Tutorial 9: PatchMatch

Digital Image Processing (236860)

Agenda

- PatchMatch
  - Key ideas
  - Algorithm
  - Extensions
- Applications

PatchMatch

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

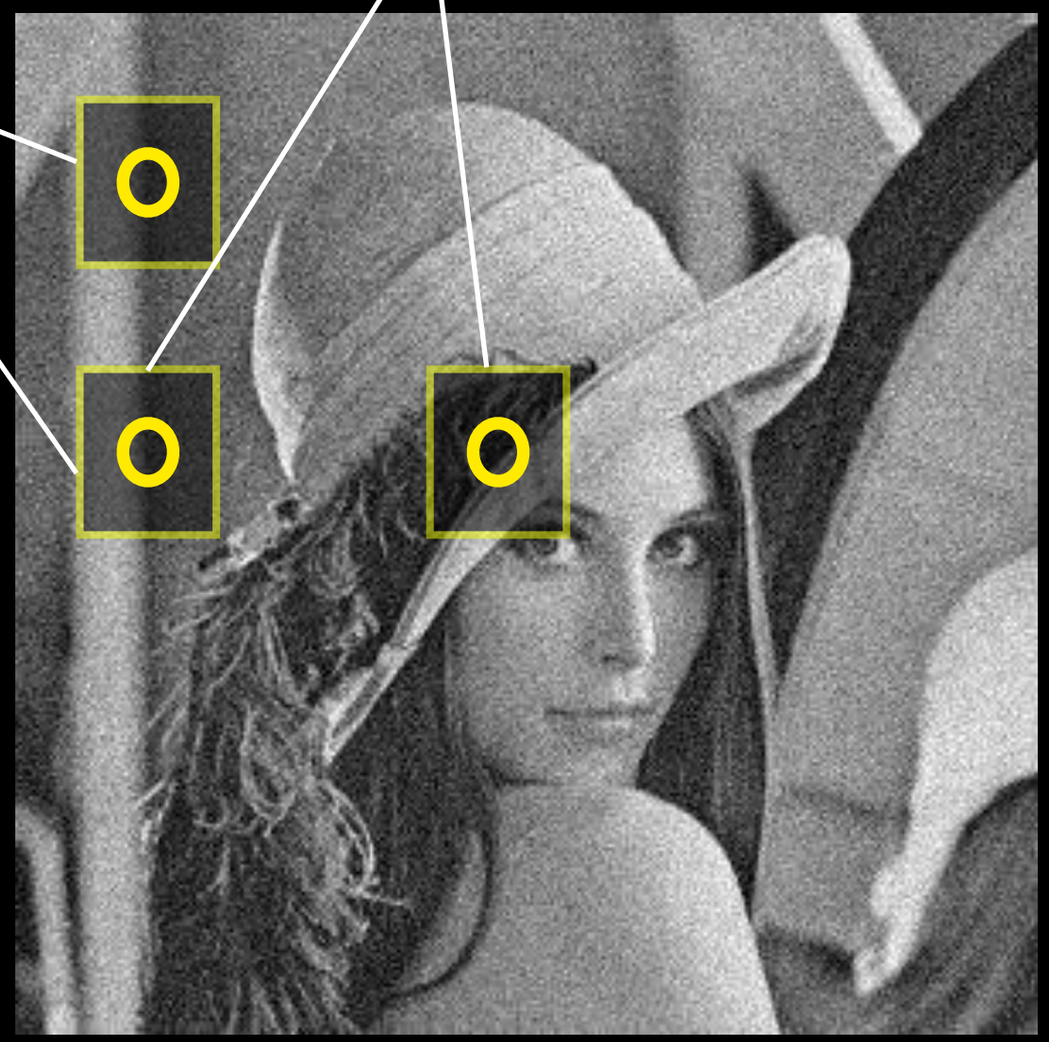
Connelly Barnes[1] Eli Shechtman[2,3] Adam Finkelstein[1] Dan B Goldman[2]
[1]Princeton University [2]Adobe Systems [3]University of Washington

(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 such as denoising, depth reconstruction from stereo, retargeting, inpainting…

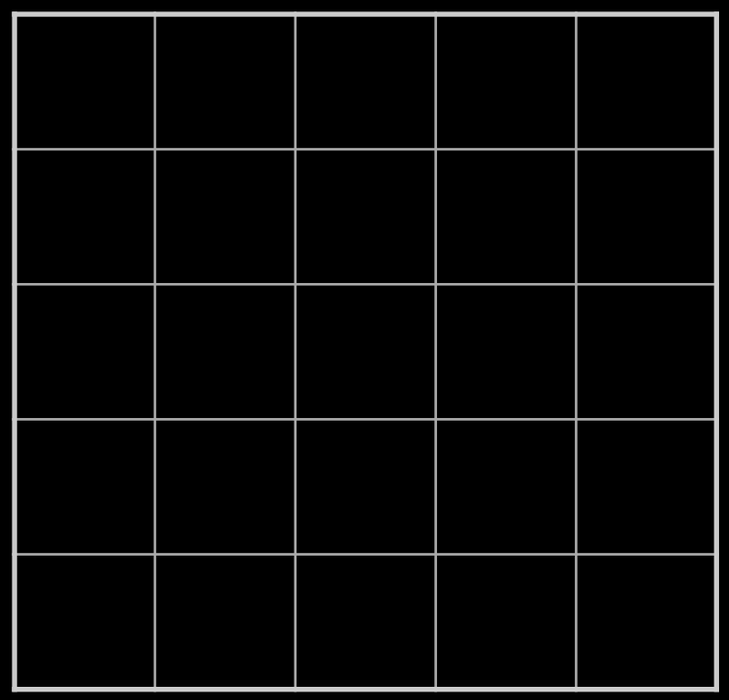
Previously: matching patches

1. **Naive search** $O\left(mwM\right)$
   for every pixel p
   - define a small neighbourhood $\mathcal{N}$
   - compare patch v_p to others in $\mathcal{N}$
2. **Nearest neighbour** $O\left(kwM\right)$
   - compute PCA of every patch
   - compare low-dimensional patches

