MxNet Notes

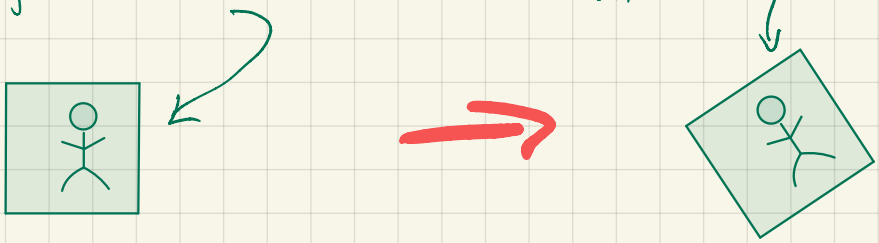
Rotating an Image

20220218

Rotating an Image

The goal is take an image such as this

And rotate it to something like this



A secondary goal is to use mxnet api as much as possible because

i) learn the api

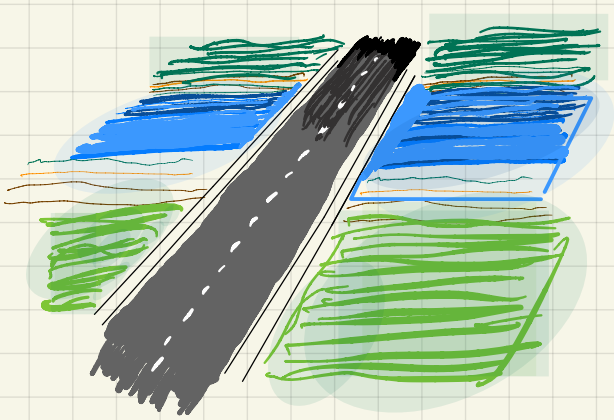
ii) speed, taking advantage of the gpu and the benefits of parallelism

Why rotate an image?

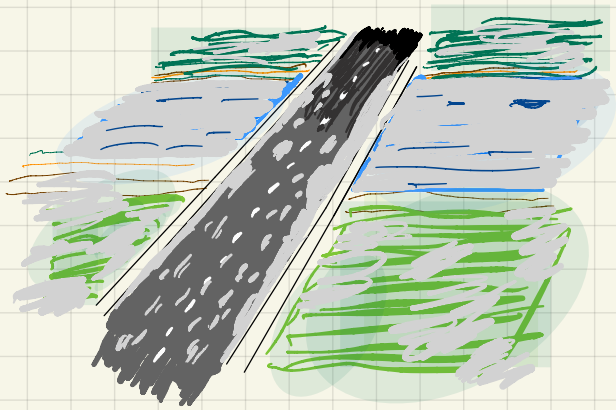
I want to use real world camera imagery of bridges in the icybridge hackathon project.

The auto ML models used as prototypes for locating the bridge, seam between bridge & roadway, and even ice failed miserably.

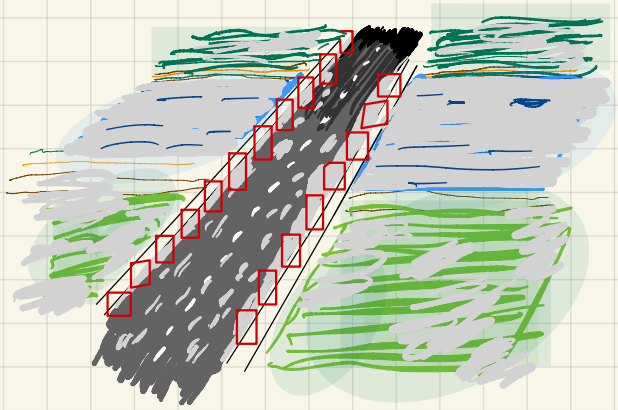
Part of this, was because the camera images were skewed.



Consider a photo of a bridge in normal conditions



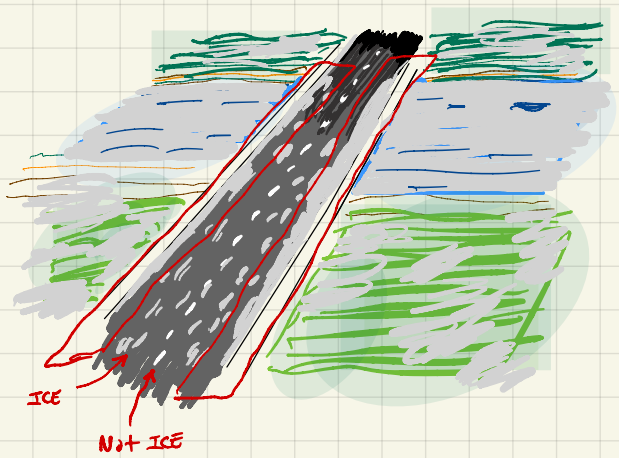
versus with ice



And with bounding boxes identifying ice

This causes problems!

