Table of Contents

Acquire pictures and point correspondences

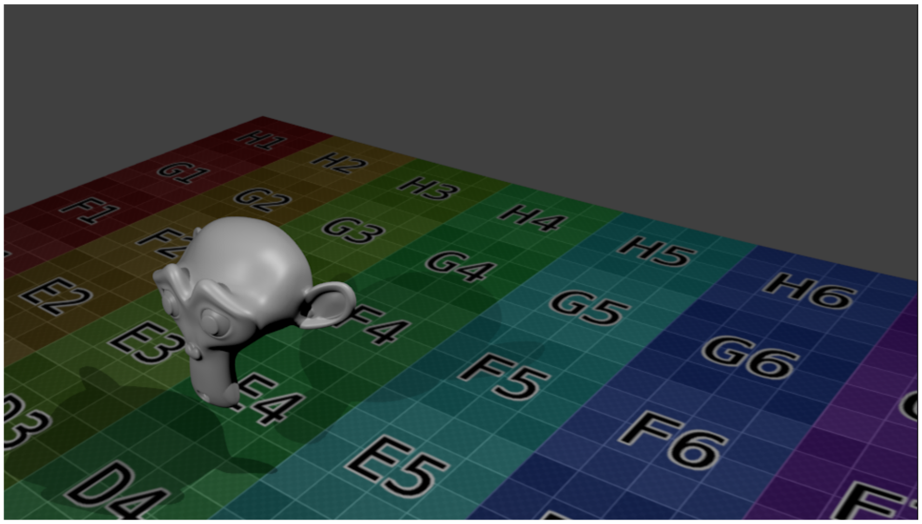
```
im1 = imread('./rendered_scene/untitled.png');
im2 = imread('./rendered_scene/untitled2.png');

% Add alpha channel:
im1(:,:,4) = ones(size(im1(:,:,1)));
im2(:,:,4) = ones(size(im2(:,:,1)));

% Ideally this would be done automatically via something like:
%im1_pts, im2_pts = findCorrespondences(im1, im2);

im1_pts = [294,117; 393,310; 270,420; 350,465; 83,260];
im2_pts = [767,127; 842,340; 679,417; 755,476; 542,230];
```

Convert images and point correspondences into cylindrical coordinates

I'm doing a quick and dirty conversion to create images in cylindrical coordinates at this early step. I loose some detail in the conversion process.

```
dist = 1500; % Focal length
im1 = projectImage(im1,dist);
im2 = projectImage(im2,dist);

% I know that im1 and im2 have mostly the same values, so I can get
 away
% with calculating thetaPerPixel once:
imsize = size(im1);
thetaRange = mapCylindrical(imsize(2),dist); % Technically should go
 from -theta/2 to theta/2
thetaPerPixel = thetaRange/imsize(2);


im1_pts(:,1) = mapCylindrical(im1_pts(:,1),dist)/thetaPerPixel;
im2_pts(:,1) = mapCylindrical(im2_pts(:,1),dist)/thetaPerPixel;

% Here's a figure of one of the images (now in cylindrical
 coordinates):
```

```
imshow(im1(:,:,1:3)/255);
```



