A3: PatchMatch

Slides by Sara Tang, adapted from Yawen Ma

Link to paper: https://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/patchmatch.pdf

Learning Goals

- Introduction to PatchMatch; what it's solving
  - Nearest-Neighbour Field
- Understand the PatchMatch algorithm
  - Propagation
  - Random search
- Applications of PatchMatch

Remember this?

```
Initialize C(p) for all pixels.
While the image is not filled:
    Calculate the priority for each pixel/patch
    that lies on the boundary of the mask.
    Get the patch with the highest priority.
    Find its closest match in the image.
    Fill the (unfilled part of the) patch with its
    match.
    Update the boundary of the mask.
```

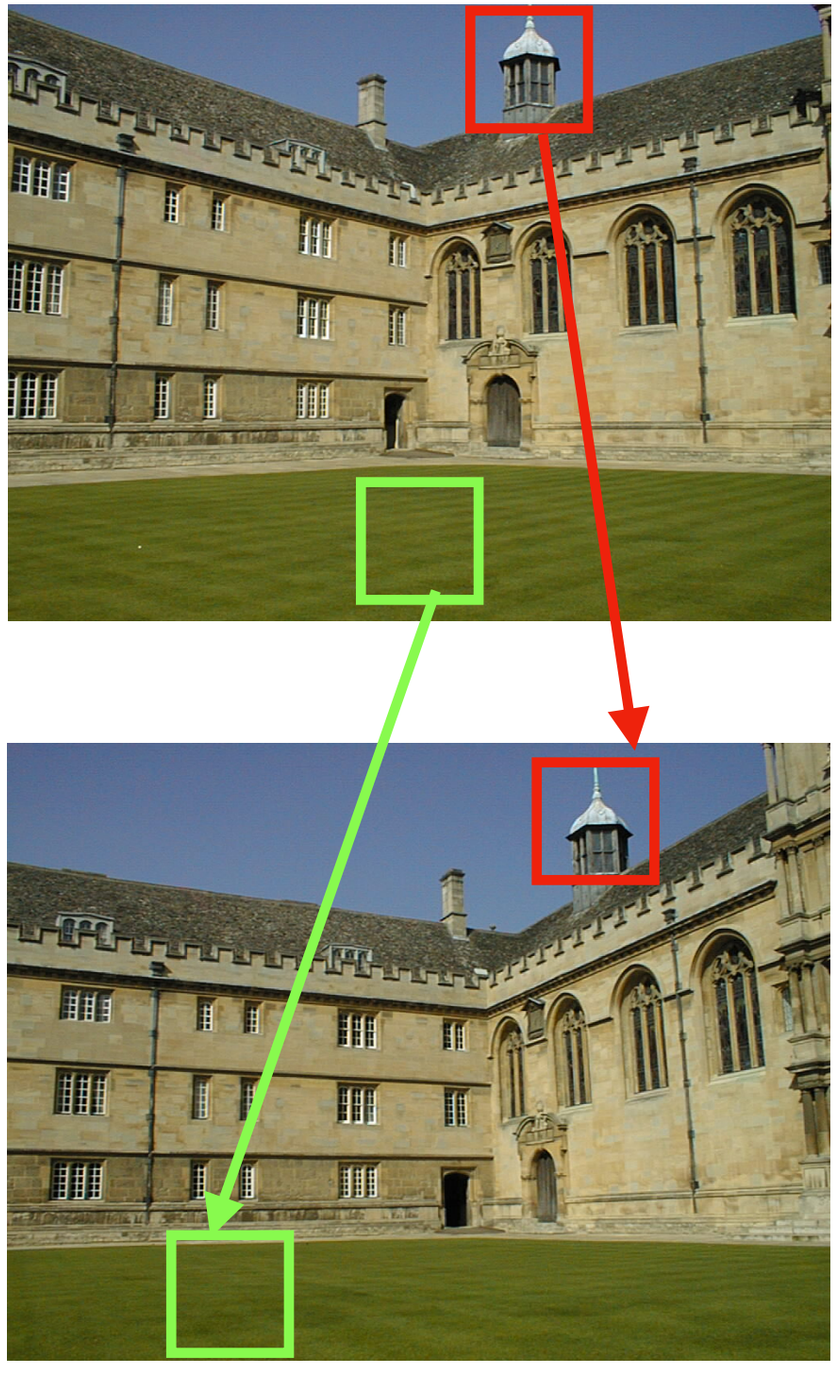
Remember this?

```
Initialize C(p) for all pixels.
While the image is not filled:
  Calculate the priority for each pixel/patch
  that lies on the boundary of the mask.
  Get the patch with the highest priority.
  Find its closest match in the image.
  Fill the (unfilled part of the) patch with its
  match.
  Update the boundary of the mask.
```

PatchMatch

- Inputs: image A, image B, patch size

- Output: correspondence that maps each patch in image A to its best match in image B

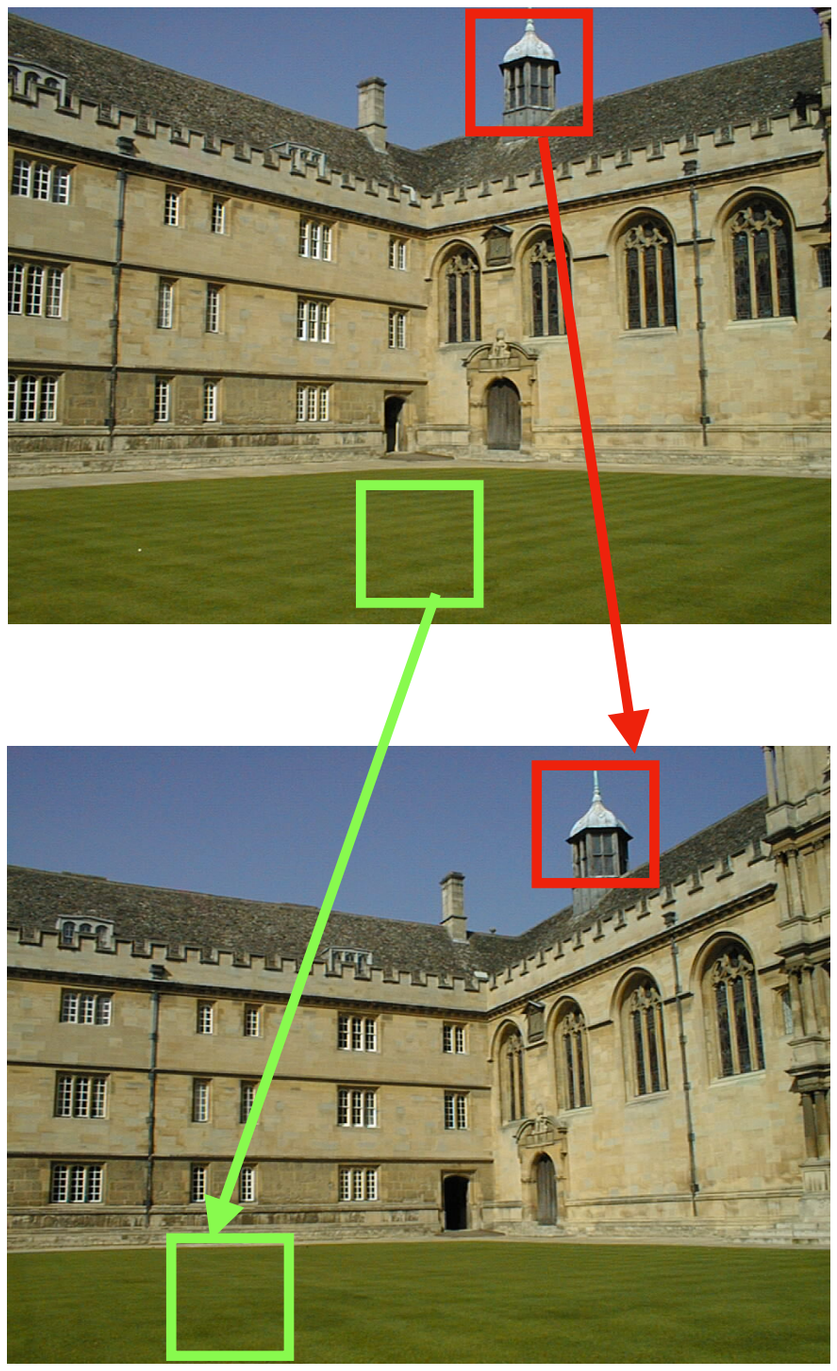


Nearest-Neighbour Field (NNF)

$$f : A \to \mathbb{R}^2,$$

where A is a pixel in image A representing the **centre of a patch**, and $\mathbb{R}^2$ is the **offset** to get from the centre of a patch in A to the centre of its match in B.

Ie. Given a patch in A centred at (x, y), its closest match in B is the patch centred at$(x, y) + f(x, y)$.



PatchMatch Intuition

- Patches in an image repeat often enough that we can **find a "good" match randomly**. (**random search**)

- Neighbouring patches are similar. Therefore...

- ...if we find a "good" match (p_A, p_B), we can probably find similarly good matches for the neighbours of p_A by **searching neighbouring patches** of p_B. (**propagation**)

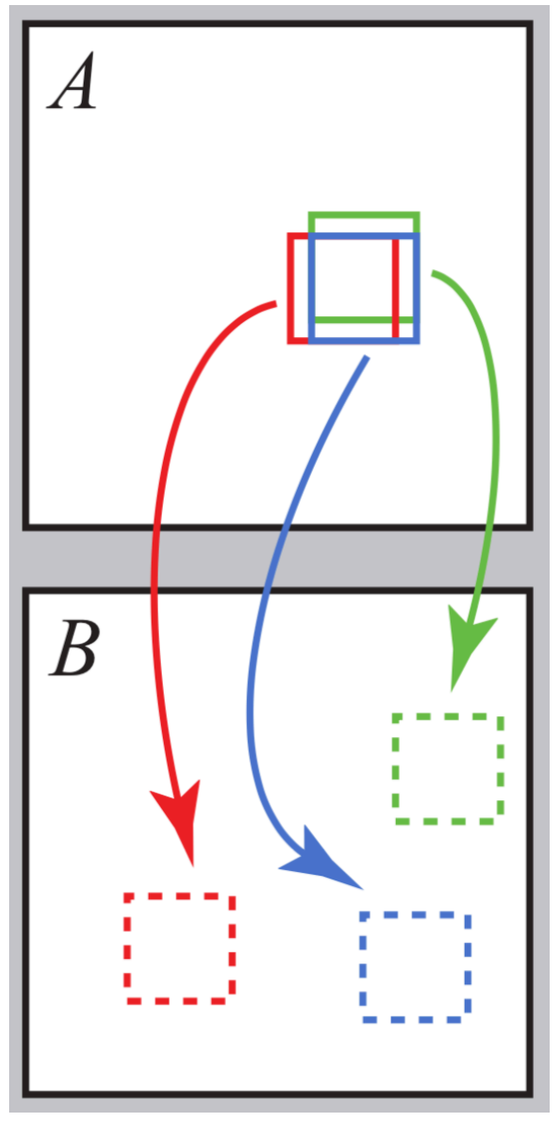
Algorithm Outline

```
Initialize a NNF for patches in image A.
Iterate max_iters times:
  For each pixel:
    Update NNF via propagation.
    Update NNF via random search.
```

Initializing the NNF

- You can either initialize the NNF randomly...

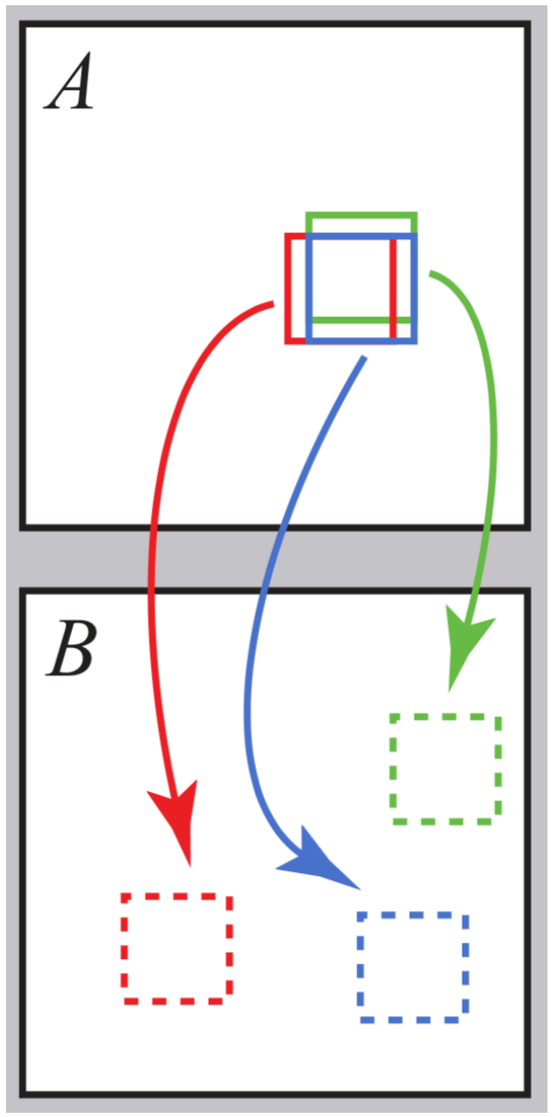
- ... or you can import an NNF from a previous run / another algorithm.



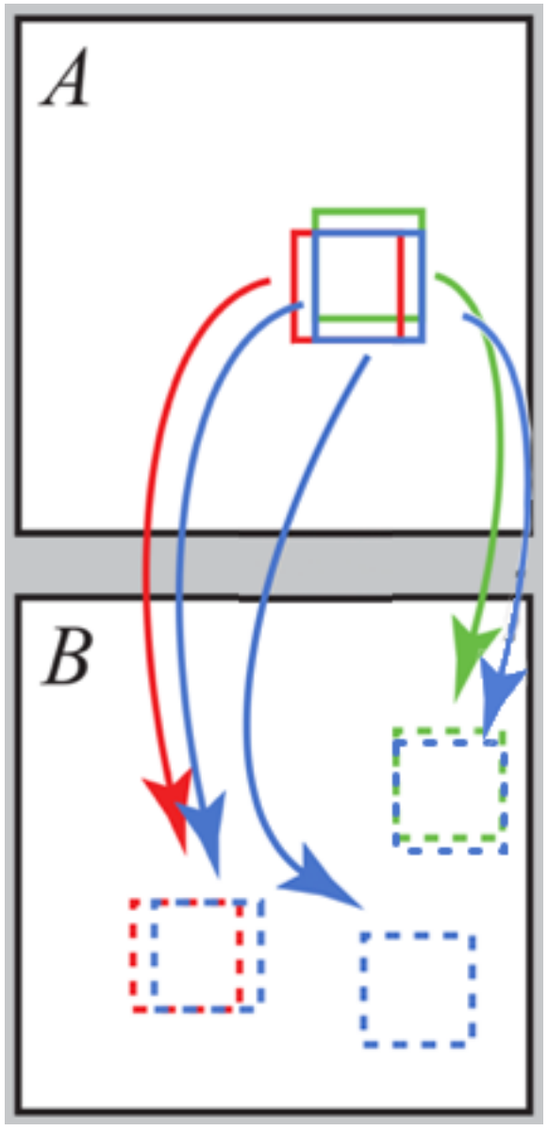
Algorithm Outline

```
Initialize a NNF for patches in image A.
Iterate max_iters times:
  For each pixel:
    Update NNF via propagation.
    Update NNF via random search.
```

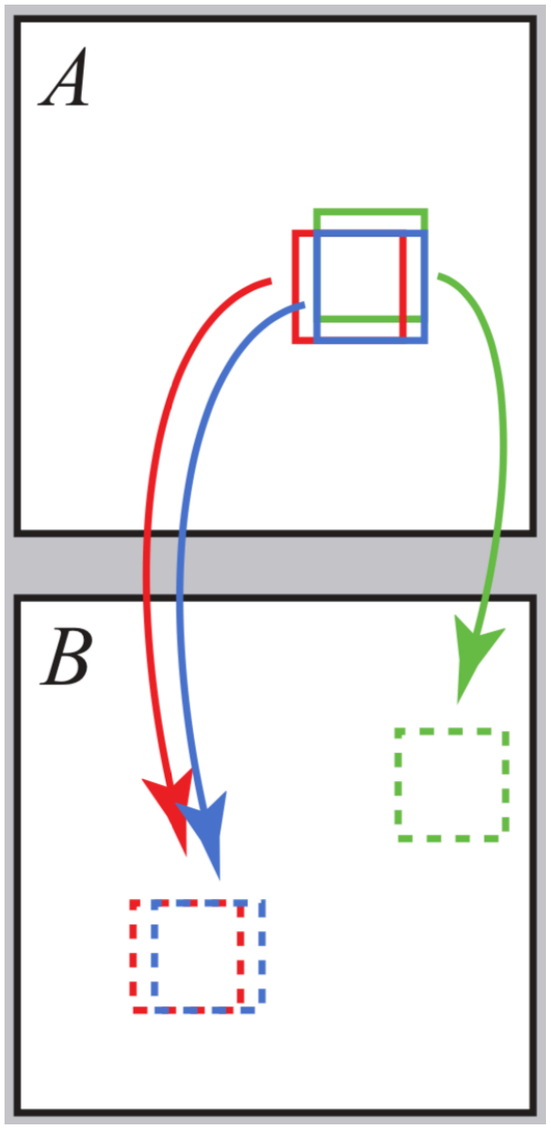
Propagation



Before.



Apply offsets.



Propagate!

Propagation

Let f be the current n.f.f. as defined previously.
Let $p_I(x,\ y)$ be denote the patch centred at $(x,\ y)$ in image I.
Let $D(p_1,\ p_2)$ be a patch-difference function.
We want to update the offset for $p_A(x_0,\ y_0)$ via propagation.