The ice can't be identified with one simple bounding box. The bounding box API doesn't permit polygons. So to identify a diagonal this is not allowed.



Creating a "digitized" stair case also creates problems. The number of bounding boxes required to identify all the bounding boxes has a limit. It seems that's its a best practice to identify objects with multiple models and then combine the results.

Other problems include identifying patches of ice seprately from the dotted white lines identifying the passing zone.

So my conjecture is to rotate the image in order to identify the ice, bridge & seam with fewer bounding boxes.

Approach One

- How do you rotate an Image?
- How is an image stored?
- what API routines are available?

Lets look at how the image stored first. Understand the data first.

Lets consider a trivial 2x2 pixel image of 1-color. That is a simple black and white image. Not even grayscale.

Sample Image

A couple of notes about this image

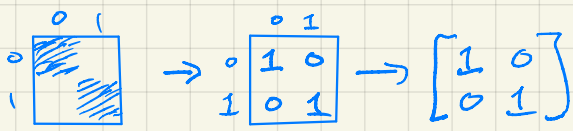
- It has two colors blue or white
- Its tiny. Just 2 pixels by 2 pixels

How would It be coded?

Ignoring file format which could use compression like JPEG or GIF.

How would it be coded in memory?

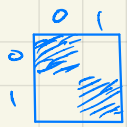
Images are stored in an array in memory.



So with a 2x2 image it could use 4x1 bits.

However, bits are typically stored in a byte and you have to access memory. Smallest addressable memory width is a byte.

If we are going to address each pixel as a byte:



$\rightarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 255 & 0 \\ 0 & 255 \end{bmatrix}$

4 pixels | black = 1, white = 0 | array w/ 1-bit encoding | array with byte encoding

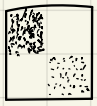
So, why 255?

Its the max value of:

8-bits → all ones → 0xFF → 255

What about the other values? 1-254?

Good Question. Its grayscale.

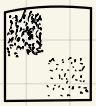


$\rightarrow \begin{bmatrix} 156 & 0 \\ 0 & 64 \end{bmatrix}$

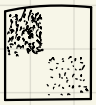
However that's not the way its actually coded.

Even though more dots, correspond with higher count and its intuitive.

It coded in reverse. The higher the number the less dark.



$\rightarrow \cancel{\begin{bmatrix} 156 & 0 \\ 0 & 64 \end{bmatrix}}$

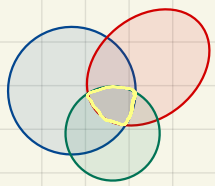


$\rightarrow \begin{bmatrix} 64 & 256 \\ 255 & 156 \end{bmatrix}$

why do that?

COLOR

Because color. Consider encoding a color image.



The highlighted area in yellow is a mix of Red, Green, and Blue.

Notice the highlighted area appears gray.

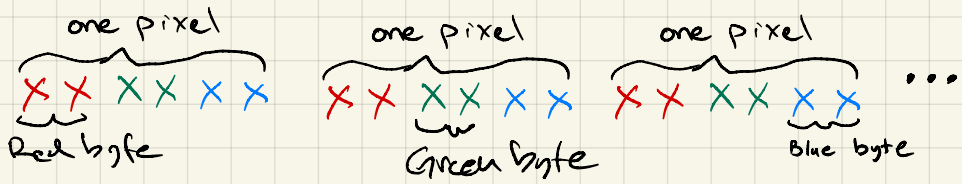
So, if you take 3 bytes and use the values to represent three levels of Red, Green, and Blue.

However if you have max values of these componets you get white. The least value, $\emptyset$ of each value is black.

Channels

So, this leads to how do you represent these values?

you could do something like this



Something where each pixel is three bytes. However if you put these in an array it would lose the connection between the array and the image.

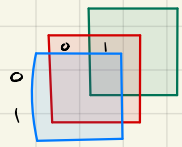
pixel i, j would be $bytes_{i,j}$, $bytes_{i+1,j+1}$, $bytes_{i+2,j+2}$

That would be problematic. Not to mention its now a 12-bit value and we lose the fundamental type.

