

Lecture 04:
3D Transformations including
Axis Angle Rotation

September 5, 2019

jupyter cs410lec03n03 (autosaved)

File Edit View Insert Cell Kernel

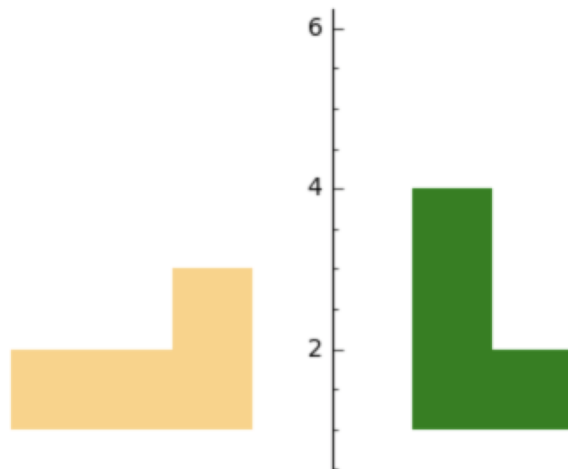
Before: Pts =

$$\begin{vmatrix} 1 & 3 & 3 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 4 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

After: Pts =

$$\begin{vmatrix} -1.0000 & -1.0000 & -2.0000 & -2.0000 & -4.0000 & -4.0000 \\ 1.0000 & 3.0000 & 3.0000 & 2.0000 & 2.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \end{vmatrix}$$

```
In [42]: ptsA = list(e1A[0:2,:].transpose())
ptsB = list(e1B[0:2,:].transpose())
gela = polygon(ptsA,color='green')
gelB = polygon(ptsB,color='orange',alpha=0.5)
bnd = 6.0
gos = gela + gelB
gos.show(xmin=-bnd, ymin=-bnd, xmax=bnd, ymax=bnd, aspect_ratio=1)
```



REVIEW: Use Notebook 3 from Lecture 3 to develop an intuition as well as mechanical understanding of 2D transformations!

Homogeneous Coordinates in 3D

- Same basic idea as for 2D.
- Now transformations are 4x4 matrices.
- 3D points represented as 4 element vectors.
- Let us consider the standard transformation
 - Translation
 - Scaling
 - Rotation

Translation and Scaling

- To transform a Point A into translated Point B.

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

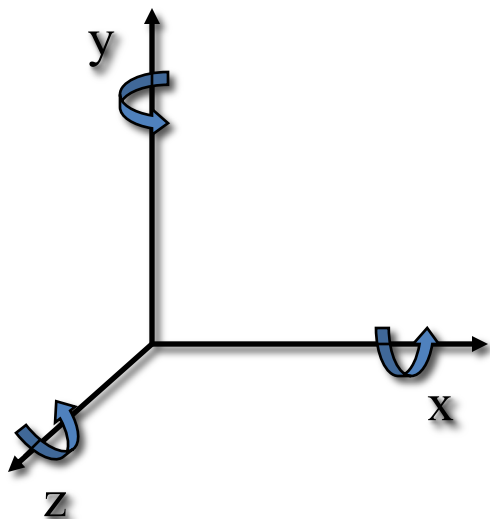
- To transform a Point A into scaled Point B.

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation – Part 1: Euler Angles

Rotations in 3D are more complex, because we now have three orthogonal axes that we can rotate around:

These are called the “Euler Angles”



Trust me:

Any rotation around any axis can be expressed as a sequence of three rotations about the x, y and z axes.

Rotation about Each Axis

- About the X axis

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- About the Y axis

$$R_y = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- About the Z axis

$$R_z = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Euler Angle Matrices - Pattern

- To rotate around any axis, start with the empty template:

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

- The axis you rotate around won't change:

$$\begin{matrix} \begin{bmatrix} & 0 & 0 \\ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & & & 0 \\ 0 & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ & 0 & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ (Z) & (X) & (Y) \end{matrix}$$

Remembering

$$\begin{bmatrix} \cos(x) & -\sin(x) & 0 & 0 \\ \sin(x) & \cos(x) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(Z)

Pay attention to the pattern.

