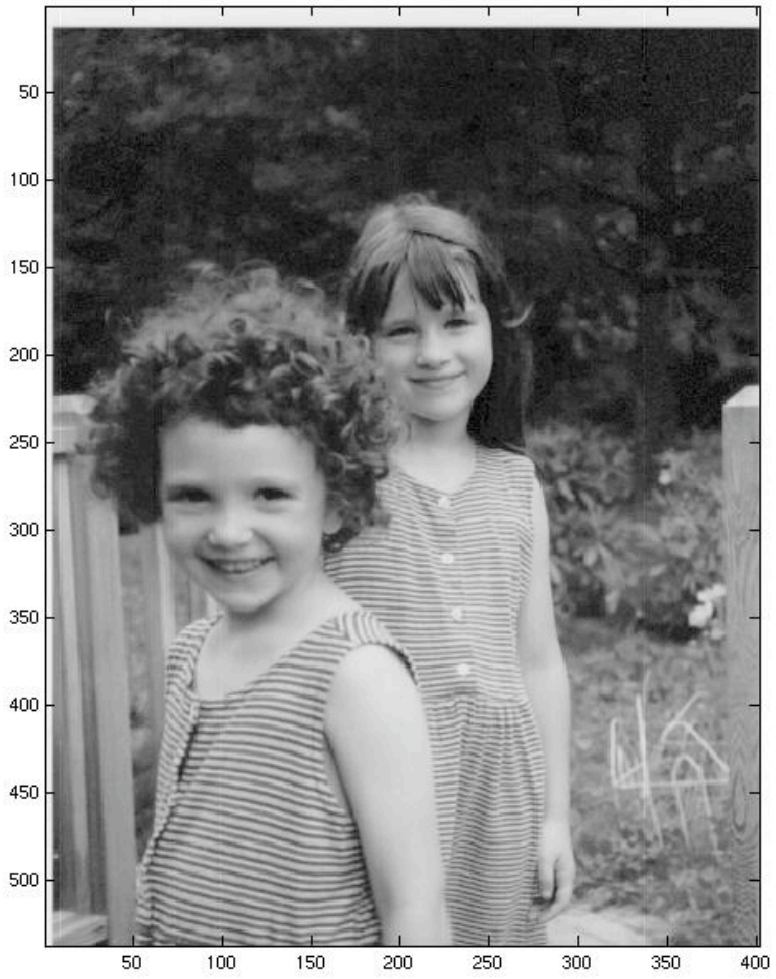
max van kleek

6.869 problem set zero

problem one

```
original = im2double(imread('data/original.tif'));
whitefield = im2double(imread('data/whiteField.tif'));

corrected = original./whitefield;
oldcmap = colormap;
colormap(repmat([0:0.01:1]',1,3));

imagesc(corrected.^.05);
```

result:



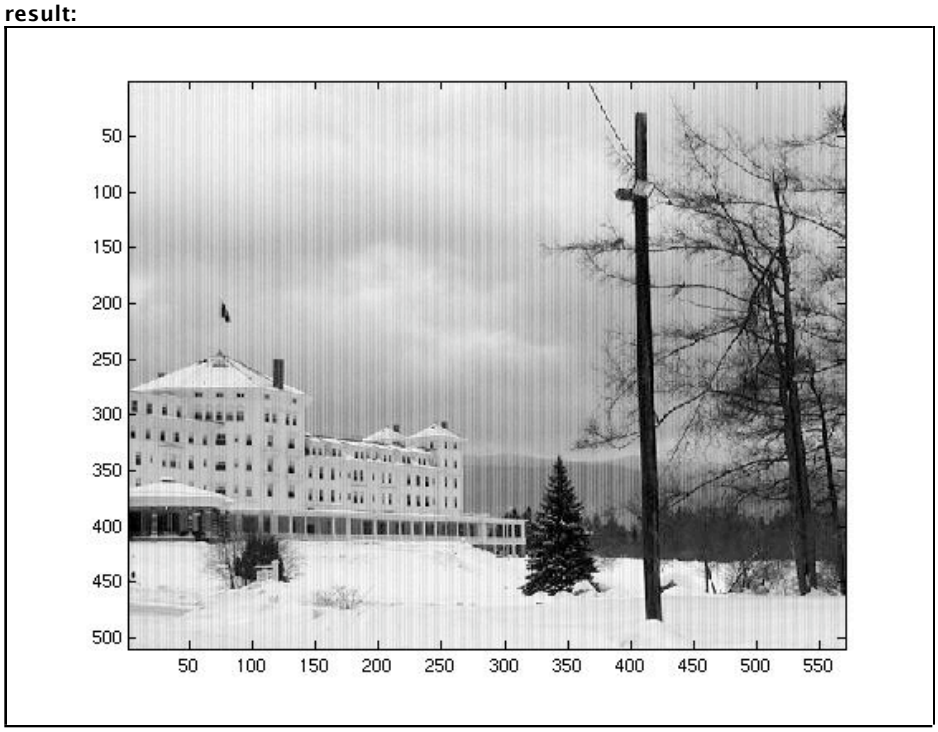
problem two

part a) simulate an RGB filter

```
% simulate rgb filter
```

```
bt = imread('data/brettan.tif');
bt = im2double(bt);

mask = repmat(permute(eye(3),[3 2 1]),size(bt,1),floor(size(bt,2)/3));

% cut off rounding error
bt=bt([1:size(bt,1)],[1:(3*floor(size(bt,2)/3))],:);

btrgb = sum(bt .* mask,3);

cmap = colormap;  % stow away the old cmap
colormap(repmat([0:0.001:1]',1,3));
imagesc(btrgb);
```

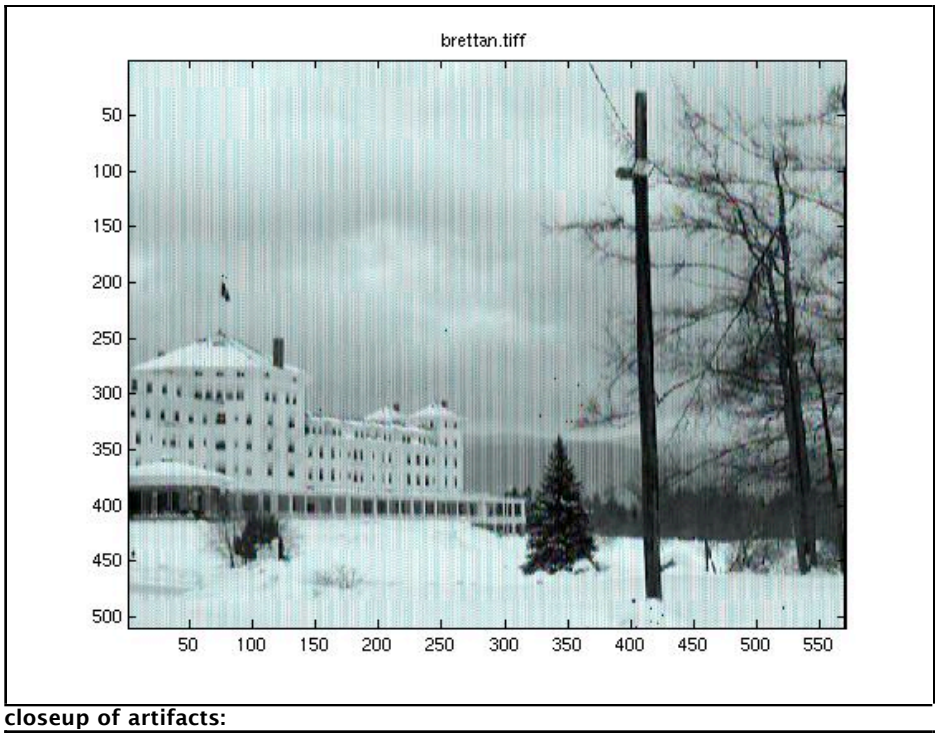
result:



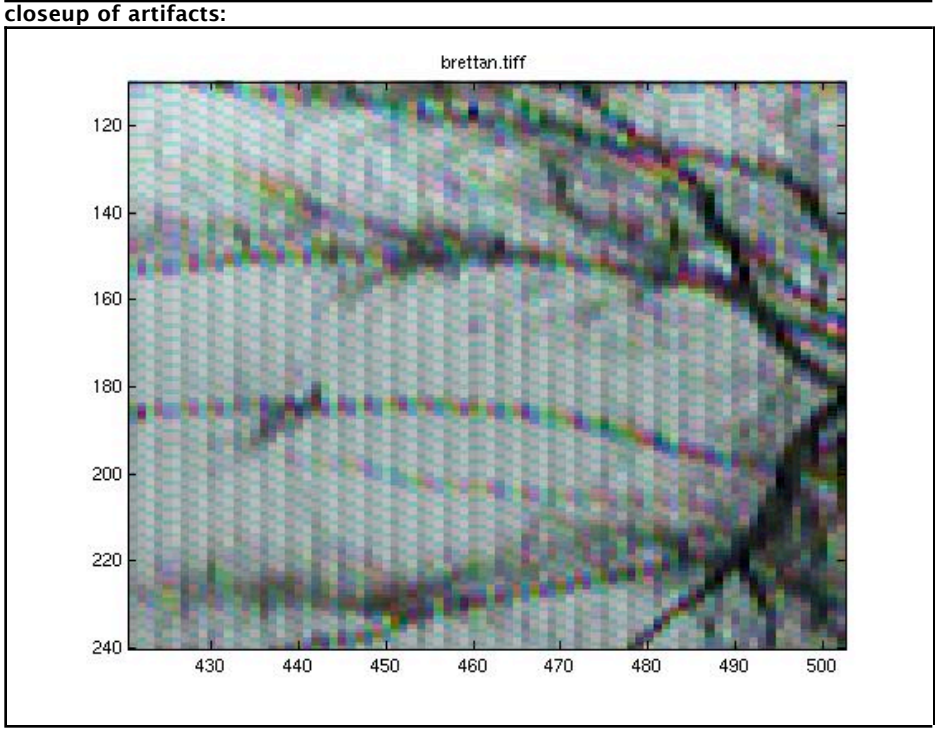
part b) linear interpolation of colors to reconstruct color

```
btvr = zeros(size(btrgb));
btvg = zeros(size(btrgb));
btvb = zeros(size(btrgb));

ridx = find(mod(find(btrgb),3)==1);
ridx = ridx([1:3*floor(length(ridx)/3)]);
gidx = ridx + 1;
bidx = ridx + 2;

btvr(ridx)=btrgb(ridx);
btvr(ridx+1)=2.0/3.0*btrgb(ridx) + 1.0/3.0*btrgb(ridx+3);
btvr(ridx+2)=1.0/3.0*btrgb(ridx) + 1.0/3.0*btrgb(ridx+3);
btvg(gidx)=btrgb(gidx);
btvg(gidx+1)=2.0/3.0*btrgb(gidx) + 1.0/3.0*btrgb(gidx+3);
btvg(gidx+2)=1.0/3.0*btrgb(gidx) + 2.0/3.0*btrgb(gidx+3);
btvb(bidx)=btrgb(bidx);
btvb(bidx+1)=2.0/3.0*btrgb(bidx)+1.0/3.0*btrgb(bidx+3);
btvb(bidx+2)=1.0/3.0*btrgb(bidx)+2.0/3.0*btrgb(bidx+3);
bt = cat(3,btvr,btvg,btvb);
figure(1);
imagesc(bt);
title('brettan.tiff');
```

result:

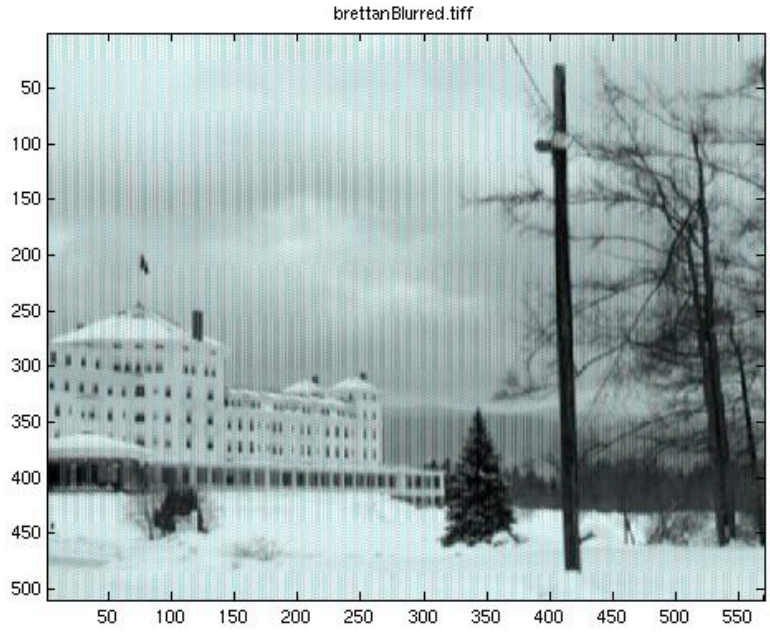


closeup of artifacts:

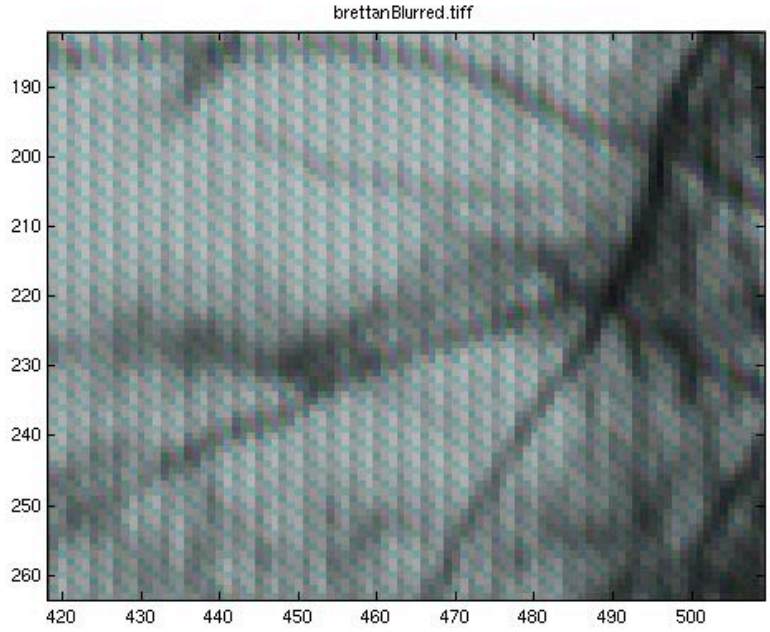


now try the same process with blurred version:

result:



closeup of blurred:



definitely fewer artifacts, but we also see overall general loss of sharpness.

problem three: object metamorphosis

part a) warp each image to the average shape:

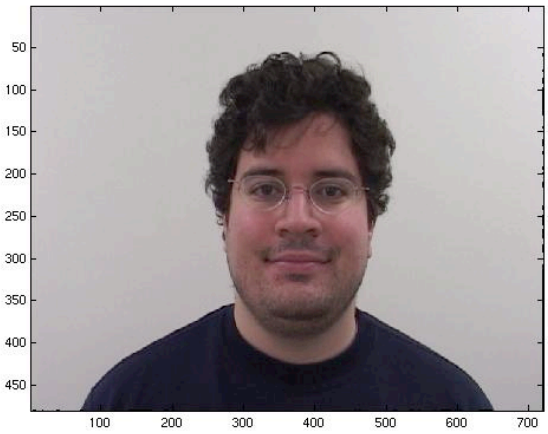
```
sm = imread('data/smile.jpg');
sm = im2double(sm);
load('data/flowNS.mat');
xcoo = meshgrid([1:size(sm,2)],[1:size(sm,1)])  + 0.5*flowNS(:,:,1);
ycoo = meshgrid([1:size(sm,1)],[1:size(sm,2)])' + 0.5*flowNS(:,:,2);
xs=meshgrid([1:720],[1:480]);
ys=meshgrid([1:480],[1:720])';
for i=1:3, mario(:,:,i) = interp2(xs,ys,sm(:,:,i),xcoo,ycoo); end;
```

```
figure(6);
imagesc(mario);
```

results:

neutral smile

average:



part b) compute a smooth transition between images

a function to make it more convenient to compute intermediate images:

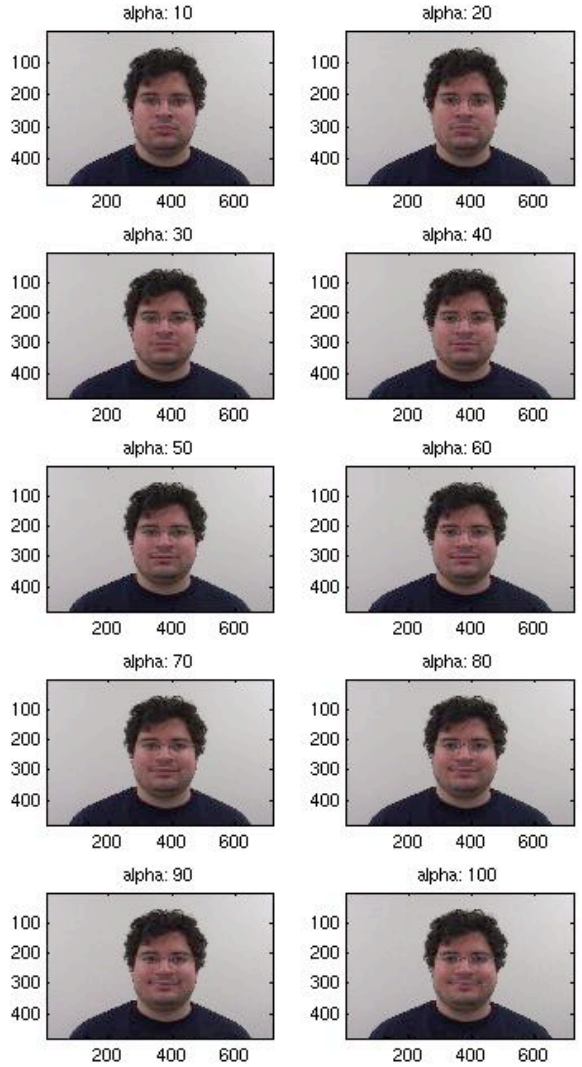
```
function img = morph(src1,src2,fv1,fv2,alpha)
% alpha <= 0 <= 1.0
% returns an image representing flowvec from src1->src2
% assumes src1 and src2 are the same dimension

img = [];

xmesh = meshgrid([1:size(src1,2)],[1:size(src1,1)]);
ymesh = meshgrid([1:size(src1,1)],[1:size(src1,2)])';

xc1 = xmesh + alpha*fv2(:,:,1);
yc1 = ymesh + alpha*fv2(:,:,2);
xc2 = xmesh + (1.0-alpha)*fv1(:,:,1);
yc2 = ymesh + (1.0-alpha)*fv1(:,:,2);

for i=1:3,
    img(:,:,i) = (1.0-alpha)*interp2(xmesh,ymesh,src1(:,:,i),xc1,yc1) + ...
       (alpha)*interp2(xmesh,ymesh,src2(:,:,i),xc2,yc2);
end;
% nuke the nans!
img(isnan(img)) = 0;
```

then all we have to do is invoke it:

```
sm = imread('data/smile.jpg');
sm = im2double(sm);
nt = imread('data/neutral.jpg');
nt = im2double(nt);
load('data/flowNS.mat');
load('data/flowSN.mat');
for i=1:10,
        subplot(5,2,i);
```

```
        imagesc(morph(nt,sm,flowNS,flowSN,1.0/10.0*i));
    end;
```

result:



problem four: background subtraction

first make a helper function for calculating the mahanoboulis function:

```
function d = mahanoboulis(xs, mean, cov)
% calculates the mahanoboulis distance of x from mean/cov
% xs should be row-per-observation, column per variable
% mean should be a row vector

means = repmat(mean,size(xs,1),1);
size(means)
size(cov)
di = (xs-means);
size(di)
da = di*pinv(cov);

N = size(xs,1);
```

```
for j=1:3,
   dz(j,:) = (da(:,j).*di(:,j))';
end

d = sum(dz)';
```

and a function that will help us vectorize the image into a single vector of [R G B] columns:

```
function p = picpermute(imagedata)

%p = reshape(permute(imagedata, [1 3 2]),[size(imagedata,1)*size(imagedata,2) 3 1]);
p(:,:,1) = reshape(imagedata(:,:,1),[size(imagedata,1)*size(imagedata,2) 1 1]);
p(:,:,2) = reshape(imagedata(:,:,2),[size(imagedata,1)*size(imagedata,2) 1 1]);
p(:,:,3) = reshape(imagedata(:,:,3),[size(imagedata,1)*size(imagedata,2) 1 1]);

p = permute(p,[1 3 2]);
```

