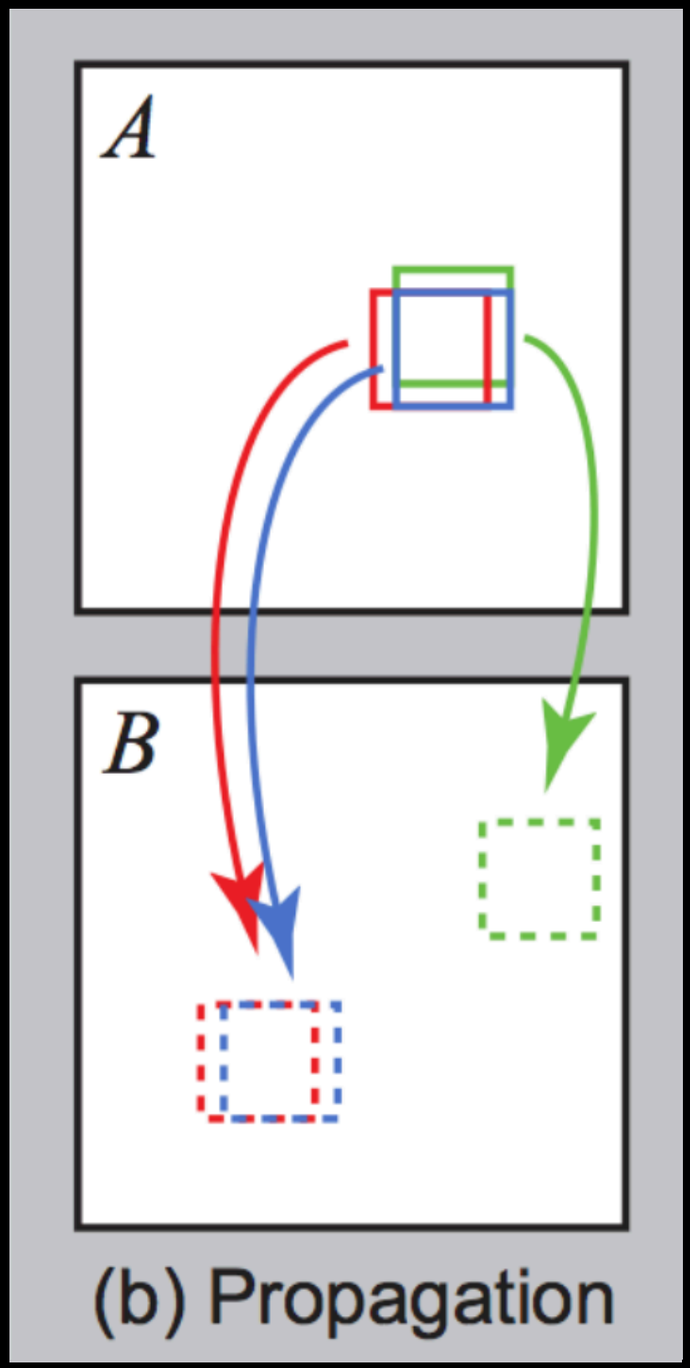
Step-1: Initialisation

Each pixel is given a random offset patch as initialisation



Step-2: Propagation

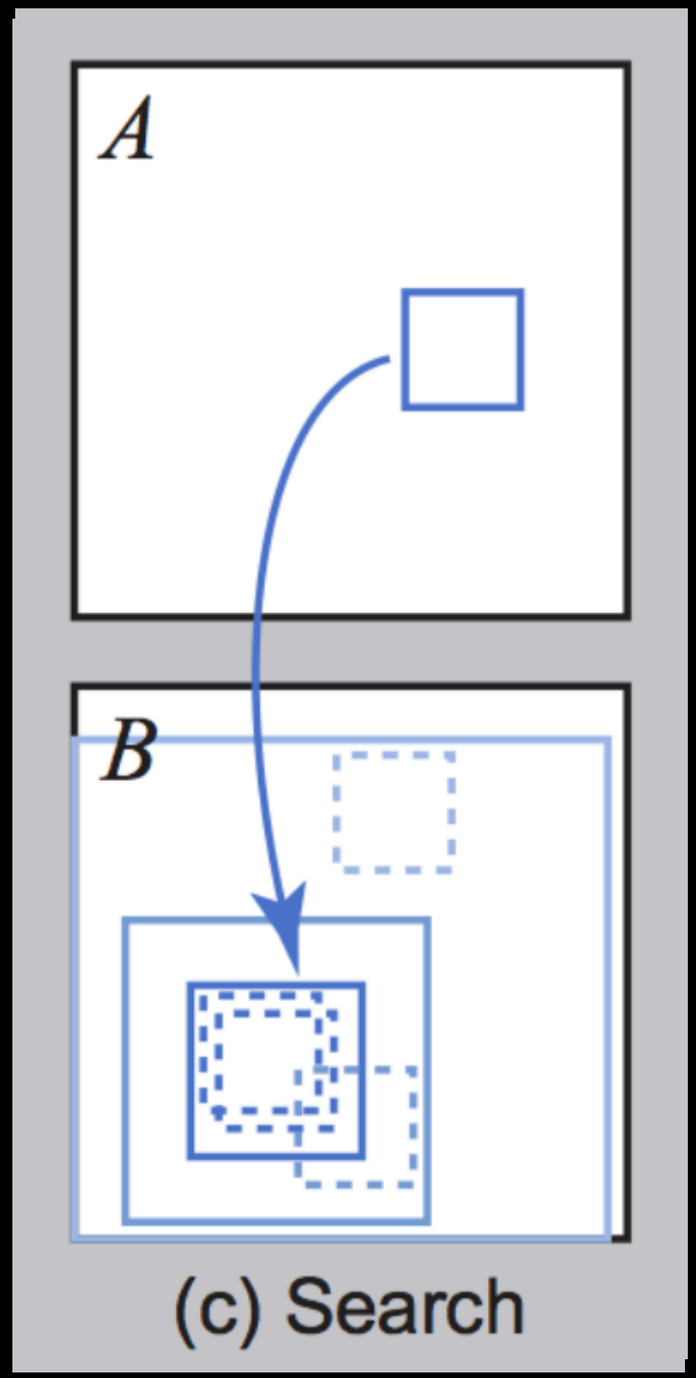
Each pixel checks if the offsets from neighbouring patches give a better matching patch. If so, adopt according to neighbours' offset



Step 3: search

Each pixel searches for better patch offsets within a concentric radius around the current offset

The search radius starts with the size of the image and is halved each time until it is 1