(X)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(x) & -\sin(x) & 0 \\ 0 & \sin(x) & \cos(x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(Y)

$$\begin{bmatrix} \cos(x) & 0 & -\sin(x) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(x) & 0 & \cos(x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Each version replicates a 2D rotation matrix in a part of the 3D rotation matrix.

Composition of Rotations

- Rotate about x, then y, then z.

$$R = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos(\gamma) \cos(\beta) & -\sin(\gamma) \cos(\alpha) - \cos(\gamma) \sin(\beta) \sin(\alpha) & \sin(\gamma) \sin(\alpha) - \cos(\gamma) \sin(\beta) \cos(\alpha) & 0 \\ \sin(\gamma) \cos(\beta) & \cos(\gamma) \cos(\alpha) + \sin(\gamma) \sin(\beta) \sin(\alpha) & -\cos(\gamma) \sin(\alpha) + \sin(\gamma) \sin(\beta) \cos(\alpha) & 0 \\ -\sin(\beta) & \cos(\beta) \sin(\alpha) & \cos(\beta) \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Order matters! Rotate about z, then y, then x.

$$R = \begin{bmatrix} \cos(\gamma) \cos(\beta) & -\sin(\gamma) \cos(\beta) & -\sin(\beta) & 0 \\ \cos(\gamma) \sin(\beta) \sin(\alpha) - \sin(\gamma) \cos(\alpha) & -\sin(\gamma) \sin(\beta) \sin(\alpha) + \cos(\gamma) \cos(\alpha) & -\cos(\beta) \sin(\alpha) & 0 \\ -\cos(\gamma) \sin(\beta) \cos(\alpha) - \sin(\gamma) \sin(\alpha) & \sin(\gamma) \sin(\beta) \cos(\alpha) + \cos(\gamma) \sin(\alpha) & \cos(\beta) \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

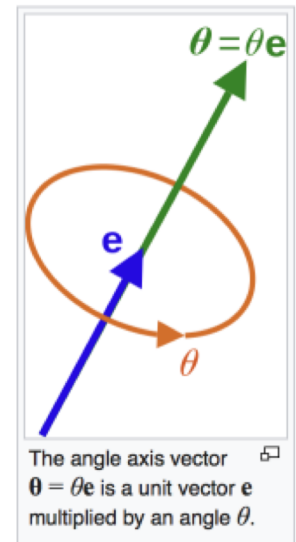
Please do not memorize these !

A Better Way – Axis Angle

To implement, best to review geometry of the cross product – next slide.

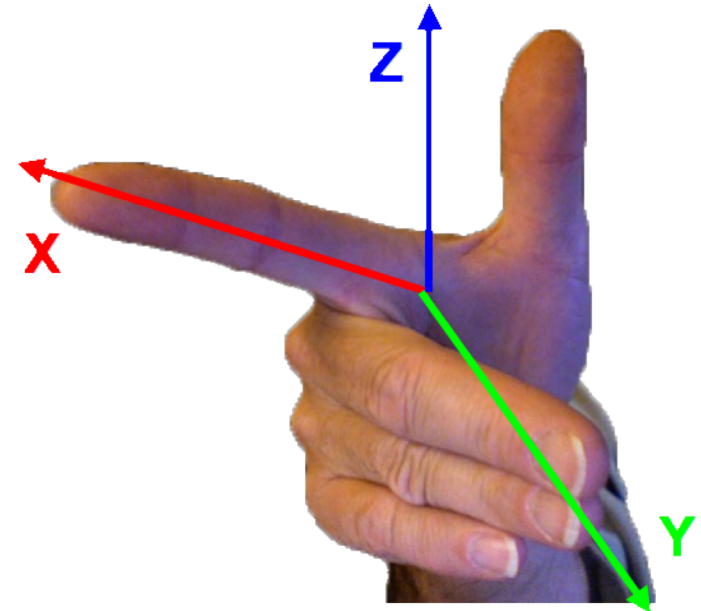
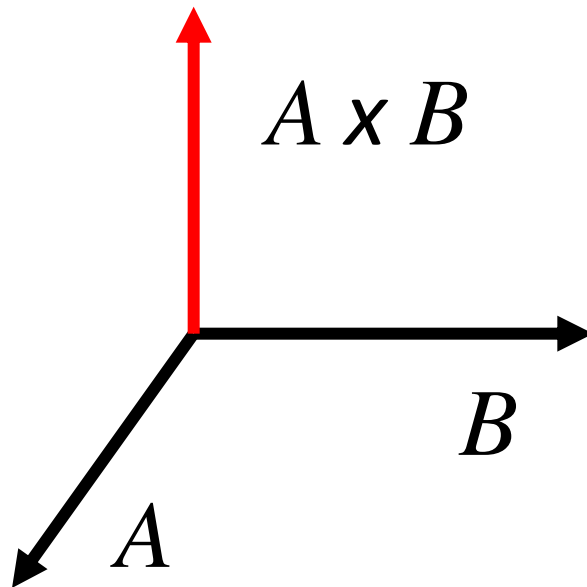


