VFXfinal

tags: VFX

b05902062 王子銘 b05902066 蔡翔陞

Coordinates for Instant Image Cloning

- This is a fast seamless image/video cloning method which could run in realtime.

Introduction

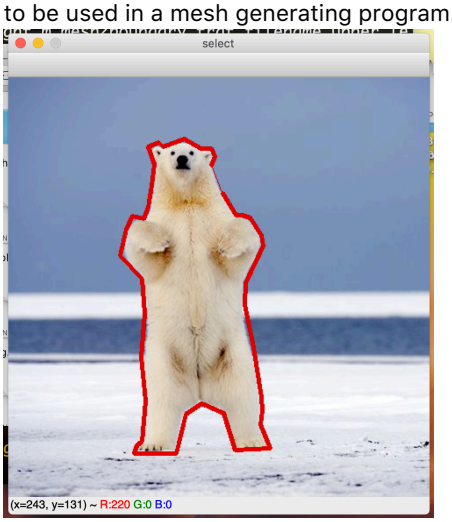
- In this project, we implement a fast image cloneing system accroading to http://www.cs.huji.ac.il/~danix/mvclone/ (http://www.cs.huji.ac.il/~danix/mvclone/).
- The paper persents a new image cloning method based on Mean-Value Coordinates with some speed up algorithm to achieve seamless realtime cloing.
- We would briefly explain how it works, what we implemebt and some result.
- Our program and a description on its usage is available on Github https://github.com/b05902062/image_cloning_tool (https://github.com/b05902062/image_cloning_tool).

How it works

- The basic idea is that human visual system is only sensitive to the difference of lightness and the slow luminance change would be ignored.
- This algorithm uses Mean-Value Coordinates(MVC) to calculate how each pixel in a cloned region should be adjusted to match a target image. MVC makes the color differences at the edge diffuse to the inner region smoothly by giving each pixel a weight.
- This method avoid the use of a huge linear system and it is also memory-efficient and all the value could be solve independently, which means it can run in parallel to further speed up its processing time.

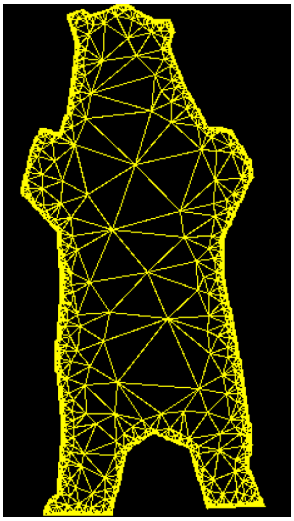
What we did

1. Crop the target region
    - we use opencv with python to create a GUI allowing a user to select the region to be cloned. In creat_poly.py, we record some mouse event and create the boundary to be used in a mesh generating program.



2. Create several submesh

- Based on the paper, we could generate some adaptive mesh to speed up the MVC calculating process by only counting the color differences at the triangle mesh's vertexs.
- We use the open-sourced library Triangle (https://www.cs.cmu.edu/~quake/triangle.html (https://www.cs.cmu.edu/~quake/triangle.html)) to help us create those meshes.



3. Hierarchical boundary sampling

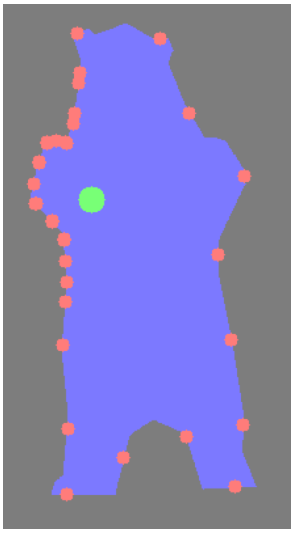
   - Because a pixel would have less influence on pixels that are farther away from it. When calculating how much should a pixel be adjusted, we could sample the vertexs on the boundary which would affect more.
   - The sampling percedure uses a coarse-to-fine method.
     - Firstly, for every vertex in the mesh, which is described above, we select 16 vertexs on the boundary that is uniformly distributed.
     - Step two, we check every boundary vertex on the following conditions:

$$\begin{aligned} \|\mathbf{x}-\mathbf{p}_i^k\| &> \varepsilon_{dist} \\ \sphericalangle \mathbf{p}_{i-s}^k, \mathbf{x}, \mathbf{p}_i^k &< \varepsilon_{ang} \\ \sphericalangle \mathbf{p}_i^k, \mathbf{x}, \mathbf{p}_{i+s}^k &< \varepsilon_{ang} \end{aligned}$$

       x is the vertex in the triangle mesh, p are the boundary vertices and k is the current depth in the hierarchy.
     - E_dist and E_angle is set to the same value as mentioned in the paper

$$\varepsilon_{dist} = \frac{\#\text{ boundary pixels}}{16 \cdot 2.5^k} \quad \text{and} \quad \varepsilon_{ang} = 0.75 \cdot 0.8^k,$$

     - For every boundary vertex which doesn't pass the condition test, we add two neighbor points of it together with itself into the next finer level (k+=1) and go back to step two.
     - After serval rounds, when all selected boundary vertices pass our test, they becomes the boundary vertices used to calculate MVC for the mesh vertex x.



4. Calculate the MVC weight:
    - We calculate MVC weights for each mesh vertex according to the paper.

$$\lambda_i(\mathbf{x}) = \frac{w_i}{\sum_{j=0}^{m-1} w_j}, \quad i = 0, \ldots, m-1, \tag{1}$$

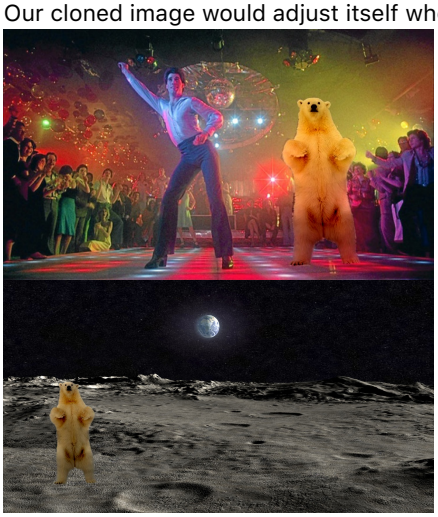
where

$$w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{\|\mathbf{p}_i - \mathbf{x}\|}, \tag{2}$$

5. A GUI to paste our cropped image:
    - We first calculate the difference on the edge of a cloned region between our source image and a target image.
    - Then we calculate the diffenrence at each vertex in the mesh using MVC weight and subsampling boundary vertices.
    - Thirdly, we calculate the difference of each pixel in a triagle by interpolating the three vertices of the triangle mesh with a weight that is proportional to the inverse of their distances.
    - Lastly, we add up our cropped image and the differences calculated above and then paste it to our target image.

result

- Our cloned image would adjust itself when put into background with different lighting:



- In the background without huge contrast, the result could be seamless and look natural.
  - The upper-left image is our source and the upper-right image is our target. The image at the bottom is our result



- This is a comparison between MVC method (bottom-left) and directly copying (bottom-right) a cropped region into a source image. The source image are cropped in the exact same way for the two output images. As can be seen,the MVC method substantially improves the smoothness in the output image.