Comp Photography (Sp 2015)
HW 3

Jonathan Hudgins

Image 1

Image 2

Image 3

Image 4

Image 5

Final

Snow falling in backyard. Composited to combine all snow keeping the background.

You can see snow falling in front of the trees in images 1 - 5 that all are present in the final image.

Details of the Pictures you Took

- What are these images of?
  - I took 6 pictures of snow falling in my backyard with my Samsung S5 camera phone using automatic camera settings. I held the camera steady and only a few seconds between each picture.
  - In order to improve processing time each picture was cropped to the bottom right corner (500x400 pixels).

Image 1

I used this image as the base image to match features and transform all other images to this space.

Notice the snow feature in the bottom left corner. This is a good reference point for seeing where the pictures will match up.

Image 2

In order to move this image into Image 1 space it will be shifted down.

Notice the three lines of snow on the tree. This feature is clearly visible in the final image.

Image 3

The foreground snow on the bottom shows that this image needs a significant shift.

Image 4

The camera was clearly moving slightly down (the foreground snow fills more space on each subsequent image).

Image 5

It looks like there is a large piece of snow (closer to the camera) in the upper left region. It blends with the entire background because it is out of focus.

Final Image

The algorithm details are below, but notice a couple of artifacts:

- The black region
- The various snow falling in front of the tree trunk
- The border to the right of the tree trunk and around the snow in the bottom left corner

What was the motivation/goal?

- Often pictures of snow falling are disappointing. The snow is so small that it is difficult to see. Usually the motion of snow makes it stand out more. In order to make a picture show more snow, I composited the pictures keeping using an “outlier” filter.

How did you generate the final?

- I used OpenCV and wrote python code according to the following pseudo-code:
  - Load and crop images (for processing speed)
  - Warp images 2 through 5 to the space of image 1 (using http://docs.opencv.org/doc/tutorials/features2d/feature_flann_matcher/feature_flann_matcher.html)
  - Blend images (choose outliers)

Warp Details

- Create features for all images (sift.detectAndCompute)
- Match features of images 2-5 to base image 1 (FlannBasedMatcher.knnMatch: k=2)
- Sort features by ratio of FlannBasedMatcher
- Choose 3 non-coincident features to use for creating transform between images 2-5 to base image 1 (getAffineTransform).
- Transform images (warpAffine)

Blend Details (Outlier)

For each pixel choose the color value from all images that is farthest away from the average. For the parts of the scene that are the same, these pixels should be identical anyway (showing the background). Whenever falling snow shows up in the foreground, this outlier will be used.

Was it successful?

Yes!

What would you do differently?

- Crop the final image
- Make camera more stationary (pressing against window)
- I had to remove a picture that was warpping poorly (skewed), probably because of selecting features that created narrow triangles. Better feature selection should fix this (more equalateral triangles).
- Create C++ code for blending (for improved performance)
- Downsample images for feature matching (improved performance)

Any other details.

- This is a great technique for enhancing features that humans normally see because of our bias towards motion.