So what to do?

Channels Again

Consider this, take our two by two image



Three arrays representing each of the color values.

This is what is coded. Later I'll show arrays which have shape (height, width, channel)

A grayscale Image will be (height, width, 1)

A color Image will be (height, width, 3)

I'm also using a convention here where instead of row, column, it's height, width.

Adjusting for more convention.

(channel, height, width) and even batch, or number of Images so an array of

- 4 images
- 2 height
- 2 width
- 3 colors/channels

would be coded as

(4, 3, 2, 2) (NCHW)

So how to rotate?

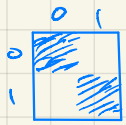
Simplify again to a 2x2 grayscale

So part of what this means is a little weird but it makes sense once you think about it.

All values in mxnet API are an array. A scalar value like 1, 2, 3, etc is encoded as an array of one element. So, instead of 1 its [1].

However for this next part keep it simple and Imagine a numpy array which is similar to mxnet's ndarray. with that said we can think in terms of an array since it is so representive of a the concept.

Remember the original image



Is encoded like this

$$\begin{array}{cc} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} \emptyset & 255 \\ 255 & \emptyset \end{bmatrix} \end{array}$$

Also notice the index starts at upper left. The values increase as you move right and down.

Rotate Again

There is a thing called a rotation matrix.

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Trust Me

Big leap. Look it up on wikipedia.

Lets write down a few values

The identity Matrix!

$$\begin{bmatrix} \boxed{1} & \emptyset \\ \emptyset & \boxed{1} \end{bmatrix} \leftarrow$$

$$\begin{bmatrix} \boxed{\emptyset} & -1 \\ 1 & \boxed{\emptyset} \end{bmatrix} \leftarrow$$

$$\begin{bmatrix} \boxed{-1} & \emptyset \\ \emptyset & \boxed{-1} \end{bmatrix} \leftarrow$$

$$\begin{bmatrix} \boxed{\emptyset} & 1 \\ -1 & \boxed{\emptyset} \end{bmatrix} \leftarrow$$

These are the same. the others are opposite.

θ	Circle	$\cos\theta$	$\sin\theta$	$\sin -\theta = -\sin\theta$
$\emptyset$		1	$\emptyset$	$\emptyset$
$\frac{\pi}{2}$		$\emptyset$	1	-1
π		-1	$\emptyset$	0
$\frac{3\pi}{4}$		$\emptyset$	-1	1
$-\frac{3\pi}{4}$			1	
$-\pi$			$\emptyset$	
$-\frac{\pi}{2}$			-1	
$\emptyset$			0	

or

mirror

So lets do some mults!

Our Image

Cols: 0, 1

Rows: 0, 1

$$\begin{bmatrix} \emptyset & 1 \\ 1 & \emptyset \end{bmatrix}$$

← The image

↓ our indexes

$\emptyset\emptyset$ $\emptyset 1$

$1\emptyset$ 11

The rotation matrices

Some rotations of 90°

$$\begin{bmatrix} 1 & \emptyset \\ \emptyset & 1 \end{bmatrix} \quad \emptyset$$

$$\begin{bmatrix} \emptyset & -1 \\ 1 & \emptyset \end{bmatrix} \quad 90^\circ$$

$$\begin{bmatrix} -1 & \emptyset \\ \emptyset & -1 \end{bmatrix} \quad 180^\circ$$

$$\begin{bmatrix} \emptyset & 1 \\ -1 & \emptyset \end{bmatrix} \quad 270^\circ$$

Take each pixel index and mult w/ Rot matrix for 90°

$1\times\boxed{2 \checkmark 2}\times 2$ (1x2)

$$\underbrace{\begin{bmatrix} \emptyset & \emptyset \end{bmatrix}}_{\text{Pixel value} = \emptyset} \underbrace{\begin{bmatrix} \emptyset & -1 \\ 1 & \emptyset \end{bmatrix}}_{\text{Rot}90} = \begin{bmatrix} \emptyset & \emptyset \end{bmatrix} \quad 1\times 2$$

$$\underbrace{\begin{bmatrix} 1 & \emptyset \end{bmatrix}}_{\text{Pixel value} = 1} \underbrace{\begin{bmatrix} \emptyset & -1 \\ 1 & \emptyset \end{bmatrix}}_{\text{Rot}90} = \begin{bmatrix} \emptyset & -1 \end{bmatrix}$$

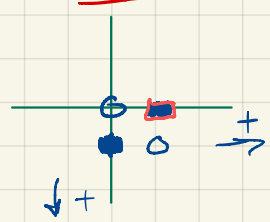
$$\underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}}_{\text{Pixel value} = \boxed{1}} \underbrace{\begin{bmatrix} \emptyset & -1 \\ 1 & \emptyset \end{bmatrix}}_{\text{Rot}90} = \begin{bmatrix} 1 & \emptyset \end{bmatrix}$$

marking to distinguish

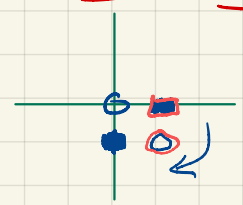
$$\underbrace{\begin{bmatrix} 1 & 1 \end{bmatrix}}_{\text{Pixel value} = \boxed{\emptyset}} \underbrace{\begin{bmatrix} \emptyset & -1 \\ 1 & \emptyset \end{bmatrix}}_{\text{Rot}90} = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

marking to distinguish

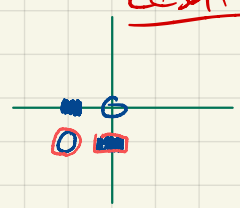
ORIG



ORIG + Rotate



Result



Hold up! The axis for the indexes are not right. And its a rotate by 90° not -90°!

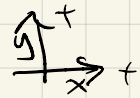
Indeed.

The indexes in the image array has $\emptyset,\emptyset$ in top left.
the array indexes are (Row, column) → (y,x)

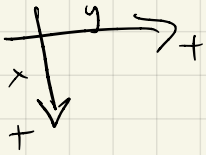
Your perceived rotate is in your coordinate system and not in the new coordinate system.

Take Away

indexes from source array contain a pixel value,

Multiply this value of index by Rot vector and the result is the destination index in "Rotated" array. The value of the pixel remains the same. The rotation angle in a normal 

coordinate system is reversed in a reversed coordinate system.



Lets continue. I don't believe you!

Fair enough, Let's.

Image

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

using symbols since we really multiply indicies.

Rotation

180/-180

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

A negative identity matrix since I have a lot of mults to write manually

Since we know previously the mults are of the indices and not the values. I could rewrite this as a matrix of $(n \times 2)$ size. I intend to multiply each row, column index by a 2×2 rotation matrix.

$$
\begin{array}{c|cccc}
\text{Row} \downarrow \;\; \text{Col} \rightarrow & 0 & 1 & 2 & 3 \\
\hline
0 & a & b & c & d \\
1 & e & f & g & h \\
2 & i & j & k & l \\
3 & m & n & o & p
\end{array}
$$

unroll / flatten / blast the matrix / array into an array list whatever

Pixel	Index Row	Index Column		Rotated Index Row	Rotated Index Column
a	∅	∅		∅	∅
e	1	∅		-1	∅
i	2	∅		-2	∅
m	3	∅		-3	∅
b	∅	1		∅	-1
f	1	1		-1	-1
j	2	1		-2	-1
n	3	1		-3	-1
c	∅	2		∅	-2
g	1	2		-1	-2
k	2	2		-2	-2
o	3	2		-3	-2
d	∅	3		∅	-3
h	1	3		-1	-3
l	2	3		-2	-3
p	3	3		-3	-3

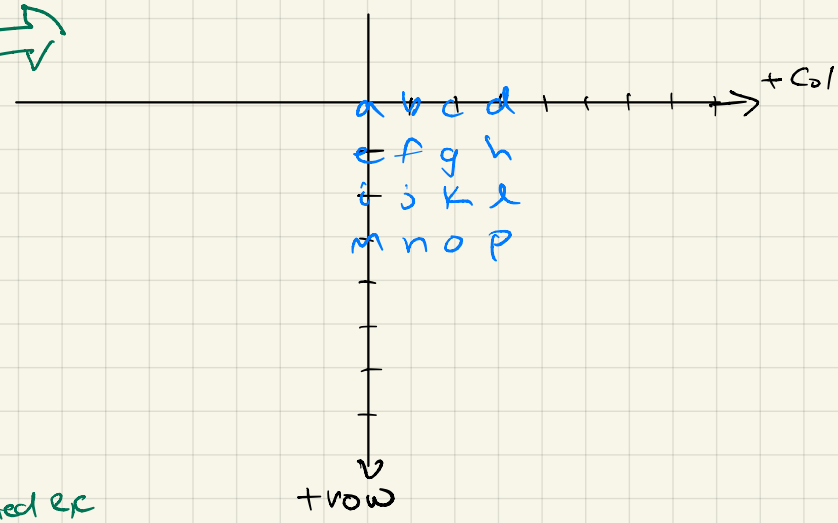
all mults → to make this colum are just neg since 180/-180 identity negative matrix

$$\begin{array}{c|c} & \text{Col} \rightarrow \\ \text{Row} \downarrow & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \end{array}$$

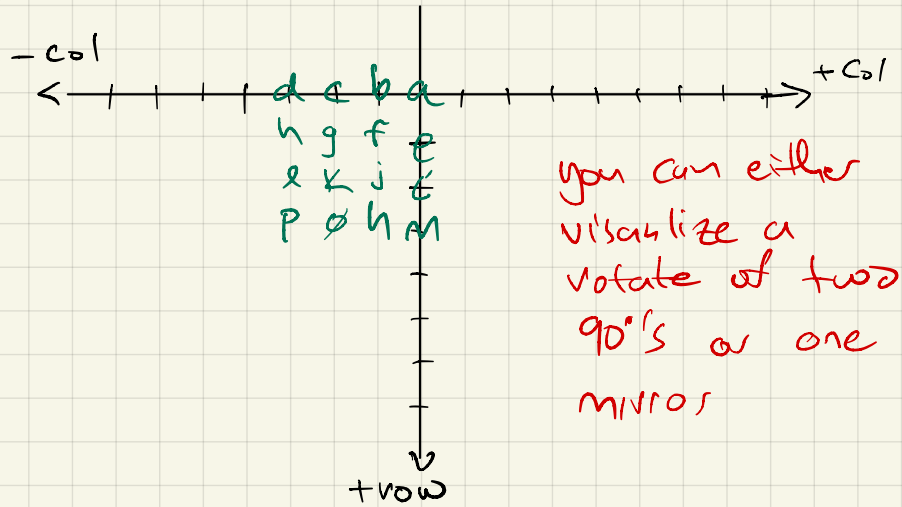
unroll / flatten / blast the matrix/array into an array list whatever

Pixel	Index Row	Index Column		Rotated Index Row	Rotated Index Column
a	0	0		0	0
e	1	0		-1	0
i	2	0		-2	0
m	3	0		-3	0
b	0	1	all mult → to make this column are just neg since 180/-180 identity negative matrix	0	-1
f	1	1		-1	-1
j	2	1		-2	-1
n	3	1		-3	-1
c	0	2		0	-2
g	1	2		-1	-2
k	2	2		-2	-2
o	3	2		-3	-2
d	0	3		0	-3
h	1	3		-1	-3
l	2	3		-2	-3
p	3	3		-3	-3

ORIG RC



Rotated RC

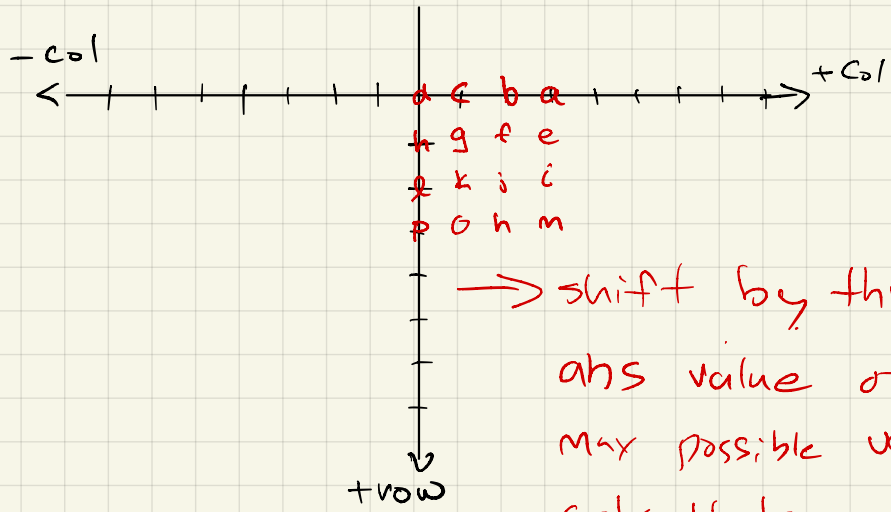


you can either visualize a rotate of two 90's or one mirror

Whoa! Hold on man, you are waving your hands a tremendous amount. You aren't even addressing the problem of an image with negative indices. How are you going to show a pixel at (-3,0) for symbol d?

Indeed but when I assign the new index assignment I can simply rewrite as

New Rotate Index = Rot mult Index result + 3



→ shift by three, then abs value of the max possible value to calculate.

How to determine the max value without scanning the results? Good question. How about do it, get an error, adjust by that amount when it says -x is not allowed.

Cut the Gordian Knot.

Ok, I kind of believe you. I just don't trust that you have tested all the possible results and really know what you are doing.

Fair enough.

Let's do a smaller matrix rotated with a non 90° rotate.

Let's use an image which looks like this.

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & a & b & c \\ 1 & d & e & f \\ 2 & g & h & i \end{array}$$

Rotate by 45°

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

these match

these are inverse

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \Leftarrow \text{the rotate matrix}$$

The matrix for image

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & a & b & c \\ 1 & d & e & f \\ 2 & g & h & i \end{array}$$

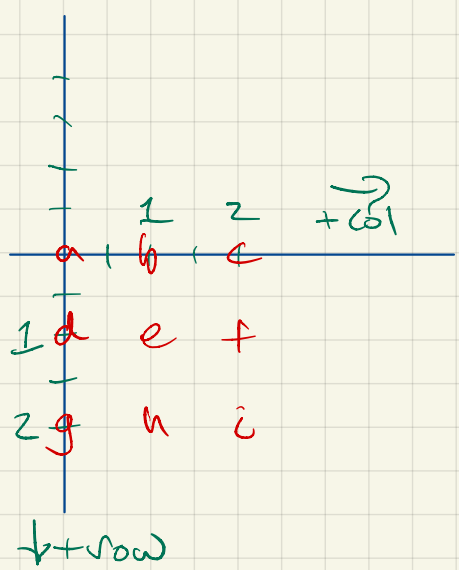
The rotation Matrix

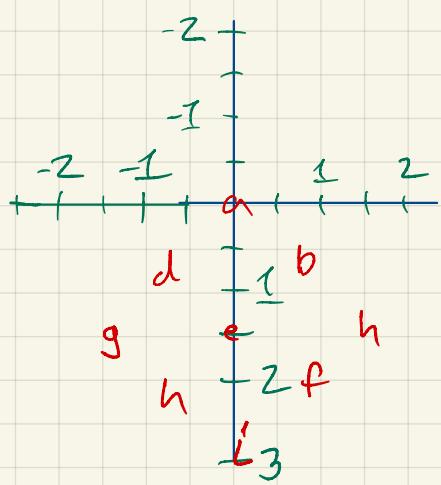
$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \Rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Rewrite to 2xn matrix of indicies.

a	[0,0]		0 0	a	0 0
d	[1,0]		A B	d	$\sqrt{2}/2$ $\sqrt{2}/2$
g	[2,0]		2A 2B	g	$\sqrt{2}$ $\sqrt{2}$
b	[0,1]	mult	C D	b	$\sqrt{2}/2$ $\frac{\sqrt{2}}{2}$
e	[1,1]	by	A+C B+D	e	$\sqrt{2}$ $\emptyset$
h	[2,1]	Rot	2A+C 2B+D	h	$\sqrt{2}+\frac{\sqrt{2}}{2}$ $-\sqrt{2}/2$
c	[0,2]	Matrix	2C 2D	c	$\sqrt{2}$ $\sqrt{2}$
f	[1,2]		A+2C B+2D	f	$\sqrt{2}+\sqrt{2}/2$ $-\sqrt{2}/2$
i	[2,2]		2A+2C 2B+2D	i	$2\sqrt{2}$ $\emptyset$

$$[x, y] \times \begin{bmatrix} A & B \\ C & D \end{bmatrix} = [xA + yC \quad xB + yD]$$





$\frac{\sqrt{2}}{2} \Rightarrow \sim 3/4$

$\sqrt{2} \Rightarrow \sim 1.5$

$\sqrt{2} + \frac{\sqrt{2}}{2} \Rightarrow \sim 2$

$2\sqrt{2} \Rightarrow \sim 3$

OK I trust you understand how to implement with MxNet?

A couple of things to note

- We won't be doing loops in python

  ie. for x=0 to n

  That will drop out of mxnet

- There is a stack overflow post where a guy asks how to do this and every single answer mentions np (numpy) routines. This violates above GOAL

- There is an iterator for ndarrays and even for images.

  Upon testing they use the batch parameter (N, C, H, W) to iterate on compete images

  ↑ REALLY just name axes