In[3]:=
Day6 – Nathan Yee

Initalization

In[4]:=
```
SetDirectory[NotebookDirectory[]]
```
Out[4]=
```
/home/nathan/QEA-Homework/module2/day6
```

Functions

Kinds of Correlations in an Image Set

In[9]:=
```
alpacas = Import["/home/nathan/QEA-Homework/module2/day6/*.jpg"];
alpacas = ColorConvert[alpacas, "Grayscale"];
alpacas = Table[ImageResize[alpacas[[i]], {200, 200}], {i, Length[alpacas]}]
dataAlpacas = Table[ImageData[alpacas[[i]]], {i, Length[alpacas]}];
```
Out[11]=


Question 1

In[13]:=
```
Dimensions[dataAlpacas]
```
Out[13]=
```
{5, 200, 200}
```

Question 2

Size nxn

Question 3

In[14]:=
```
Dimensions[ACorrelation[dataAlpacas[[1]]][[1]]]
```
Out[14]=
```
{200, 200}
```

In[15]:=
```
dataAlpacas = Table[ImageData[i], {i, alpacas}];
Dimensions[dataAlpacas]
happy = Table[Flatten[dataAlpacas[[i]], 1], {i, Length[dataAlpacas]}]
Dimensions[happy]
```
Out[16]=
```
{5, 200, 200}
```

Out[17]=
```
{{0.447059, 0.403922, 0.337255, 0.298039, ··· 39 992 ···,
  0.686275, 0.560784, 0.541176, 0.654902}, ··· 3 ···, {··· 1 ···}}
```
large output | show less | show more | show all | set size limit...

Out[18]= {5, 40 000}

In[19]:=
```
ACorrelation[Transpose@happy]
```

Out[19]//MatrixForm=

1.	−0.102891	0.153491	0.0106476	−0.365763
−0.102891	1.	0.00543687	0.143584	0.192929
0.153491	0.00543687	1.	0.167695	−0.216173
0.0106476	0.143584	0.167695	1.	0.114689
−0.365763	0.192929	−0.216173	0.114689	1.

Question 4

Expect a high correlation somewhere for the opposite column.

Test your understanding...

Question 1

Data

In[20]:=
```
imageData = Import["/home/nathan/QEA-Homework/module2/classdata.mat"][[1]]
```

Out[20]=
```
{{··· 1 ···}, {··· 1 ···}, {··· 1 ···}, {··· 1 ···}, {··· 1 ···},
 {··· 1 ···}, {··· 1 ···}, ··· 330 ···, {··· 1 ···}, {··· 1 ···},
 {··· 1 ···}, {··· 1 ···}, {··· 1 ···}, {··· 1 ···}, {··· 1 ···}}
```
large output | show less | show more | show all | set size limit...

In[21]:=
```
Dimensions[imageData]
```

Out[21]= {344, 360, 256}

Question 2

In[22]:=
```
indexList = {53, 133, 205, 253, 302};
images = Map[Image, imageData[[indexList]]]
resized = Table[ImageResize[images[[i]], {200, 200}], {i, Length[images]}]
```

Out[23]=

Out[24]=

In[25]:=
```
data = Table[ImageData[i], {i, images}];
Dimensions[data]
sad = Table[Flatten[data[[i]], 1], {i, Length[data]}]
```

Out[26]=
```
{5, 360, 256}
```

Out[27]=
```
{ ··· 1 ··· }
```
large output | show less | show more | show all | set size limit...

In[28]:=
```
correlation = ACorrelation[Transpose@sad]
```

Out[28]//MatrixForm=

$$\begin{pmatrix} 1. & 0.523229 & 0.683955 & 0.343091 & 0.270954 \\ 0.523229 & 1. & 0.588773 & 0.463517 & 0.548209 \\ 0.683955 & 0.588773 & 1. & 0.468519 & 0.379433 \\ 0.343091 & 0.463517 & 0.468519 & 1. & 0.463523 \\ 0.270954 & 0.548209 & 0.379433 & 0.463523 & 1. \end{pmatrix}$$

In[29]:=
```
MatrixForm[sad][[1]][[1]]
```

Out[29]=

```
{1.055, 1.055, 1.07, 1.07, 1.06, 1.02, 1.025, 1.045, 1.05, 1.035,
 1.025, 1.025, 1.03, 0.98, 0.845, ··· 92130 ···, 0.71, 0.71, 0.72, 0.72,
 0.71, 0.71, 0.72, 0.78, 0.79, 0.81, 0.83, 0.835, 0.82, 0.82, 0.79}
```

large output | show less | show more | show all | set size limit...

