PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

Ajay Shrihari - 20171097

Chaitanya Kharyal - 20171208

Rahul Sajnani - 20171056

Anoushka Vyas - 20171057

Problem Statement

- The main objective of this project is to present interactive image editing tools using a new randomized algorithm for quickly finding approximate nearest neighbor matches between image patches. This algorithm forms the basis for a variety of tools – that can be used together in the context of a high-level image editing application.
- One more feature is additional intuitive constraints on the synthesis process that offer the user a level of control unavailable in any other methods. Previous methods in this field are generally very slow and could not be ported into applications where user input was allowed. This method aims to allow for the use of tools in a real time scenario.

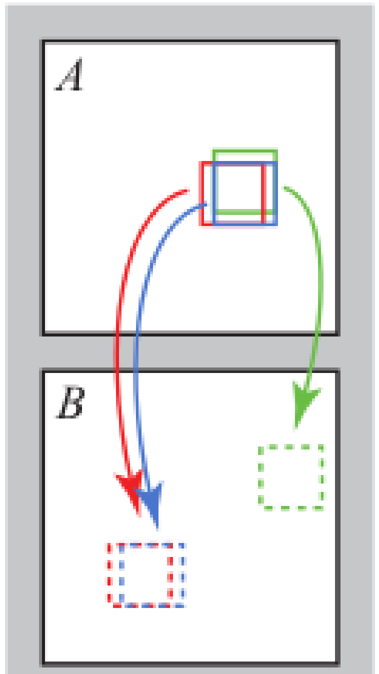
Method

Notations

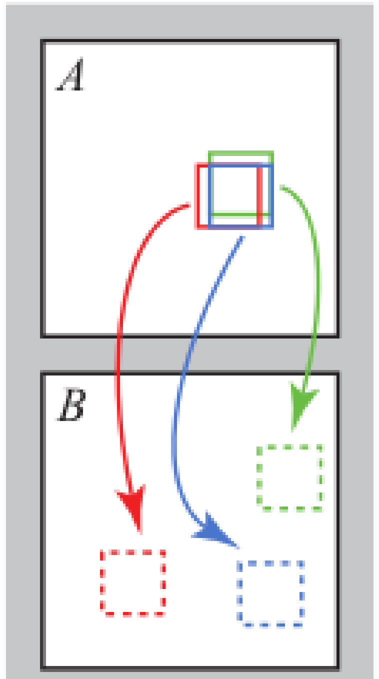
f(x, y) determines the location of the patch in image **B** that is closest to the patch at the location (x, y) in image **A**. D(A[x, y]. B[f(x,y)]) determines the distance (can be mean L1 cost) between the patches of image **A** and image **B**. Here, A[x,y] and B[x,y] determine the patch with center at (x,y) in image A and image B.

Algorithm -1

- **Initialization**: The algorithm initializes the patch locations by sampling from a uniform random distribution of locations and assigns the initial patch matching error. It then iteratively finds the closest matching patch in a 2-step process.
- **Propagation**: In this step, we look at the locations of (f(x-1, y), f(x, y - 1)) (in even itr) and (f(x + 1, y), f(x, y + 1)) (in odd itr) for closest patch to (x,y). The idea here is that the patch for the neighbors of (x, y) should be close to the patch at (x,y).
- The final distance D = min(D(A[x, y]. B[f(x,y)]), D(A[x, y]. B[f(x - 1,y)]), D(A[x, y]. B[f(x,y - 1)])).



