DIP Project Proposal - PatchMatch

Team cv2

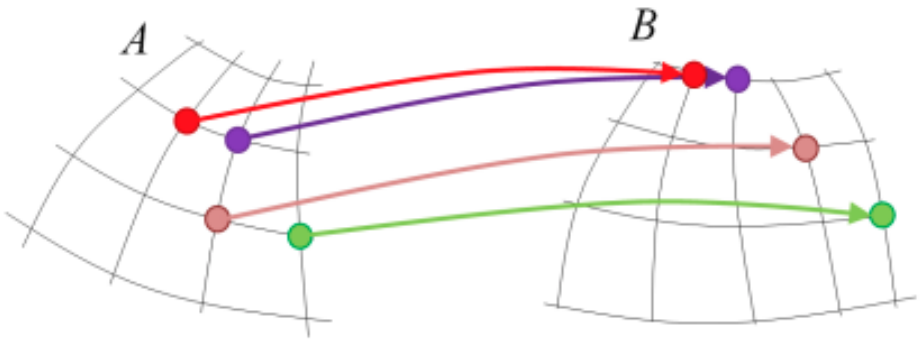
Introduction

Existing algorithms for image completion allow users to simply erase an unwanted portion of an image, and the computer automatically synthesises a fill region that plausibly matches the remainder of the image. However, searching by patches carried considerable overhead in time and memory, as finding similar patches was challenging. PatchMatch greatly accelerates the search for image patches to such an extent that these tasks can now be done interactively.

Methodology Used

Image Details

Natural images exhibit coherence, where adjacent patches tend to have similar nearest neighbours, and a peaked distribution, where better matches are often spatially proximate. These properties enable efficient search algorithms by propagating good matches and focusing on nearby areas for improved accuracy in finding nearest neighbour patches between images.



High level motivation behind the algorithm. We match descriptors computed on two manifolds, visualised as colour circles. Each descriptor independently finds its most similar match in the other manifold. Coherence here could be indicated by the red descriptor being similar to the purple descriptor, and that their nearest neighbours are spatially close to each other.

PatchMatch Algorithm

1. **Random Initialisation:** Start by initialising the nearest-neighbour field (NNF) with random values. Each patch in image A is assigned a random patch in image B as its nearest neighbor, using independent uniform samples. Alternatively, we can initialise the NNF with some fixed values that are not to be modified.
2. **Propagation:** For each patch in image A, we attempt to improve its current nearest neighbour using the nearest neighbours of adjacent patches (e.g., left or above). This propagates good matches across coherent regions of the image.
3. **Random Search:** For each patch, we generate a sequence of candidate nearest neighbours by adding a random offset to the current nearest neighbour.
4. **Evaluation Criteria:** We evaluate these candidates and update the nearest neighbour if any of them provide a smaller patch distance defined by the l2-norm of the difference of RGB values.

5. **Halting Criteria:** We iterate the above process a fixed number of times (typically 4-5 iterations) until the NNF converges, meaning that further iterations no longer significantly improve the nearest neighbour assignments.

Implementation Strategy

We plan to implement 3 tasks:

1. Image in-painting

   This task involves completing an image after masking out unwanted objects from the scene.

2. Deformation with constraints

   This task focuses on warping or reshaping parts of an image while maintaining specific constraints or anchor points.

3. Object Relocation

   This task involves moving objects within an image to new positions while maintaining visual consistency.

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

Structural image editing. Left to right: (a) the original image; (b) a hole is marked (magenta) and we use line constraints (red/green/blue) to improve the continuity of the roofline; (c) the hole is filled in; (d) user-supplied line constraints for retargeting; (e) retargeting using constraints eliminates two columns automatically; and (f) user translates the roof upward using reshuffling.

We also plan on making an interactive UI similar to the youtube video reference given below.

References

dl.acm.org
https://dl.acm.org/doi/pdf/10.1145/1531326.1531330

gfx.cs.princeton.edu
https://gfx.cs.princeton.edu/pubs/_2011_PAF/connelly_barnes_phd_thesis.pdf

PatchMatch - A Randomized Correspondence Algorithm for Structural Image Editing
Advanced Technology Labs - Graphics Imaging

https://www.youtube.com/watch?v=fMe19oTz6vk