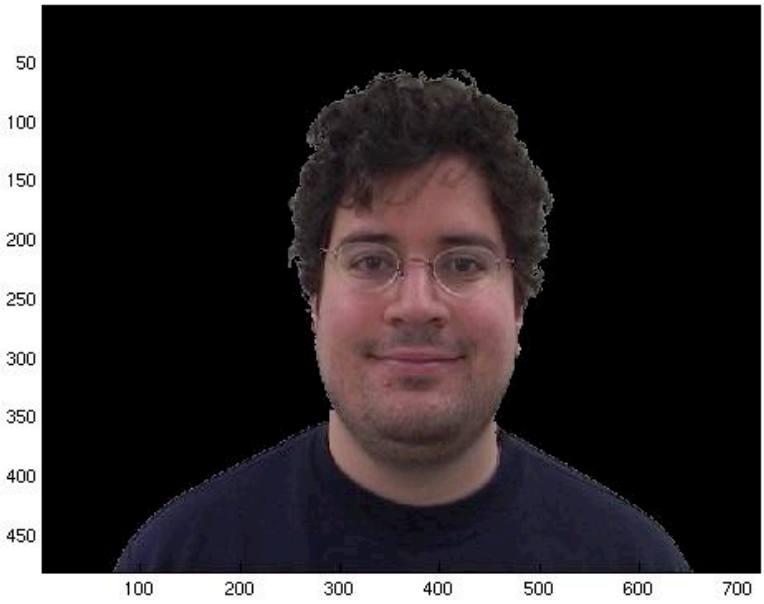
similarly, a function to un-vectgorize the image back to color planes:

```
function smm = unpicpermute(imgdata, originalimg)
% takes a column (r g b) linear space

imgdata = permute(imgdata, [1 3 2]);
planesize = size(originalimg(:,:,1));
sm1 = reshape(imgdata(:,:,1),planesize);
sm2 = reshape(imgdata(:,:,2),planesize);
sm3 = reshape(imgdata(:,:,3),planesize);
smm = cat(3,sm1,sm2,sm3);
```

putting the pieces together:

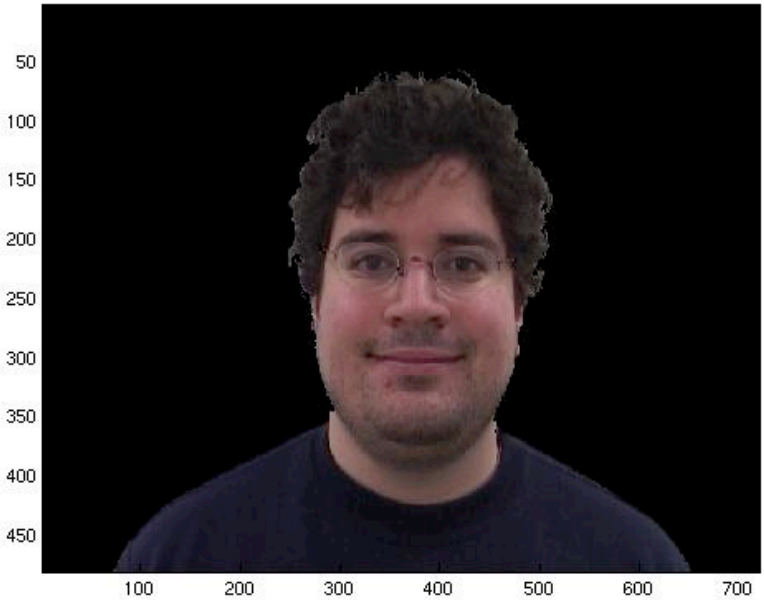
```
%% backgrounds
% first grab some of the background pixeels
sm = imread('data/smile.jpg');
sm = im2double(sm);
bkpix = sm([1:350],[1:241],:);
bkpixp = picpermute(bkpix);
bkmean = mean(bkpixp);
bkcov=cov(bkpixp);

smp = picpermute(sm);
bkidx = find(mahanoboulis(smp,bkmean,bkcov)<200);
smp(bkidx,:,:) = repmat([0 0 0],size(bkidx,1),1);
smnew = unpicpermute(smp,sm);
imagesc(smnew);
```

result:



applying some erosion:

```
se = strel('line',2,90);
imagesc(imerode(smnew,se));
```



a function to help us turn it into a movie:

```
function movie = ps0movie(img1,img2,bkimg,fv1,fv2,frames)
frame = 1;
sequence = [];
bkimgp = picpermute(bkimg);
for alpha=0:(1.0/frames):1,
    img = morph(img1,img2,fv1,fv2,alpha);

    nans = isnan(img);
    img(nans) = 0;
```

```
            bkpix = img1([1:350],[1:241],:);
            bkpixp = picpermute(bkpix);
            bkmean = mean(bkpixp);
            bkcov=cov(bkpixp);

            smp = picpermute(img);
            bkidx = find(mahanoboulis(smp,bkmean,bkcov)<200);
            smp(bkidx,:,:) = bkimgp(bkidx,:,:);
            img = unpicpermute(smp,img);

            %max(img)
            %max(max(img))
            %maximgg = max(max(max(img)))
            %minimgg = min(min(min(img)))
            %size(img)
            %im2frame(img,colormap)
            sequence(:,:,:,frame) = img; % /(max(max(max(img))));

           % figure(frame);
           % imagesc(img);
            frame = frame + 1;

        end;
        movie = images2movie(sequence);
```

results: