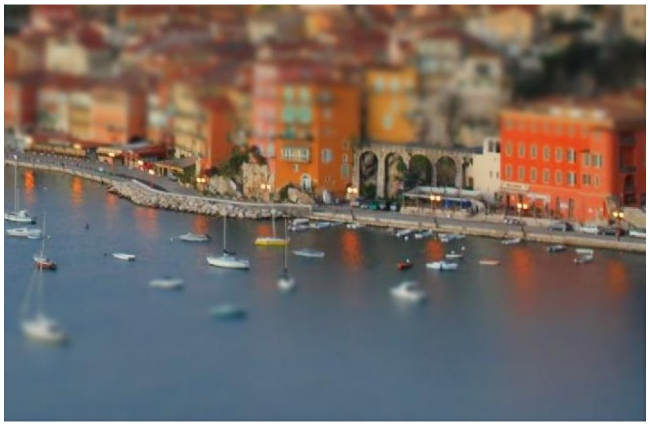
Programming Project #1 (proj2)
15-463: Computational Photography

Focus on Miniatures



Due Date: 11:59pm on Mon, February 8, 2010

Overview

In this assignment you will create fake miniatures (a prolific flickr topic) by simulating the effect of selective focus cameras, also known as Tilt Shift. Most methods rely on photo editors like GIMP and Photoshop to achieve the effect. A user selects a focus plane by masking a region of interests and applying a blurring filter to the rest of the image. Effectively this narrows the perceived depth of field in the scene and creates the illusion that the lens was really close to the subject.

Details

In your prorgam you should be able to do the following:

1. Set a **focus line** by drawing lines or selecting points in the image.
2. Define the size of the fake **depth of field (DOF)** with respect to the focus plane as several pixels below and above.
3. **Increasingly blurr** the image around the focus plane to simulate depth of field effect.
4.

Focus line: From a 2D image it is technically impossible to recover depth information. Rather than accurately defining at what distance objects are, we can use a good heuristic that objects on the same straigh line are at the same depth. This heuristic holds for many landscape photographs, especially when you have a nice sloppy panoramic view (hint: travel to France). In matlab you might find *ginput* useful.

Depth of field: The depth of field is the part of the image in focus. Notice that in an actual camera focusing on a plane means that all objects in that plane are in focus. Since we are faking the process and approximating the focus plane by sa simple line choosing the right size for the depth of field is very important. In the simple situations the DOF is just a rectangle around the focus line. For extra credit you can better define the depth of field region to account for irregular objects and various shooting angles like in this image (matlab functions like *roipoly* and *bwmorph(...,'dilate')* may be useful).

Blurring: Regions of the image further away from the DOF should have more blur. In this part of the assignment you can explore the effect of combining gaussian filters. For example, in class we learned that the convolution of two gaussians is also (magically) another gaussian. Why would we want that? We can increase the blurriness of a region by applying consecutive gaussians to it (matlab hint: fspecial is your friend).

Feel free to use any method of defining the depth planes and various parameters for the filters. Manipulating colors for the final version of the images drastically increases the illusion of a miniature. You can convert the image to a hue saturation value format (*rgb2hsv*) and directly increase by a percentage the saturation channel. To convert the image back use *hsv2rgb*.

Your final code should be able to, given an input image, apply blur filters and color adjustments such that the end result resembles more a miniature model than a real scene. Of course, some scenes, with multiple, clearly delineated, depth planes, are better suited to this type of manipulations. The images we have provided have been successfully used to create the fake miniature effect. Choose the rest the pictures for the assignment wisely. Since you have the freedom to define your own subject we expect grandiose results (pun intended). For your inspiration check out some of the examples on flickr or here. Finally, here is a good starting point of what your program should be able to do.

Bells & Whistles (Extra Credit)

Try defining non-horizontal depth planes either by masking the foreground and adding it in last or by coming up with an altogether new method, and increase your score:

- (**4.76 points**) Take some real tilt shift pictures with a camera with a decent lens (if you can find one) and compare results.
- (**10 points**) Use object masks to define complex DOF regions .
- (**25 points**) Take a series of pictures from a high vantage point (or anywhere for that matter) and create fake stop motion animations by putting your frames together in a short clip/gif. Great examples here.
- (**20 points**) Improve the results using edge preserving filters. Some people have already done so.

Deliverables

The project will be graded out of 100 points, with points allocated as noted below and above. Use the `proj2/www` and `proj2/code` folders, as described in the submission instructions.

Code: Please supply the all the code you used to generate your results, along with a `README` file or sufficient comments to allow us to test it out. (Basic miniature faking with horizontal and vertical depth planes for the 3 images provided is worth **50 points** for those in the undergrad version of the class, **30**

points for the grad version. On top of this base score, you will earn points for your report [see **Web** below].)

Web: Create a project page to showcase your results. By now you know the drill: simple html, nothing too complicated. Please:

- (**45 points**) Include a couple (4 or more) great examples of images (2 taken by YOU and 2 from the internet, but remember to attribute!). Show us your implementation works.
- (***N* points**) Dazzle us with any bells and/or whistles you've added in.
- Make sure to tell your story with weighty words as well as potent pictures.