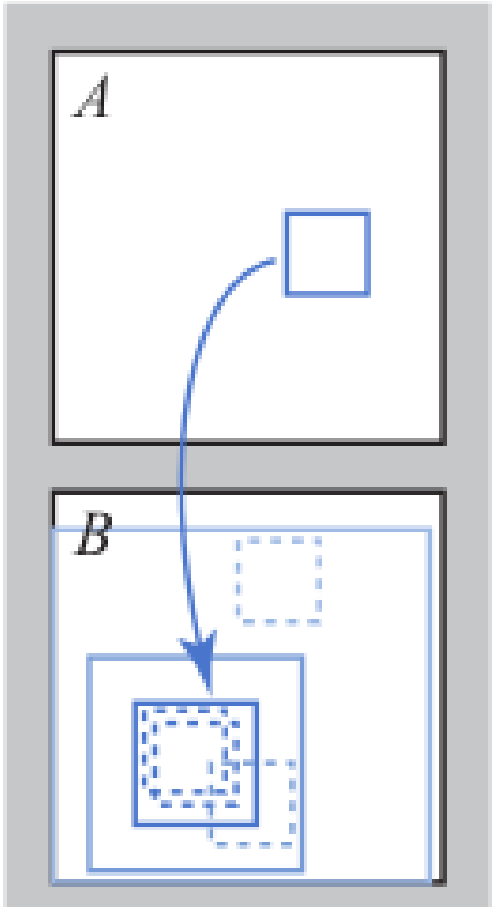
(b) Propagation



(a) Initialization

Algorithm-2

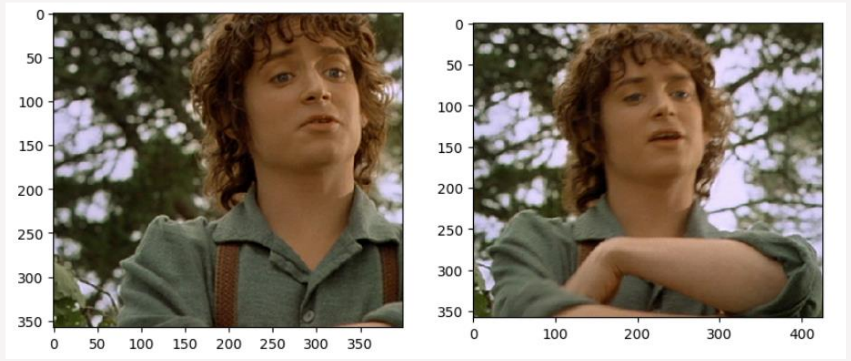
- **Random search**: Propagation step might converge to a non-optimal minima in finding the nearest patch to resolve this the paper uses random search. We search in a larger patch in image B at certain offsets.
- If pos = f(x, y) (remember this provide position of the closest patch in image B to the (x, y) of image A).
- pos_new = pos + w*a^{i}*R
- pos_new gives us the new location to search over. R is uniform random in [-1, 1] x [-1, 1]. We search till w*a^{i} is smaller than 1 pixel. Here a=1/2.