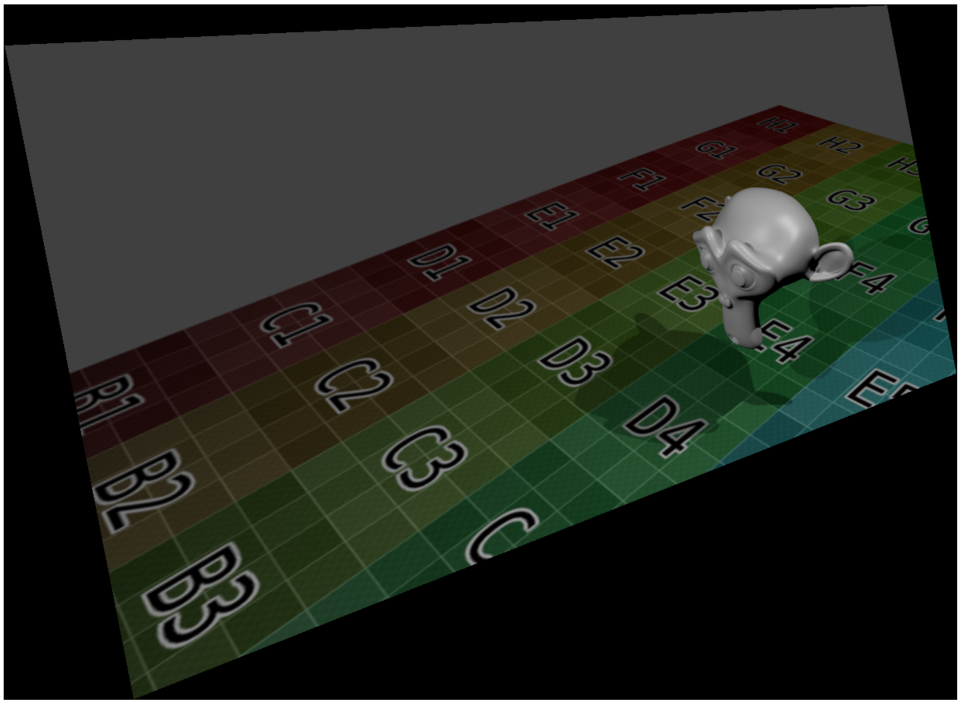
Recover homographies by picking point correspondences

```
% H transforms im2_pts into im1_pts (more or less)
H = computeH(im2_pts, im1_pts); % Note: H is row-major.
% H also acts on the images assuming that 0,0 is the upper left hand
 corner
% and points are x,y such that x,0 is on the top row of the image.

H
```

```
H =

    1.5200    0.3021 -733.1094
   -0.0654    1.5754   37.4771
    0.0008    0.0000    1.0000
```

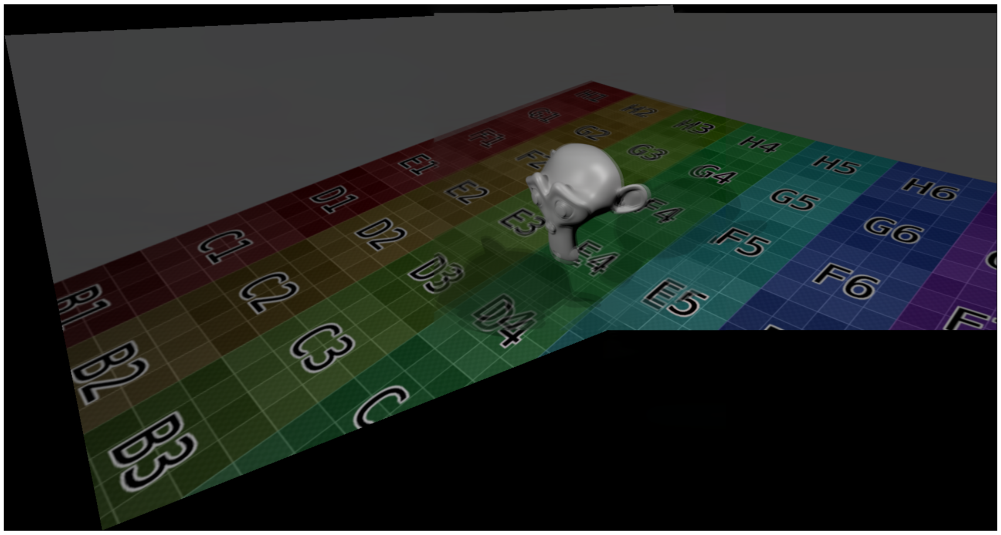
Warp the images

```
[imwarped, xoffset, yoffset] = warpImage(im2, H); % Note: imwarped has
 an alpha channel

imshow(imwarped(:,:,1:3)/255);
```



Blend images into a mosaic

```
ims = {im1, imwarped};
offsets = [0,0;xoffset,yoffset];
img = blendImage(ims, offsets);

imshow(img(:,:,1:3)/255);
```