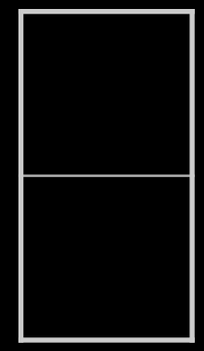


Key ideas

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



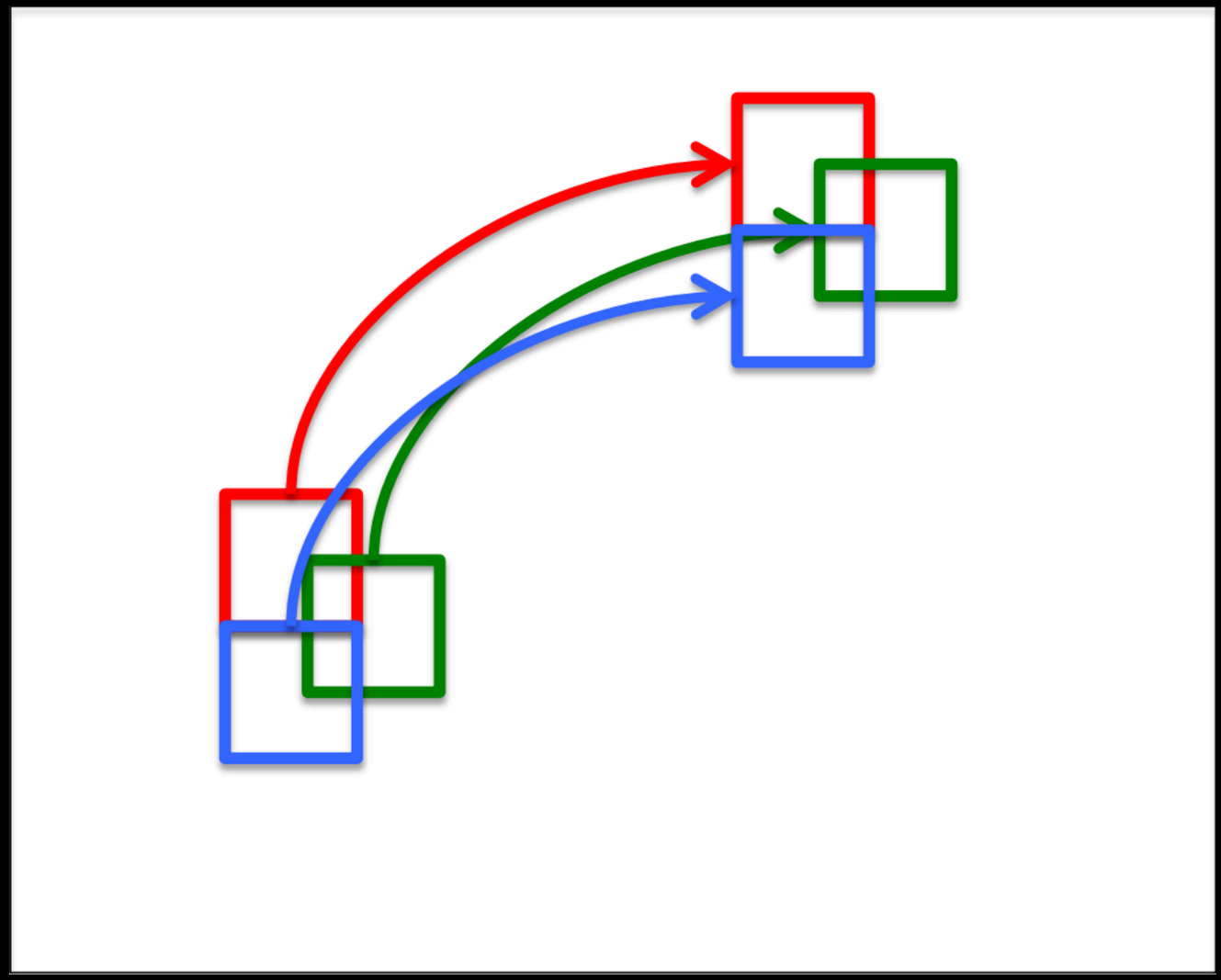
Patches of size 25



Patch offsets of size 2

Key ideas

Coherent matches with neighbours



Key ideas

M — total number of pixels

Probability of correct random guess: $\frac{1}{M}$

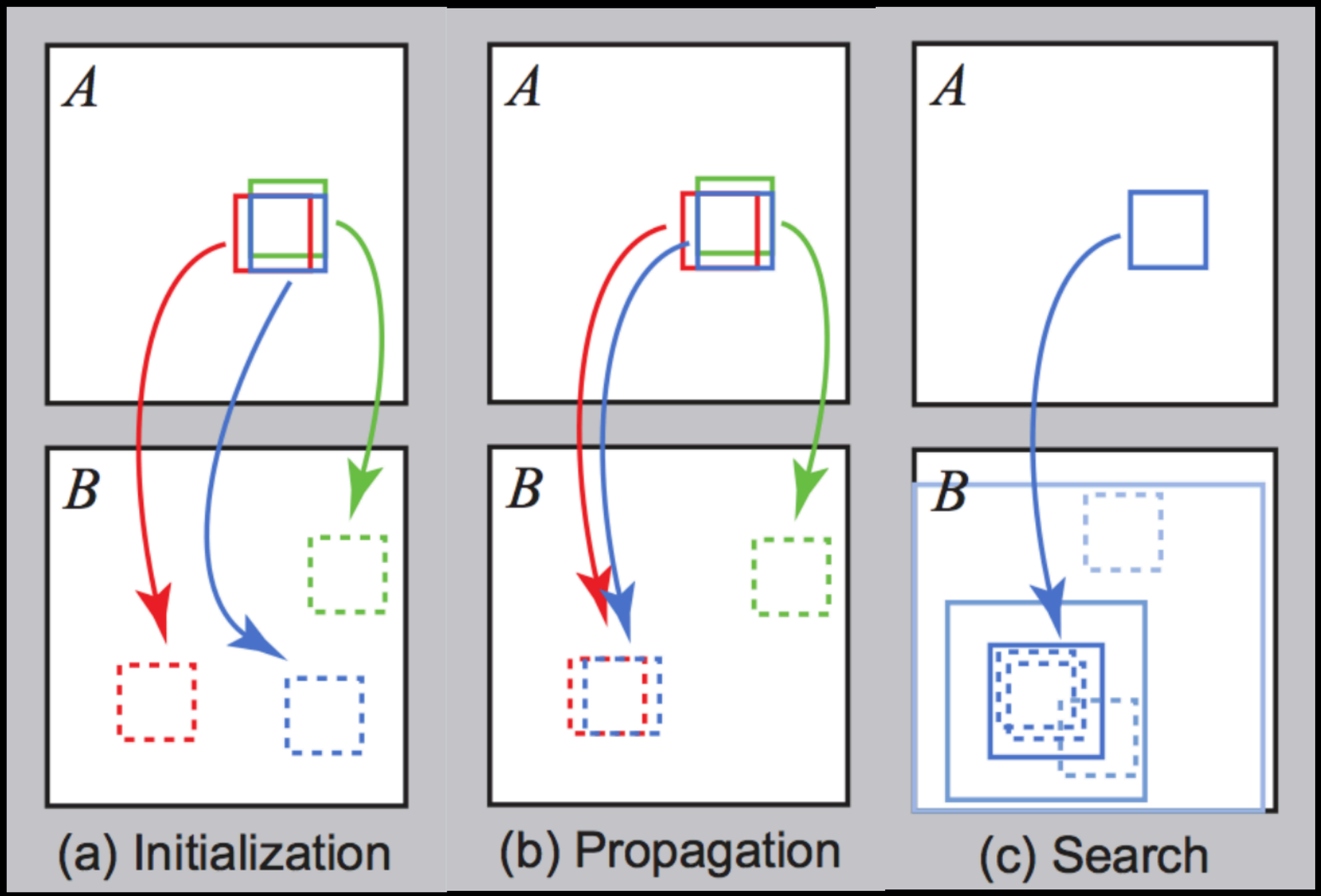
Probability of incorrect random guess: $1 - \frac{1}{M}$

Probability of all pixels with incorrect random guess: $\left(1 - \frac{1}{M}\right)^M \approx 0.37$

=> Probability of at least one pixel with a correct guess: $1 - \left(1 - \frac{1}{M}\right)^M$

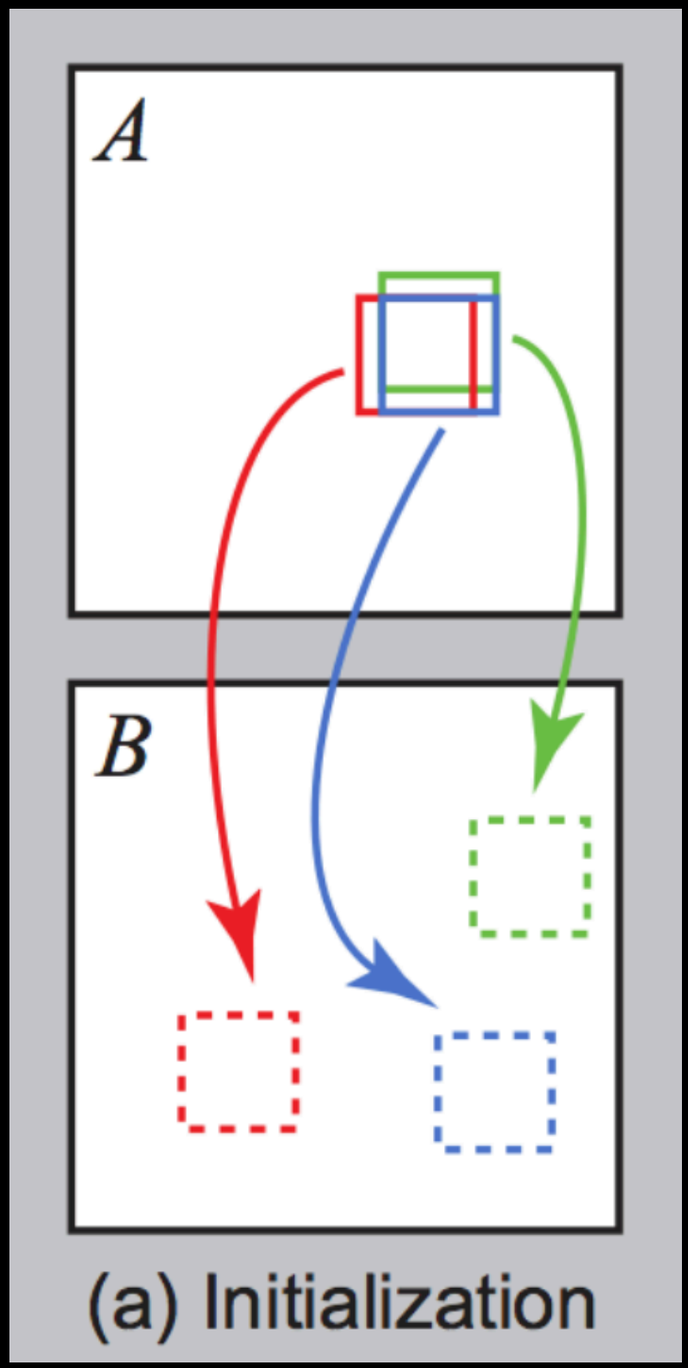
=> Probability of at least one pixel with a good enough guess: $1 - \left(1 - \frac{C}{M}\right)^M$

Algorithm: 3 steps



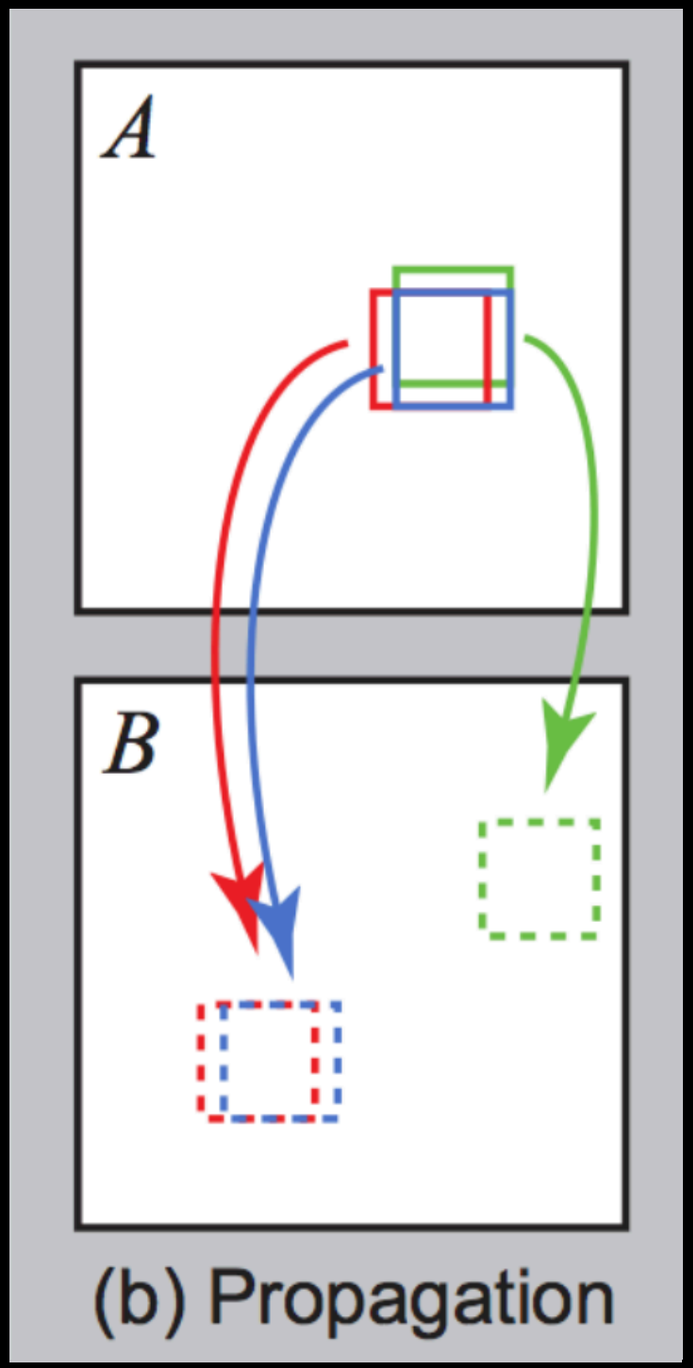
Step 1: Initialisation

Each pixel is given a random offset as initialisation.



Step 2: Propagation

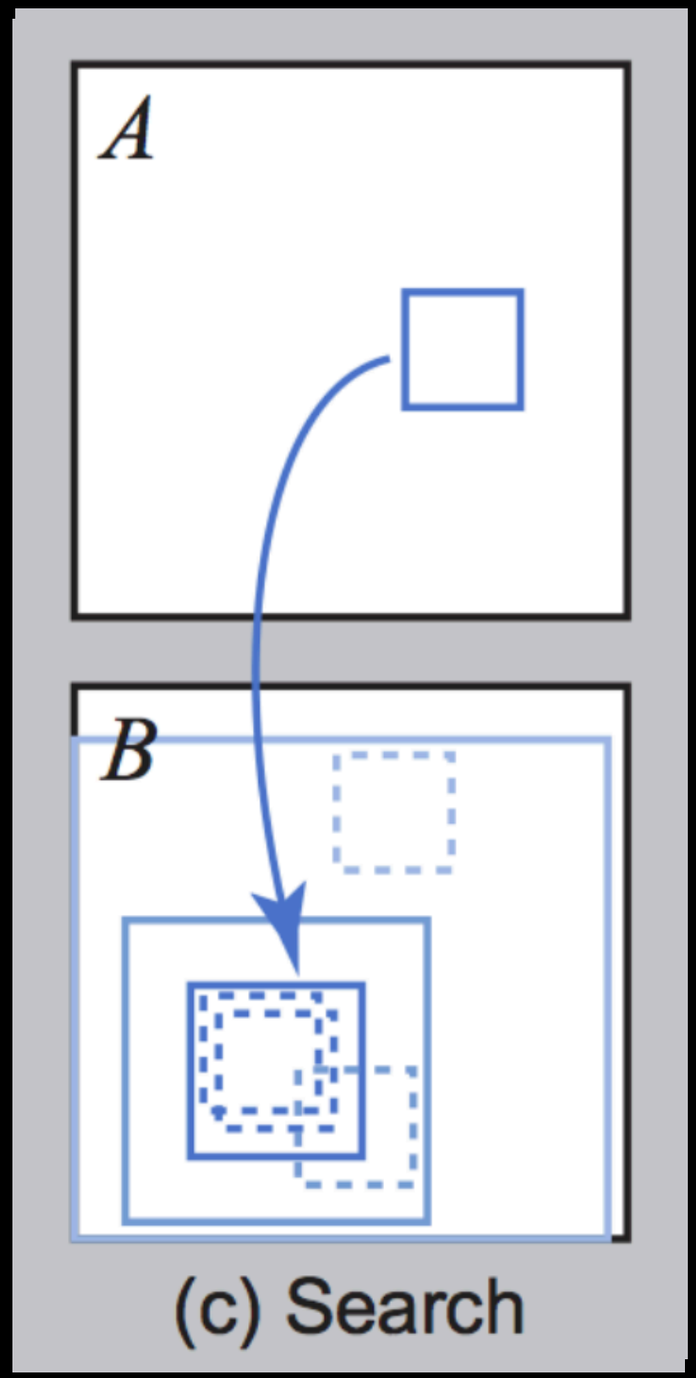
Each pixel checks if the offset of one of the neighbouring pixels give a better matching offset. If so, adopt the offset.



Step 3: Random search

Each pixel randomly chooses a candidate offset within a concentric radius around the current offset. If this offset is better, adopt it.

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



Algorithm

1. Initialise pixels with random offsets.

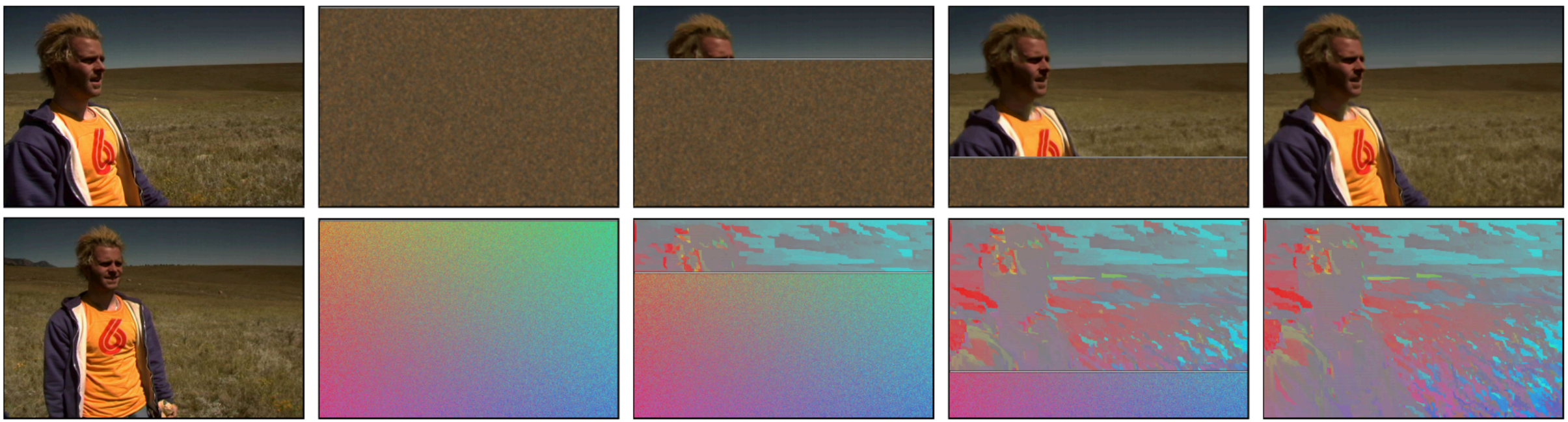
2. For each pixel from top-left to bottom-right:
   1. Adopt a better offset from top and left neighbours.
   2. Randomly search for a better offset.

$$O\left(mM \log M\right)$$

3. Repeat step 2, but go from bottom-right to top-left and check bottom and right neighbours.

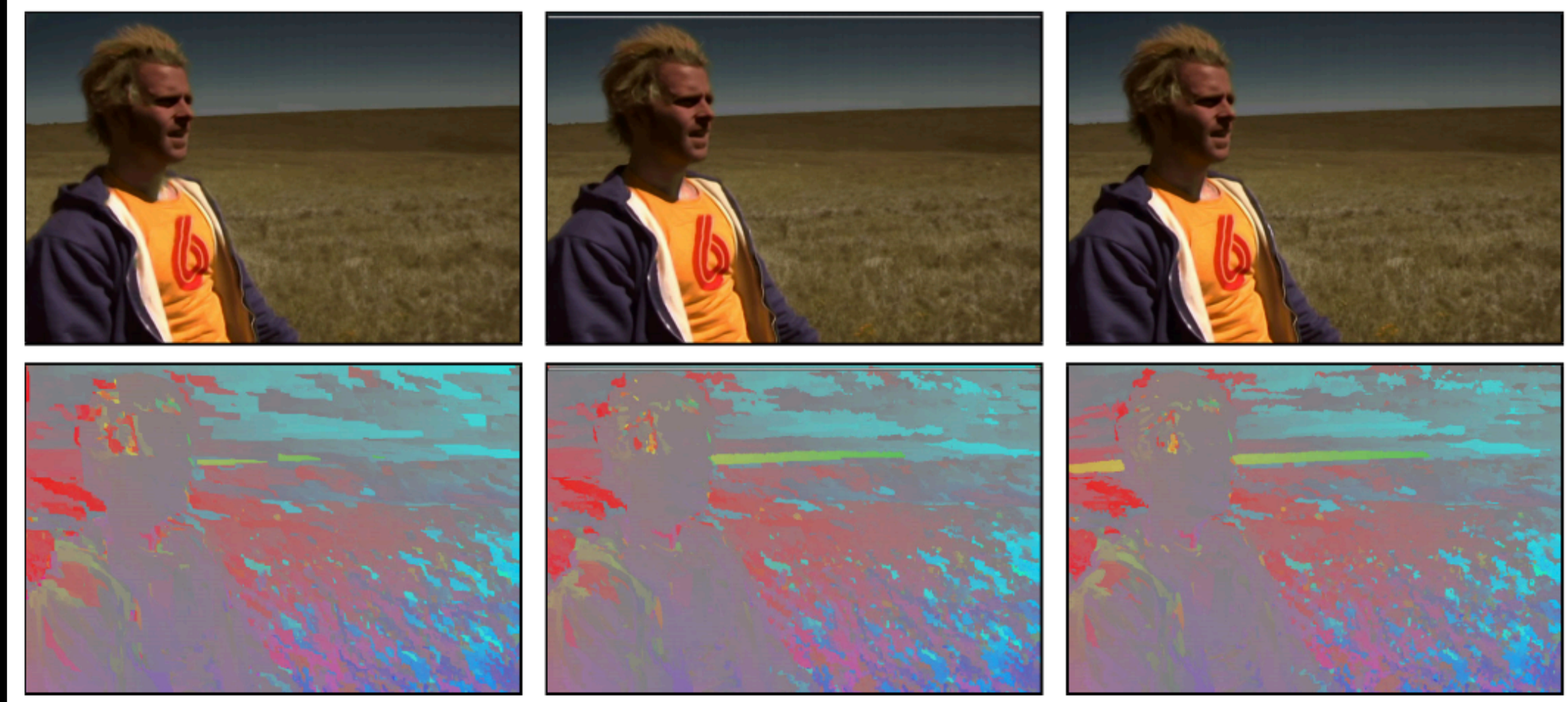
4. Repeat steps 2 and 3.

Example



(a) originals (b) random (c) $\frac{1}{4}$ iteration (d) $\frac{3}{4}$ iteration (e) 1 iteration

Example



(e) 1 iteration (f) 2 iterations (g) 5 iterations

Image retargeting

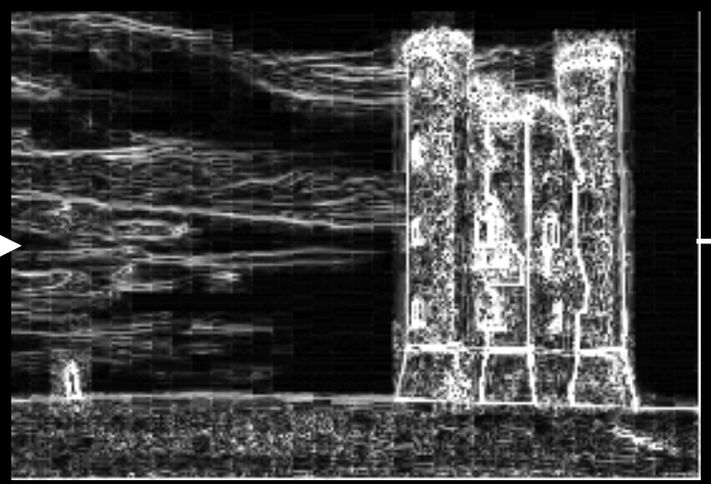
(c) Seam Carving, SIGGRAPH 2007

Problem invented by Setlur et al., 2005

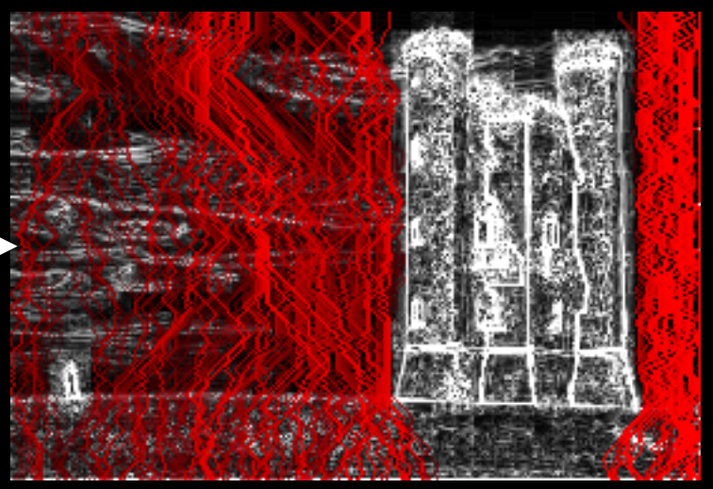
https://github.com/vivianhylee/seam-carving

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.

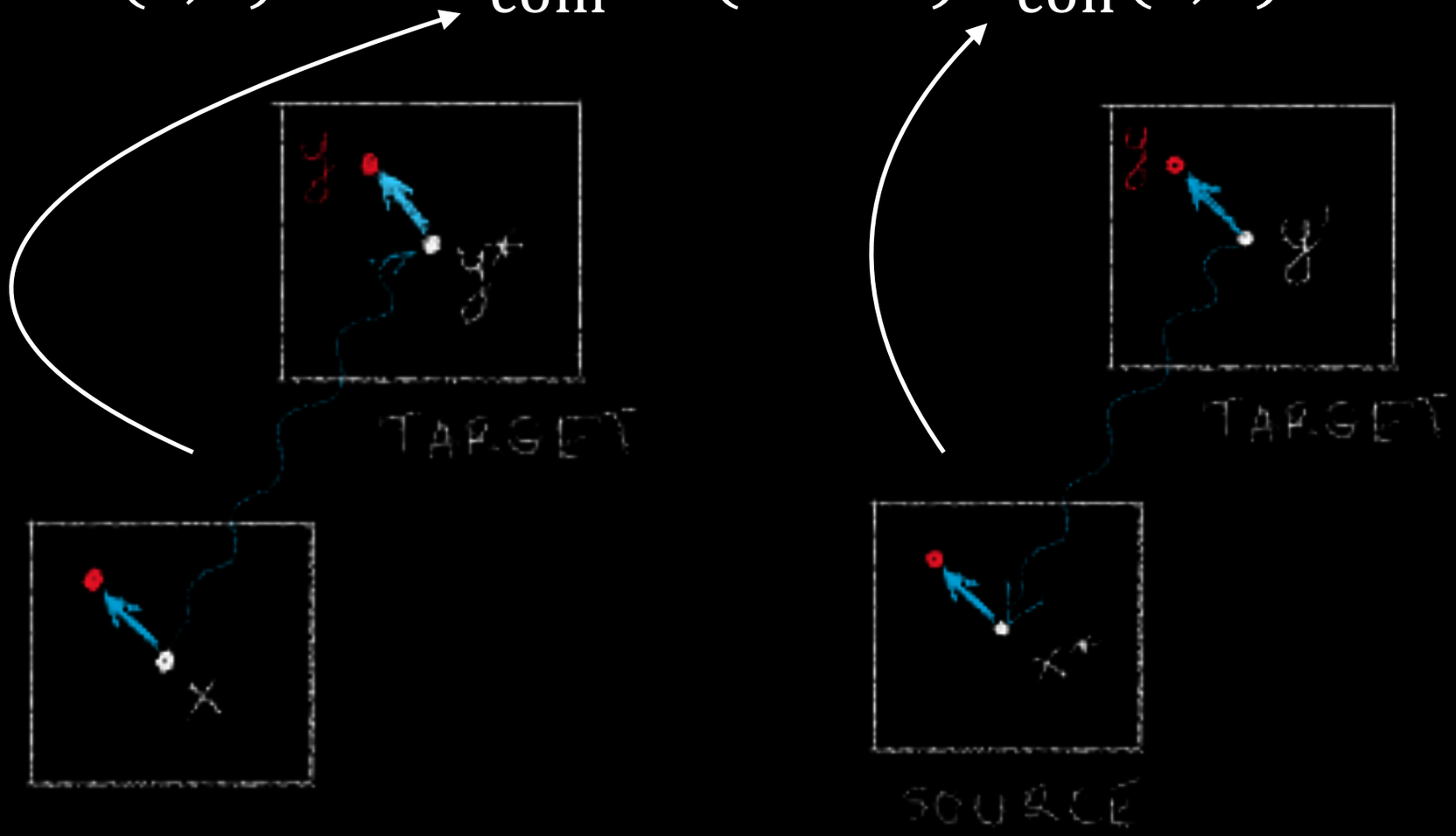
Rank the seams according to energy and remove them in an ascending order of energy

(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

PatchMatch

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

- Initialize $t^{k+1} \equiv 0 \quad c \equiv 0$
- For each $\mathbf{y} \in \Omega'$
  - Find $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} \left\| \pi_{\mathbf{x}} s - \pi_{\mathbf{y}} t \right\|_2^2$
  - $t^{k+1}|_{\mathbf{y}+\blacksquare} \leftarrow t^{k+1}|_{\mathbf{y}+\blacksquare} + \frac{1-\alpha}{|\Omega'|} \pi_{\mathbf{x}^*} s$
  - $c|_{\mathbf{y}+\blacksquare} \leftarrow c|_{\mathbf{y}+\blacksquare} + \frac{1-\alpha}{|\Omega'|}$
- 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.t coherence and completeness
- Average them appropriately
- Decrease the dimensions of t by a bit and iterate

Line constraints

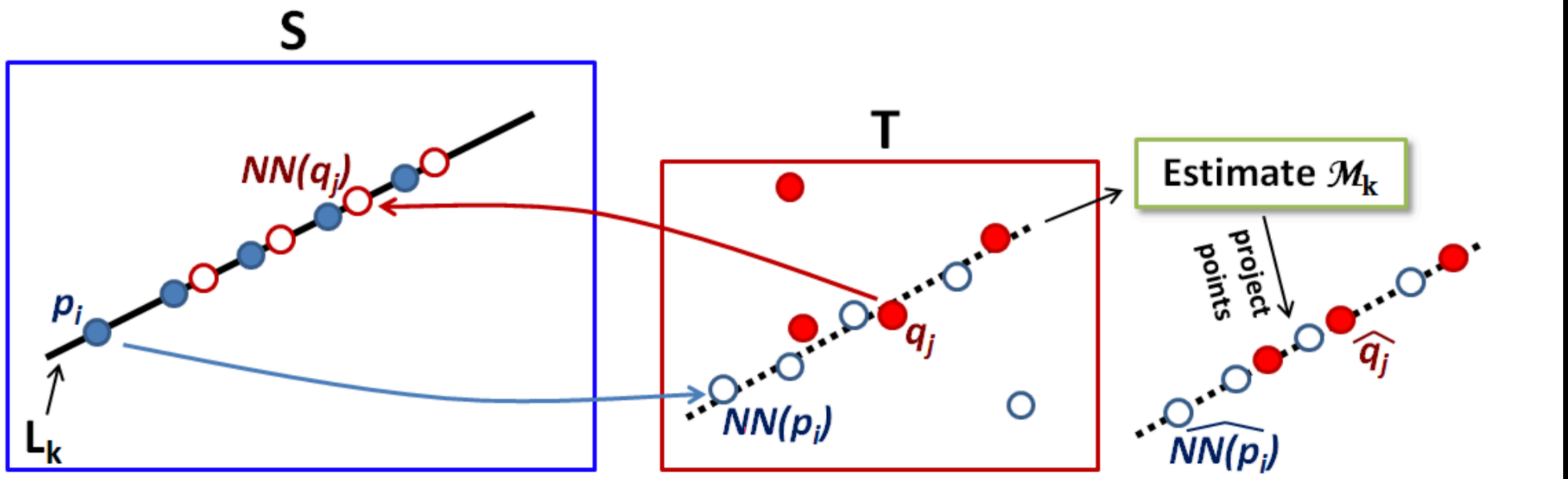
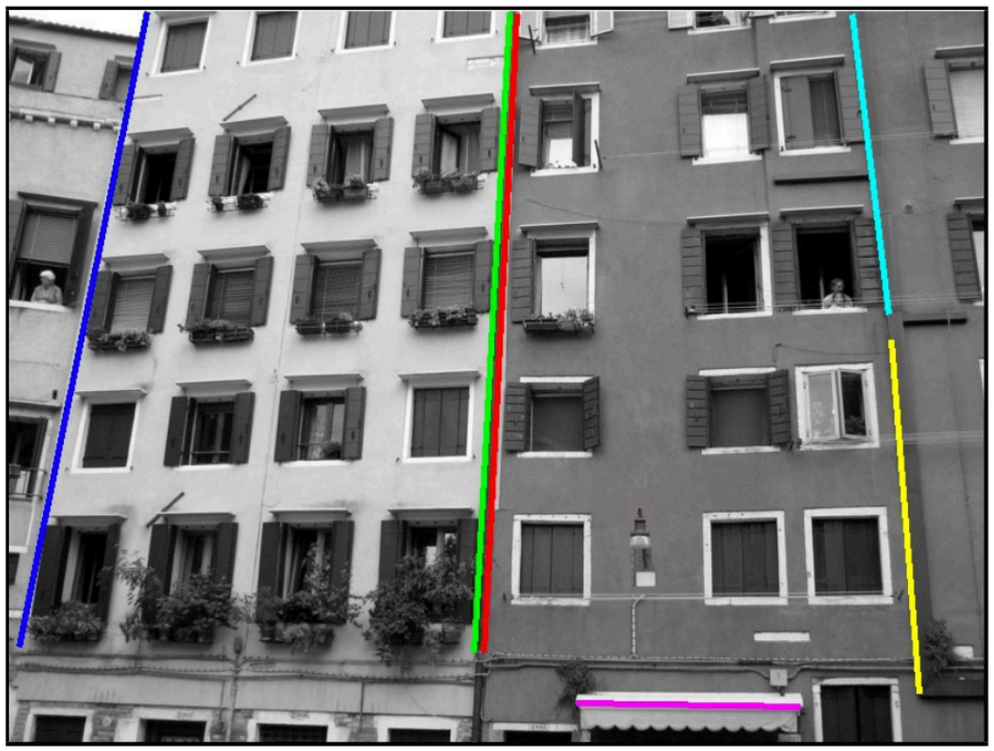


Figure 7: *Model constraints. L_k is a straight line in the source image S. For point p_i on L_k, the nearest neighbor in T is $NN(p_i)$. A point q_i in T has nearest neighbor $NN(q_i)$ that lies on L_k. We collect all such points $NN(p_i)$ and q_i and robustly compute the best line M_k in T, then project the points to the estimated line.*

Retargeting

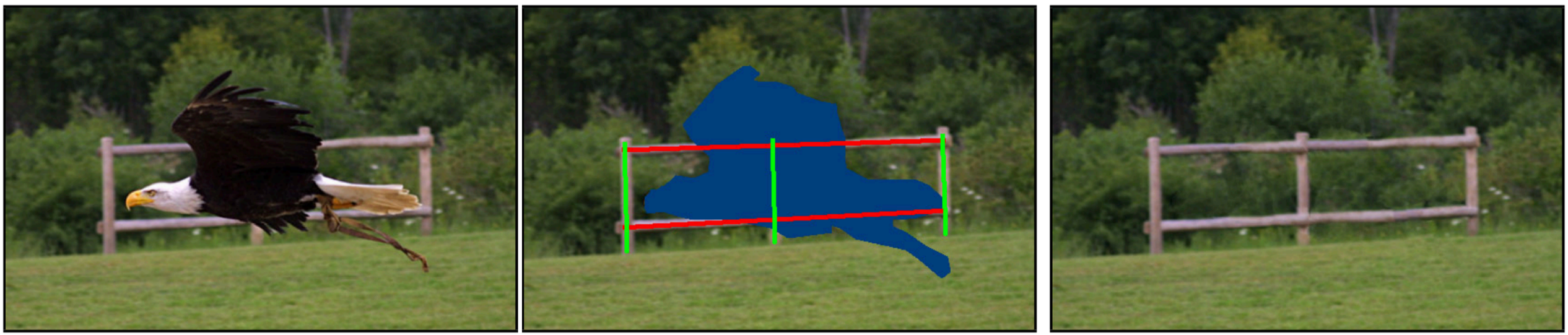


(a) original

(b) retargeted

(c) with constraints

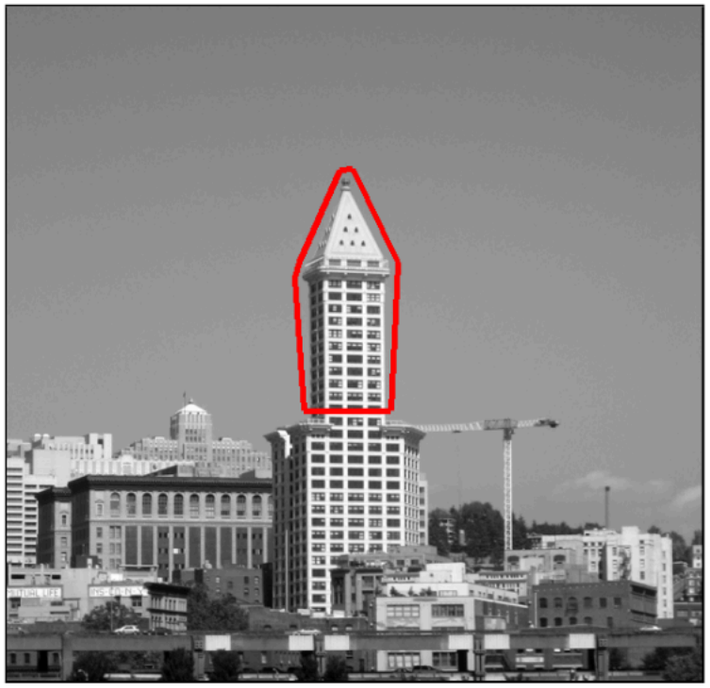
Constrained inpainting



Reshuffling

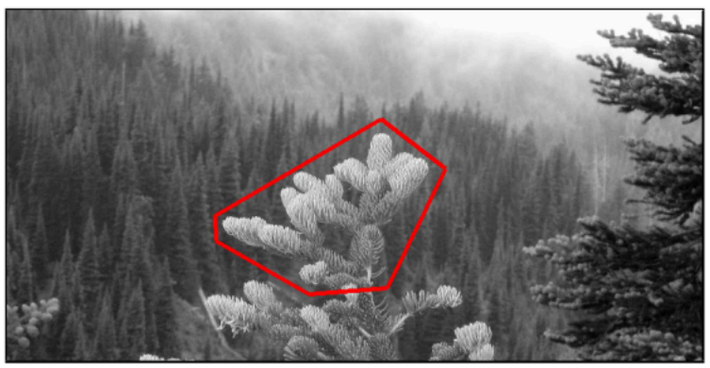
(a) input (b) our reshuffling

Local structural scaling



(a) building marked by user

(b) scaled up, preserving texture



(c) bush marked by user

(d) scaled up, preserving texture.

Summary

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

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