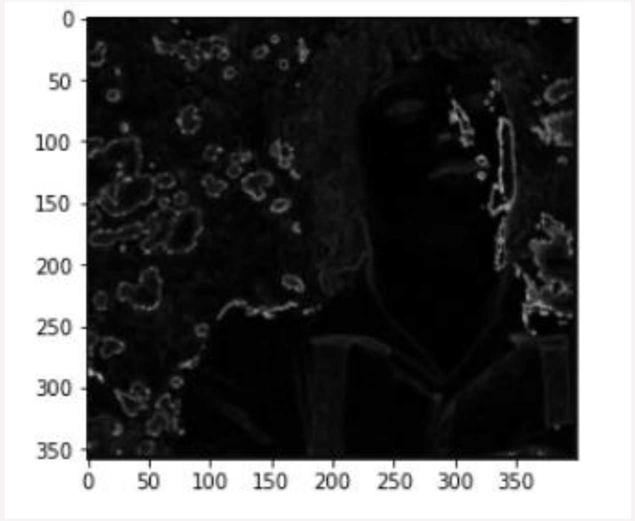
(c) Search

Current progress

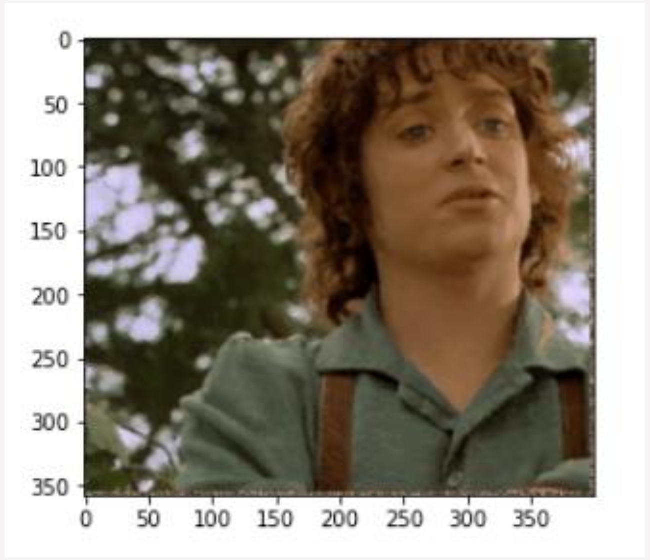
- Basic patch match implementation, with **initialization**, **propagation**, and **random search** described above is complete.
- Using patch match, reconstruction of one image given another image is complete, as shown in the paper.
- Code is modular, and distinct functions have been written for each step within the same class.
- Helper functions have been written for plotting and reading images from directories.
- The mid evaluation milestones have been achieved, and we are on track with the timeline given in the project proposal.



Input images



Output image from patch match: nearest patch distances

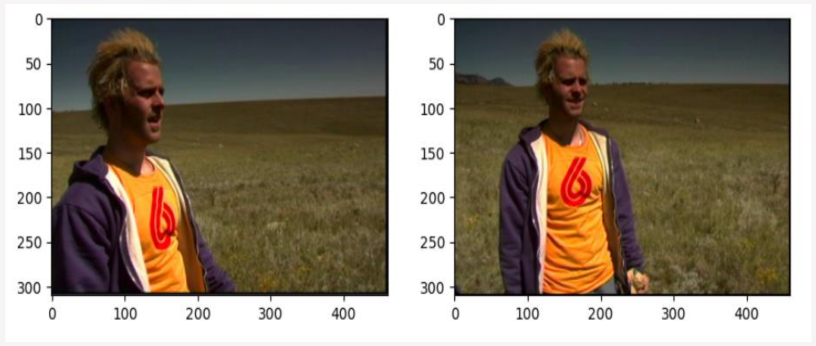


Reconstructed image

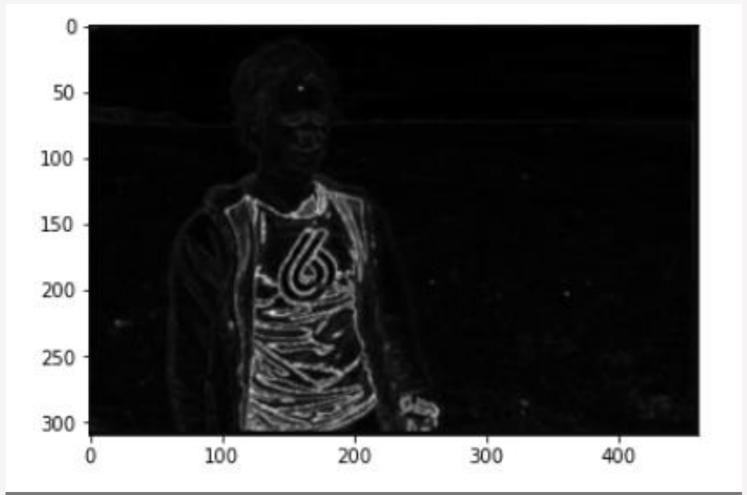
Results –1

Using the patch match algorithm, image reconstruction of the input image on the left, using patches from the input image on the right.

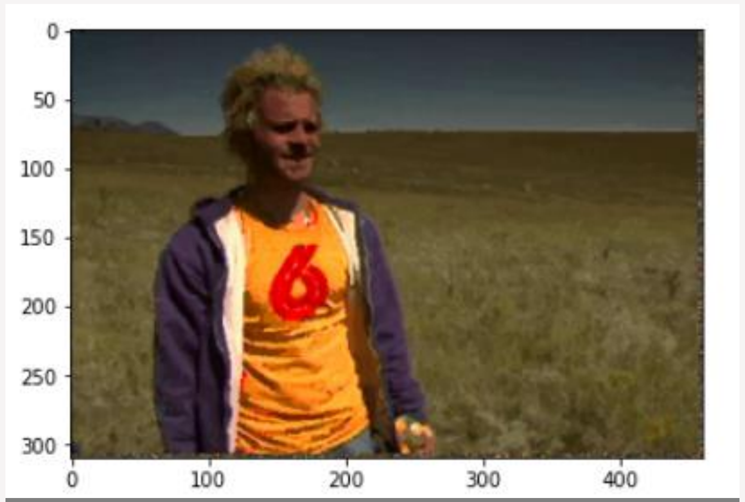
Results - 2



Input images

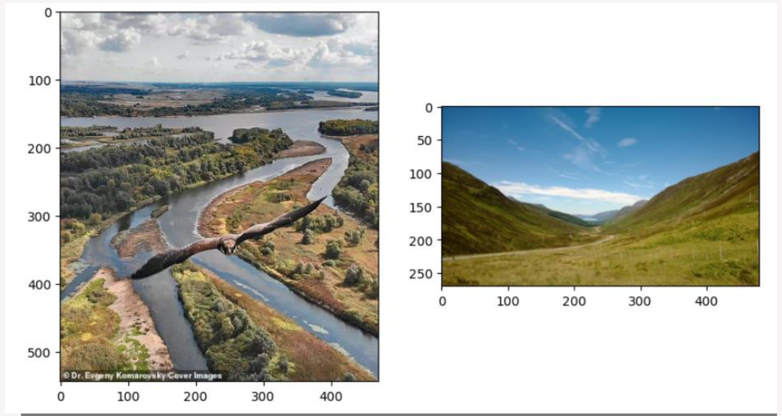


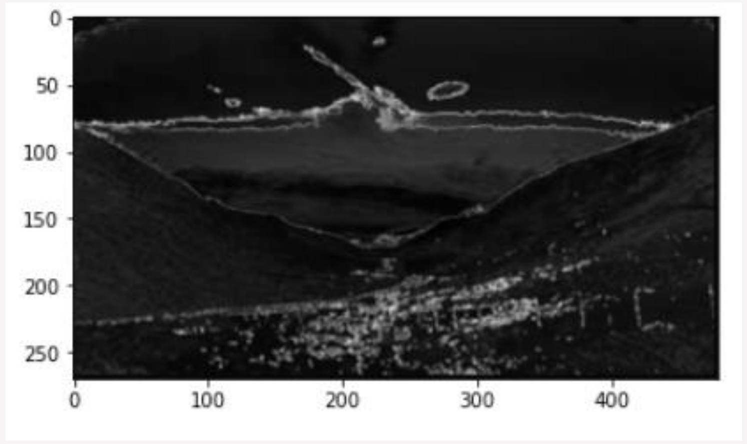
Output image from patch match: nearest patch distances

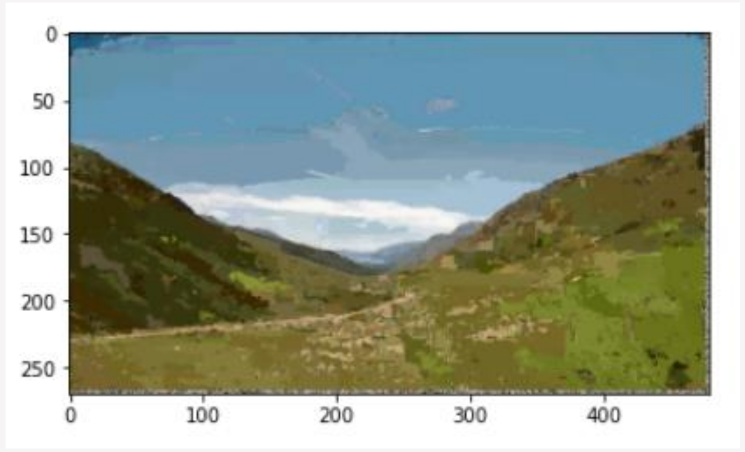


Reconstructed image

Results - 3


Input images


Output image from patch match: nearest patch distances


Reconstructed image

What remains?

- Further applying the basic patch match algorithm to the datasets given in the proposal – depending on the degree of visual similarity.
- A GUI needs to be built for the interactive tool, in order to show a visualization and efficacy of the speed enhancements patch match provides.
- The GUI will have user input capabilities.
- Applying the patch match algorithm to certain use-cases.