Size nxn

From Data to Dimensions

Quesion 1

5 rows, 1 column. (5x1)

Quesion 2

$\left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}{}^{2}\right)^{\frac{1}{2}}$

Quesion 3

$\left(\sum_{i=1}^{m}\sum_{j=1}^{n} (a_{ij} - b_{ij})^{2}\right)^{\frac{1}{2}}$

Quesion 4

$A^T B$ corresponds tells how parallel the matricies are. I'm not sure how this is useful. It seems like I would want to use this dot product to do a bunch of the subtraction/addition for me.

Quesion 5

In[30]:=

```
indexList = {233, 237, 133}
threeFacesData = Table[imageData[[i]], {i, indexList}]
threeFaces = Map[Image, imageData[[indexList]]]
```

Out[30]=

```
{233, 237, 133}
```

Out[31]=

```
{{··· 1 ···}, {··· 1 ···}, {··· 1 ···}}
```

large output | show less | show more | show all | set size limit...

Out[32]=

In[33]:=
```
diff = Flatten[Transpose@{{threeFacesData[[1]] - threeFacesData[[2]]},
    {threeFacesData[[2]] - threeFacesData[[3]]},
    {threeFacesData[[3]] - threeFacesData[[1]]}}, 1]
```

Out[33]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

In[34]:=
```
Dimensions[diff]
```

Out[34]=
```
{3, 360, 256}
```

In[35]:=

In[36]:=
```
squared = diff^2
```

Out[36]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

In[37]:=
```
Total[squared[[1]], 2]
Total[squared[[2]], 2]
Total[squared[[3]], 2]
```

Out[37]=
```
1426.38
```

Out[38]=
```
10 815.
```

Out[39]=
```
11 731.9
```

These differences make sense because the two images of me are very similar to one another compared to john.

In[40]:=

Smoothing and Downsampling

Out[40]=

Smoothing and Downsampling

In[41]:=
```
index = {233};
face = imageData[[index]][[1]];
Image[face]
```

Out[43]=

In[44]:=
```
k1 = {{0, 0, 0}, {0, 1, 0}, {0, 0, 0}};
k2 = {{1/9, 1/9, 1/9}, {1/9, 1/9, 1/9}, {1/9, 1/9, 1/9}};
Image@Kernel2D[face, k1]
Image@Kernel2D[face, k2]
```

Out[46]=

Out[47]=

In[48]:=
```
n = 1.3;
down = Table[face[[Floor[n * i], Floor[n * j]]], {i, Floor[Dimensions[face][[1]] / n]},
   {j, Floor[Dimensions[face][[2]] / n]}, n = 1.3];
Image[
 down]
```

Out[50]=

In[51]:=
```
Dimensions@face
```

Out[51]=
```
{360, 256}
```

In[52]:=
```
s = .05
ImageResize[Image[face], {s * 256, s * 360}]
```

Out[52]=
```
0.05
```

Out[53]=

downsizing can take average values (kernel operation) and reuturn those to get a nicer looking downsize.

In[139]:=
Image rotation and scaling (example code taken from Eric Miller) (which is taken from ShellforImTrans.m, Cleaning Up Image Processing)

Out[139]=
```
Image rotation and scaling (example
  code taken from Eric Miller) (which
  is taken from ShellforImTrans.m,
  Cleaning Up Image Processing)
```

Conversion functions (double click at right to expand)

Because images are lists of rows, this intensities list traverses one row at a time.