For odd iterations,

$$\begin{aligned} f(x_0, y_0) = \arg\min\{ & D(p_A(x_0, y_0),\ p_B((x_0, y_0) + f(x_0, y_0))), \\ & D(p_A(x_0, y_0),\ p_B((x_0, y_0) + f(x_0 - 1, y_0))), \\ & D(p_A(x_0, y_0),\ p_B((x_0, y_0) + f(x_0, y_0 - 1))\} \end{aligned}$$

For even iterations,

$$\begin{aligned} f(x_0, y_0) = \arg\min\{ & D(p_A(x_0, y_0),\ p_B((x_0, y_0) + f(x_0, y_0))), \\ & D(p_A(x_0, y_0),\ p_B((x_0, y_0) + f(x_0 + 1, y_0))), \\ & D(p_A(x_0, y_0),\ p_B((x_0, y_0) + f(x_0, y_0 + 1))\} \end{aligned}$$

Iterating through the patches

In odd iterations, update the pixels from top to bottom, left to right.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

In even iterations, update the pixels from bottom to top, right to left.

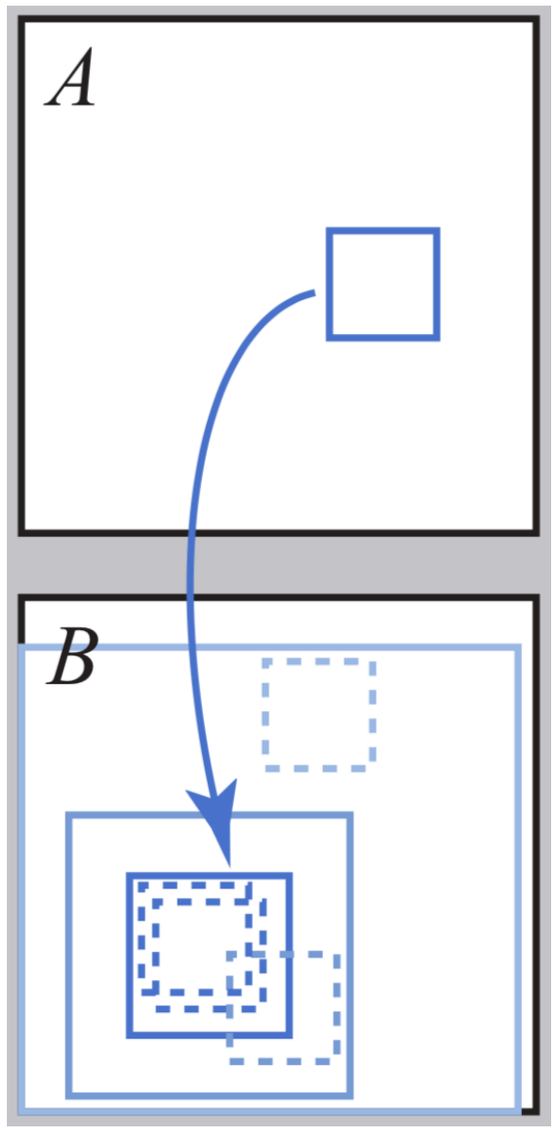
15	14	13	12	11
10	9	8	7	6
5	4	3	2	1

Why??

Algorithm Outline

```
Initialize a NNF for patches in image A.
Iterate max_iters times:
  For each pixel:
    Update NNF via propagation.
    Update NNF via random search.
```

Random Search



Choose random offsets within smaller and smaller windows.
Pick the best one!

Random Search

We want to update the offset for $p_A(x_0,\ y_0)$ via random search. Define the random search candidates in B to be centred at:

$$u_i = v_0 + wa^i R_i$$

where

$$\begin{aligned} v_0 &= f(x_0, y_0) \\ w &= \text{maximum image dimension} \\ a &= \text{ratio between search window sizes} \\ R_i &= \text{uniform random in } [-1,1] \times [-1,1] \end{aligned}$$

Then,

$$\begin{aligned} f(x_0, y_0) = \arg\min\{ & D(p_A(x_0, y_0), p_B((x_0, y_0) + v_0)), \\ & D(p_A(x_0, y_0), p_B((x_0, y_0) + u_1)), \\ & \vdots \\ & D(p_A(x_0, y_0), p_B((x_0, y_0) + u_n))\} \end{aligned}$$

Questions?

```
Initialize a NNF for patches in image A.
Iterate max_iters times:
  For each pixel:
    Update NNF via propagation.
    Update NNF via random search.
```

Applications of PatchMatch