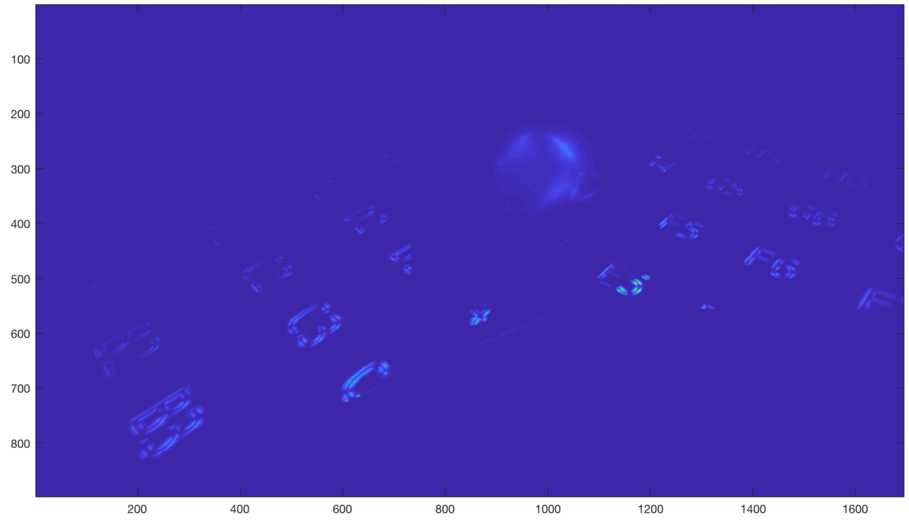
>= 3 Bells and Whistles

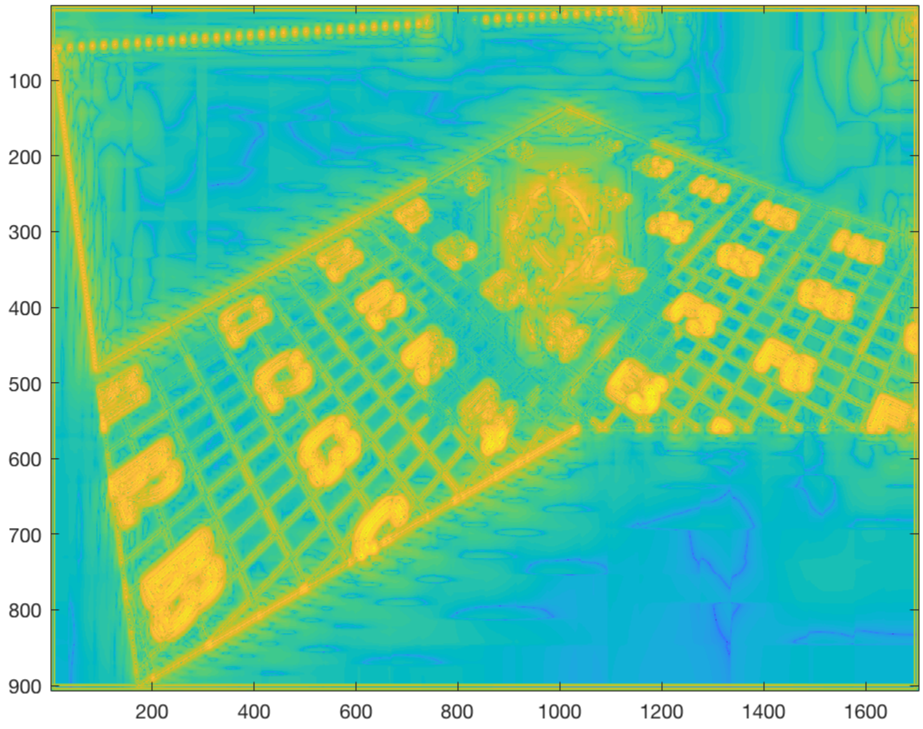
- Laplacian pyramid blending (at the bottom of blendImage.m)

- Cylindrical coordinate mapping (above and in projectImage.m)

- I'm partway through a method of finding correspondences between images I've got a Harris Corner detector working.

```
results = findFeatures(double(sum(img(:,:,1:3), 3)));

% Log2 of the results also looks pretty :-P
% (though we're trying to suppress the intermediate values that log2
 really
% brings out)
figure();imagesc(real(log2(results)));
```





Coolest thing I learned

I knew nothing about Laplacian pyramid blending. It's pretty cool. Also, I will now never forget that images in Matlab are y,x ordered.

```
function theta = mapCylindrical(x,dist)
% Dumb helper function for mapping a point on a plane at distance
% `dist`into cylindrical coordinates.
theta = asin(x/dist);
end
```

Published with MATLAB® R2017a