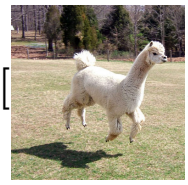
In[140]:=
```
Clear@unconvert
unconvert[tranxypos_, intensities_] := Module[{m, n, xytran, imageData},
  {m, n} = ImageDimensions[picture];
xytran = Transpose[Ceiling[tranxypos]];
  xytran[[All, 3]] = intensities;
xytran = Cases[xytran, {x_, y_, _} /; (1 ≤ x ≤ m && 1 ≤ y ≤ n)];
  imageData = ConstantArray[0, {n, m}];
Table[
   imageData[[i[[2]], i[[1]]]] = i[[3]];
   , {i, xytran}];
  imageData
 ]
```

In[142]:=
```
Clear[convert];
convert[picture] := Module[{m, n, intensities, xydata},
  {m, n} = ImageDimensions[picture];
  (* intensities is put into a global variable,
  which isn't optimal, but works *)
  intensities = Flatten[ImageData[ColorConvert[picture, "Grayscale"]]];
(* m is width, n is height *)
  xydata = Transpose@Flatten[Table[{x, y, 1}, {y, 1, n}, {x, 1, m}], 1];
  {xydata, intensities}
 ]
```

Generate a list of pixel positions

Your code

In[144]:=
```
(*Load your own picture using the Import command or Control-V,
or use the Transformer file here*)
(*Import["somefile.png"];*)
```

In[145]:=
```
picture = ColorConvert[
```

```
, "Grayscale"];
```

In[146]:=
```
{xypos, intensities} = convert[picture];
Dimensions[xypos]
Dimensions[intensities]
```

Out[147]= {3, 144 399}

Out[148]= {144 399}

Here, transform those positions into new ones

In[149]:=
```
ImageDimensions[picture]
```

Out[149]= {379, 381}

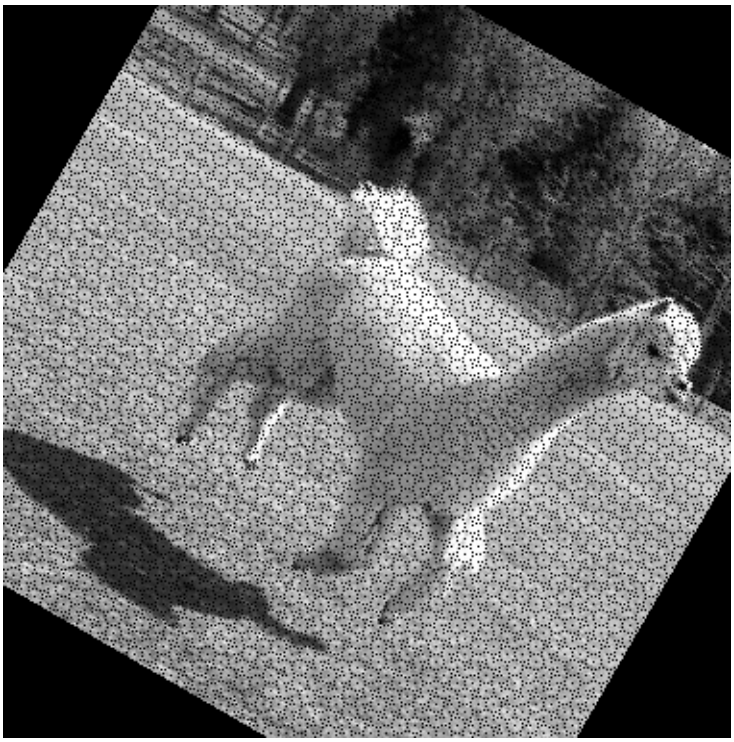
In[150]:=
```
θ = 30 Degree;
r = {{Cos[θ], -Sin[θ], 0}, {Sin[θ], Cos[θ], 0}, {0, 0, 1}};
t = {{1, 0, 380/2}, {0, 1, 380/2}, {0, 0, 1}}
s = {{1, 1, 0}, {0, 1, 0}, {0, 0, 1}}
```

Out[152]= {{1, 0, 190}, {0, 1, 190}, {0, 0, 1}}

Out[153]= {{1, 1, 0}, {0, 1, 0}, {0, 0, 1}}

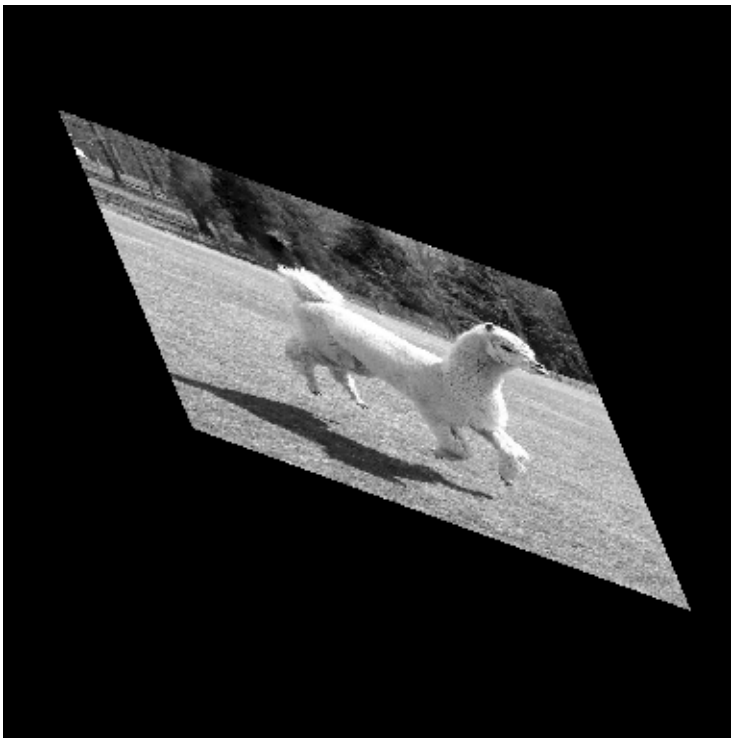
In[154]:=
```
tranxypos = t.r.Inverse[t].xypos;
Image[unconvert[tranxypos, intensities]]
```

Out[155]=



In[156]:=
```
tranxypos =
{{1, 1, 0}, {0, 1, 0}, {0, 0, 1}}.{{1/2, 0, 0}, {0, 1/2, 0}, {0, 0, 1}}.{{Cos[30 Degree], -Sin[30 Degree], 0}, {Sin[30 Degree], Cos[30 Degree], 0}, {0, 0, 1}}.{{1, 0, 10}, {0, 1, 120}, {0, 0, 1}}.xypos;
Image[unconvert[tranxypos, intensities]]
```

Out[157]=



Other Image Processing

In[158]:=
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} \text{ // MatrixForm}$$

Out[158]//MatrixForm=
$$\begin{pmatrix} 10 & 11 & 12 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

The transformation matrix moves the bottom row to the top of the matrix.

In[159]:=
```
Dimensions[face]
```

Out[159]=
```
{360, 256}
```

In[160]:=
```
n = 360;
r = DiagonalMatrix[Table[1, n], -1, n] + DiagonalMatrix[{1}, n - 1, n]
```

Out[161]=

{ ··· 1 ··· }

large output | show less | show more | show all | set size limit...

In[162]:=
```
rn = r
For[i = 0, i < 150, i++, rn = rn.r]
Image[rn.face]
```

Out[162]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

Out[164]=

Detects in horizontal detection because it is comparing in that direction.

In[165]:=
```
Dimensions@DiagonalMatrix[Table[-1, n - 1], 1]
Dimensions@DiagonalMatrix[Table[1, n], 0]
Dimensions@DiagonalMatrix[{-1}, 1 - n, n]
```

Out[165]= {360, 360}

Out[166]= {360, 360}

Out[167]= {360, 360}

In[168]:=
```
e1 = DiagonalMatrix[Table[-1, n - 1], 1] +
  DiagonalMatrix[Table[1, n], 0] + DiagonalMatrix[{-1}, 1 - n, n]
Image[e1.rn.face]
```

Out[168]=

{ … 1 … }

large output | show less | show more | show all | set size limit...

Out[169]=

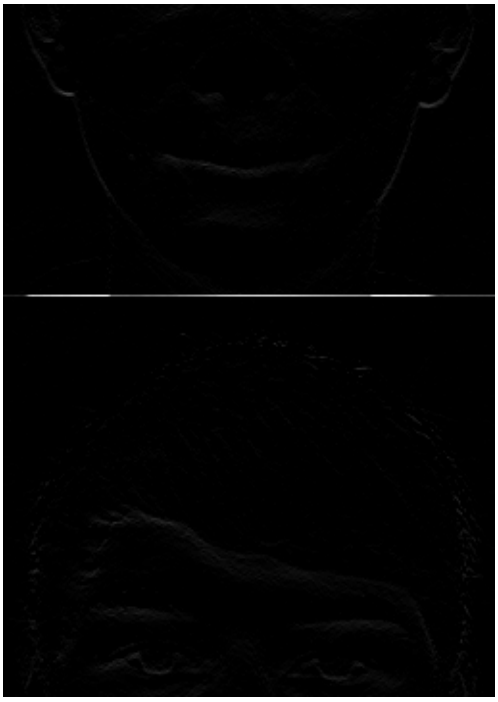
In[170]:=
```
e2 = DiagonalMatrix[Table[-1, n - 1], -1] +
  DiagonalMatrix[Table[1, n], 0] + DiagonalMatrix[{-1}, n - 1, n]
Image[e2.rn.face]
```

Out[170]=

{ ··· 1 ··· }

large output | show less | show more | show all | set size limit...

Out[171]=

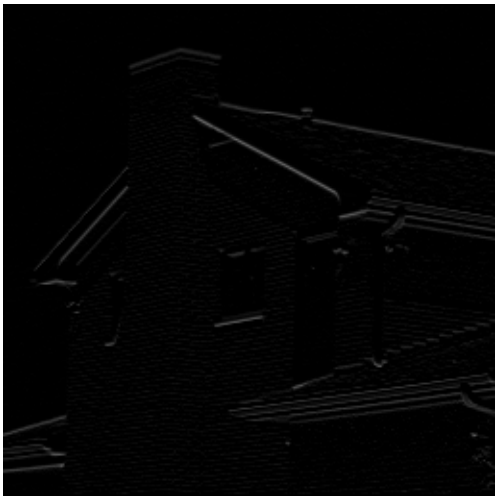


In[172]:=
```
house = ImageData[ ]
```

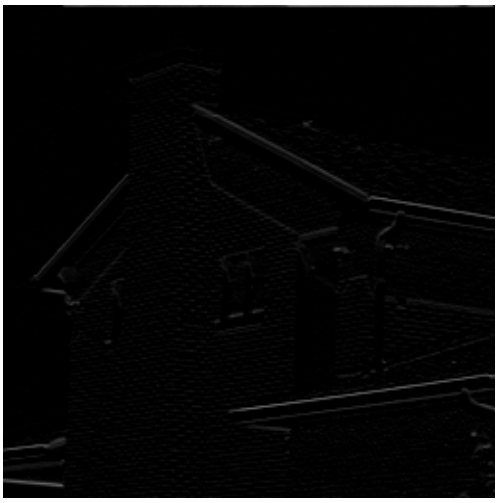

Out[172]=
```
{{0.737255, 0.733333, 0.733333, 0.729412, ··· 248 ···,
  0.741176, 0.737255, 0.741176, 0.741176}, ··· 254 ···, {··· 1 ···}}
```
large output | show less | show more | show all | set size limit...

In[173]:=
```
Dimensions[house]
```
Out[173]=
```
{256, 256}
```

In[174]:=
```
n = Dimensions[house][[1]]
e1 = DiagonalMatrix[Table[-1, n - 1], 1] +
   DiagonalMatrix[Table[1, n], 0] + DiagonalMatrix[{-1}, 1 - n, n];
Image[e1.house]
e2 = DiagonalMatrix[Table[-1, n - 1], -1] +
   DiagonalMatrix[Table[1, n], 0] + DiagonalMatrix[{-1}, n - 1, n];
Image[
 e2.
  house]
```
Out[174]=
```
256
```

Out[176]=


Out[178]=



In[179]:=
```
    -1  -1  -1
k = -1   8  -1
    -1  -1  -1

     0  -1   0
s = -1   5  -1
     0  -1   0

Image[Kernel2D[house, k]]
Image[Kernel2D[face, s]]
```

Out[179]= {{-1, -1, -1}, {-1, 8, -1}, {-1, -1, -1}}

Out[180]= {{0, -1, 0}, {-1, 5, -1}, {0, -1, 0}}

Out[181]=

Out[182]=

A first attempt at facial recognition

In[210]:=
```
avg = Total[Total[Total[imageData]]] / Length[imageData];
sumIntensity = Table[Total[Flatten[i], 1], {i, imageData}];
```

In[214]:=
```
numPixels = Dimensions[imageData][[2]] * Dimensions[imageData][[3]];
```

In[217]:=
```
brightnessScale = 1 - ((sumIntensity - avg) / numPixels);
```

In[227]:=
```
rescaledBrightness = brightnessScale * imageData
```

Out[227]=

{ ··· 1 ··· }

large output | show less | show more | show all | set size limit...

In[264]:=
```
Table[Image[i], {i, rescaledBrightness}]
```

Out[264]=