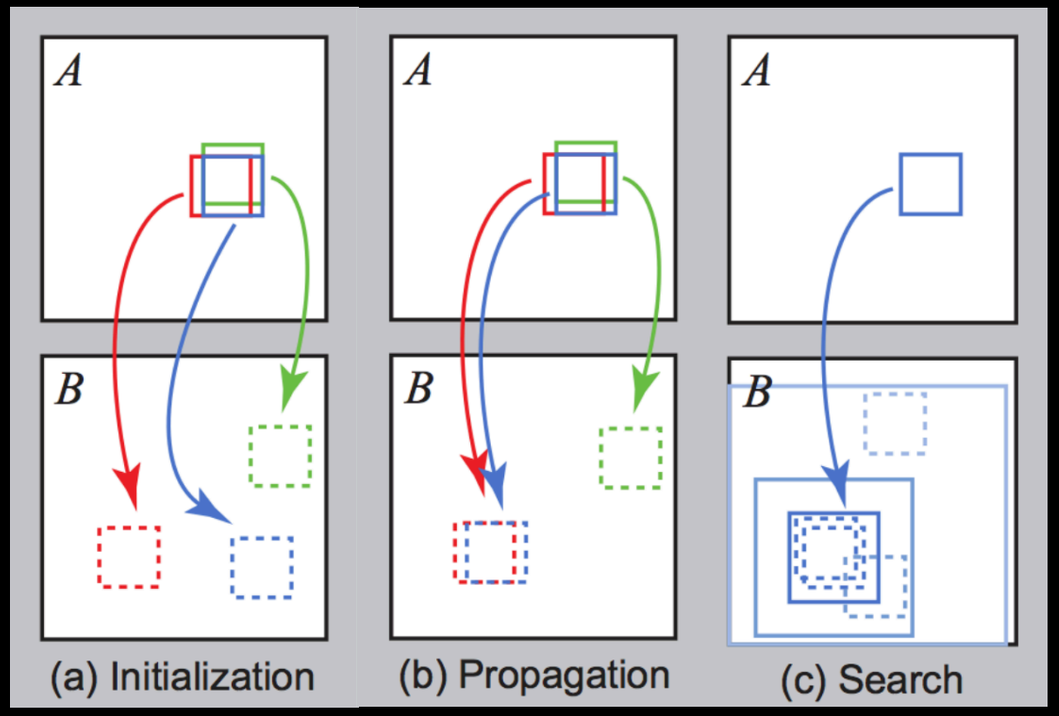
Algorithm

1. Initialize pixels with random patch offsets

2. Check if neighbours have better patches

3. Search in concentric radius around the offset for better patch offsets

4. Go to step 2 until convergence (~5 iterations)



O(mNlogN)

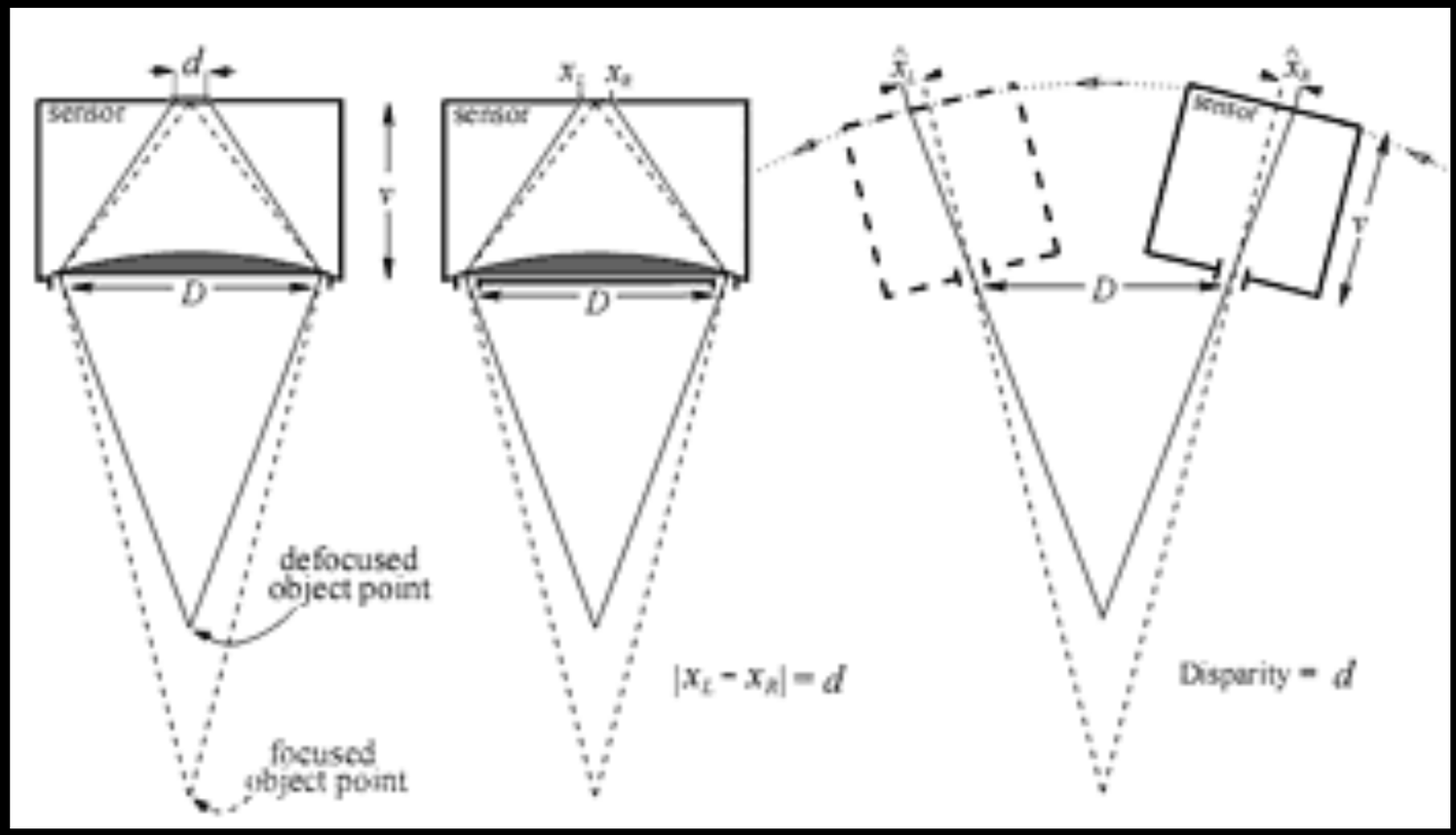
Megapixels	Time [s] Ours	Time [s] *k*d-tree	Memory [MB] Ours	Memory [MB] *k*d-tree
0.1	0.68	15.2	1.7	33.9
0.2	1.54	37.2	3.4	68.9
0.35	2.65	87.7	5.6	118.3

Table 1: *Running time and memory comparison for the input shown in Figure 3. We compare our algorithm against a method commonly used for patch-based search: kd-tree with approximate nearest neighbor matching. Our algorithm uses $n = 5$ iterations. The parameters for kd-tree have been adjusted to provide equal mean error to our algorithm.*

(c) PatchMatch, SIGGRAPH 2009

Implementation

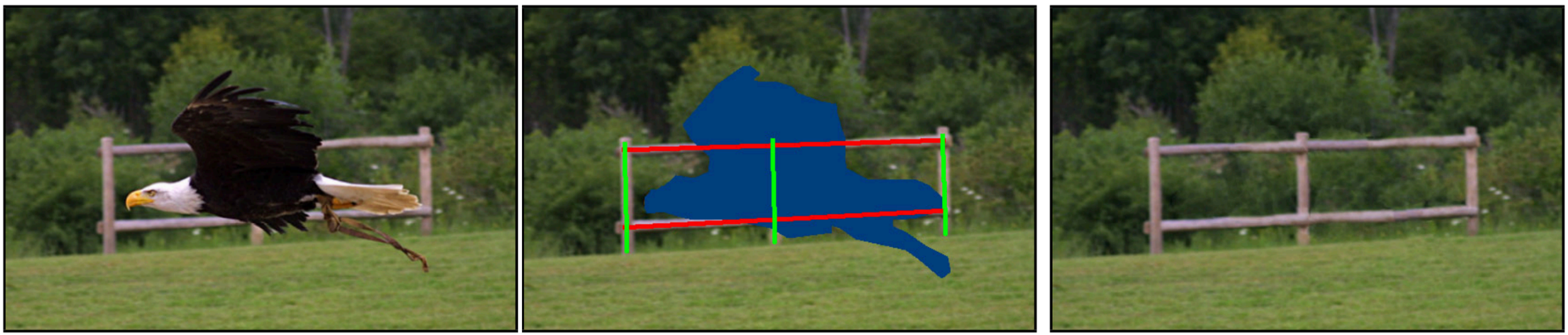
- Dense pixel correspondence for stereo images
- Semantic inpainting



(c) PatchMatch, SIGGRAPH 2009, Yoav Schechner (figure)

Constrained inpainting

Guided search:



Guiding to replace green lines with patches along green lines and to inpaint the blue image from the rest of the pixels

Image retargeting

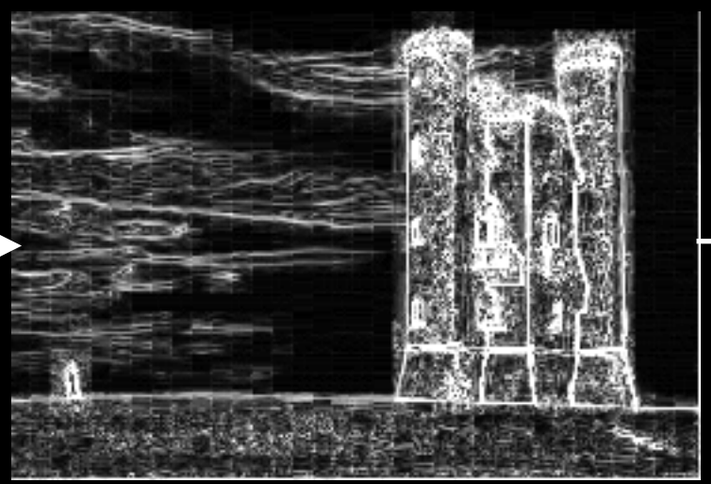
(c) Seam Carving, SIGGRAPH 2007

Problem invented by Setlur et al., 2005

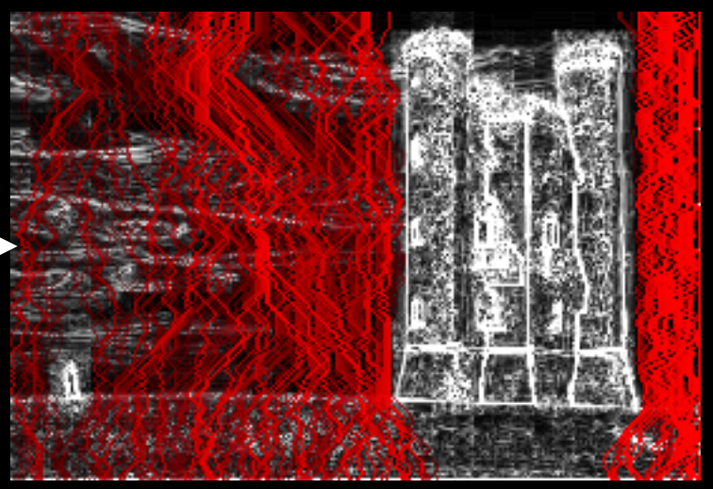
https://github.com/vivianhylee/seam-carving

Seam carving

Get an energy map, e.g. gradients, entropy, visual saliency etc.



Generate “seams” — minimum energy path to reach from one end of the image to the other



Seams are computed efficiently using greedy methods, Dijkstra’s, dynamic programming, etc.

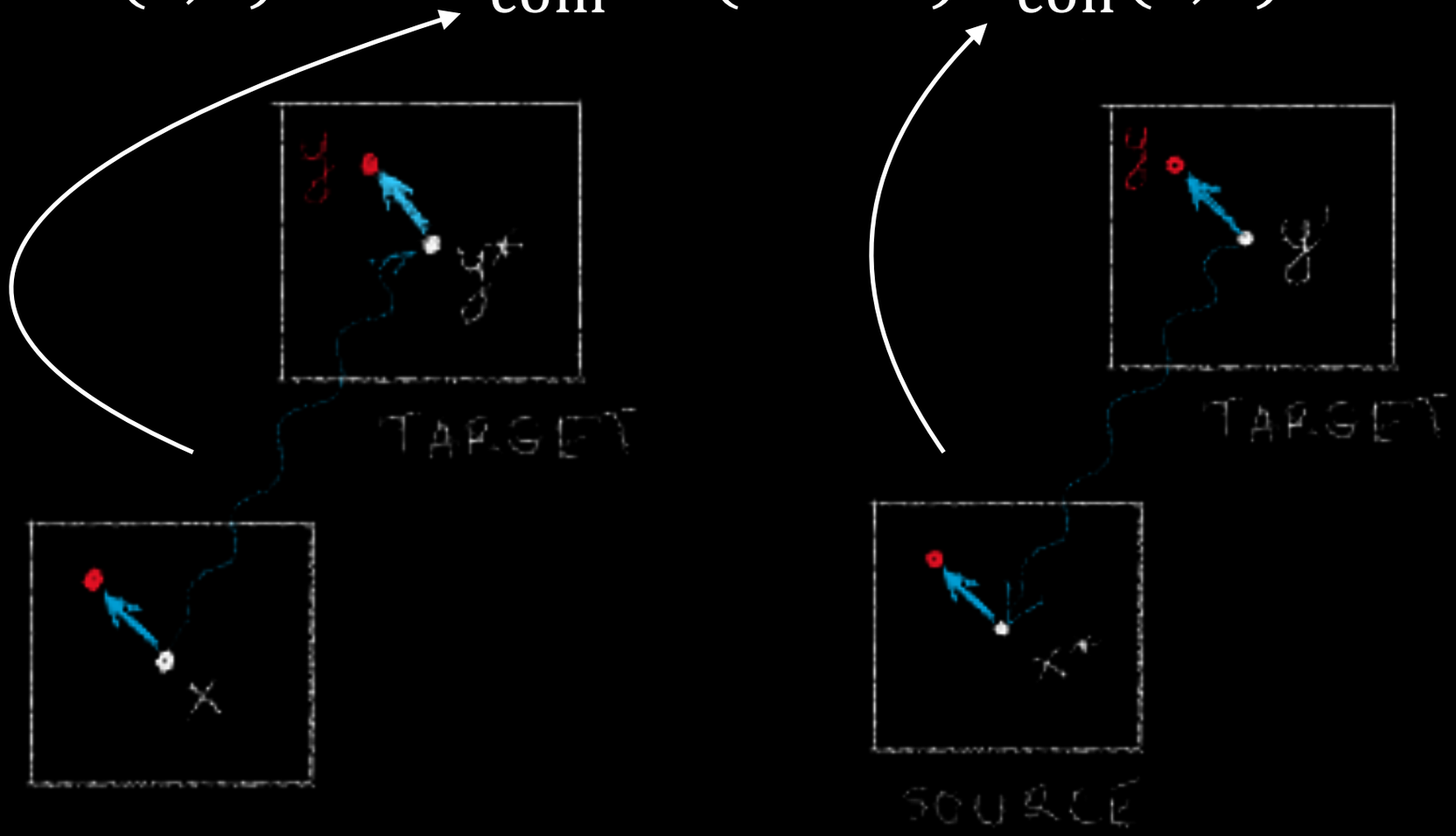
Rank the seams according to energy and remove the them in an ascending order of energy

(c) Seam Carving, SIGGRAPH 2007

Bi-directional similarity

Energy:

$$D(s,t) = \alpha D_{\text{com}} + (1-\alpha) D_{\text{coh}}(s,t) \qquad \alpha \in (0,1)$$



Find t that minimises the above energy

$$t = \arg \min_{t:\Omega' \to \mathbb{R}} D(s,t)$$

Image retargeting

For $k = 1, \ldots,$ until convergence

- Initialize $t^{k+1} \equiv 0 \quad c \equiv 0$
- For each $\mathbf{y} \in \Omega'$
  - Find $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} \left\| \pi_{\mathbf{x}} s - \pi_{\mathbf{y}} t \right\|_2^2$
  - $t^{k+1}|_{\mathbf{y}+\blacksquare} \leftarrow t^{k+1}|_{\mathbf{y}+\blacksquare} + \frac{1-\alpha}{|\Omega'|} \pi_{\mathbf{x}^*} s$
  - $c|_{\mathbf{y}+\blacksquare} \leftarrow c|_{\mathbf{y}+\blacksquare} + \frac{1-\alpha}{|\Omega'|}$

PatchMatch

- For each $\mathbf{x} \in \Omega$
  - Find $\mathbf{y}^* = \arg\min_{\mathbf{y}\in\Omega'} \left\| \pi_{\mathbf{x}} s - \pi_{\mathbf{y}} t \right\|_2^2$
  - $t^{k+1}|_{\mathbf{y}^*+\blacksquare} \leftarrow t^{k+1}|_{\mathbf{y}^*+\blacksquare} + \frac{\alpha}{|\Omega|} \pi_{\mathbf{x}^*} s$
  - $c|_{\mathbf{y}^*+\blacksquare} \leftarrow c|_{\mathbf{y}^*+\blacksquare} + \frac{1}{|\Omega|}$

- Normalize $t^{k+1} \leftarrow t^{k+1}/c$

Image retargeting: in code

Until we reach the target aspect ratio

Until convergence:

- PatchMatch s -> t (completeness)
- PatchMatch t -> s (coherence)
- Get votes for each pixel w.r.to coherence and completeness
- Average them appropriately
- Decrease the scale by a bit and iterate

Geometric constraints

(a) original

(b) retargeted

(c) with constraints

Summary

- PatchMatch is a very powerful tool. Used “extensively” in many tools such as Photoshop CS6
- Can run in real-time when implemented appropriately
- Stereo depth estimation, inpainting, image and video retargeting (in internet applications) and many more
- Many modifications are explored (1400+ citations till date)

https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/index.php