The screenshot shows a web browser window displaying the Wikipedia article titled "Axis-angle representation". The browser's address bar shows "en.wikipedia.org/wiki/A". The article's title is "Axis-angle representation" and it is categorized as an "Article". The main text of the article is visible, starting with "From Wikipedia, the free encyclopedia" and "Main article: Rotation group SO(3)". The text explains that in mathematics, the axis-angle representation of a rotation is defined by a unit vector \mathbf{e} and an angle θ . It states that the angle θ scalar multiplied by the unit vector \mathbf{e} is the axis-angle vector, represented as $\boldsymbol{\theta} = \theta \mathbf{e}$. The article also mentions that the rotation occurs in the sense prescribed by the right-hand rule and that the rotation axis is sometimes called the Euler axis. It concludes by stating that this representation is predicated on Euler's rotation theorem, which dictates that any rotation or sequence of rotations of a rigid body in a three-dimensional space is equivalent to a pure rotation about a single fixed axis.



Cross-Product

The cross product of two (3D) vectors is a new 3D vector that is perpendicular to both of the original vectors.



Right-hand Rule

More Formally

$$V_1 \times V_2 = u |V_1| |V_2| \sin \theta$$

Where u is a unit vector perpendicular to both V_1 and V_2 , as determined by the right-hand rule

If $V = (V_x, V_y, V_z)$ then:

$$V_1 \times V_2 = \left(V_{1y} V_{2z} - V_{1z} V_{2y}, V_{1z} V_{2x} - V_{1x} V_{2z}, V_{1x} V_{2y} - V_{1y} V_{2x} \right)$$

...don't despair!

It's easier to compute by hand than it looks

$$V_1 \times V_2 = \left(V_{1y}V_{2z} - V_{1z}V_{2y}, V_{1z}V_{2x} - V_{1x}V_{2z}, V_{1x}V_{2y} - V_{1y}V_{2x} \right)$$

$$V_1 \times V_2 = \det \begin{bmatrix} x & y & z \\ V_{1x} & V_{1y} & V_{1z} \\ V_{2x} & V_{2y} & V_{2z} \end{bmatrix} \leftarrow \text{Axes}$$

$$V_1 \times V_2 = \begin{bmatrix} x & y & z \\ V_{1x} & V_{1y} & V_{1z} \\ V_{2x} & V_{2y} & V_{2z} \end{bmatrix} \begin{bmatrix} x & y & z \\ V_{1x} & V_{1y} & V_{1z} \\ V_{2x} & V_{2y} & V_{2z} \end{bmatrix}$$

Blue products minus red products

Back to 3D Rotation

- In $P' = MP$, the points in P are projected onto the rows of M .
- In a rotation matrix:
 - The rows are unit length
 - Otherwise it scales the data...
 - The rows are orthogonal
 - Otherwise it shears the data...

To specify a rotation matrix, just specify the (orthogonal, unit) basis vectors of the new coordinate system!

Axis Angle Rotation

- In the special case that the axis is the Z axis, no problem:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where θ is the magnitude of the rotation

- But what about all the other possible axes?

Axis Angle (II)

- General rule: if there is a special-case coordinate system that makes life easy, *adopt that coordinate system!*
- In this case:
 1. Rotate data to make Z the axis of rotation
 2. Rotate about Z
 3. Apply the inverse of the original rotation

Axis Angle (III)

- How do we rotate the data to make the axis of rotation Z?
 - Multiplication is projection onto the rows of M
 - If M is orthonormal, it is a rotation matrix
 - Magnitude of every row is 1
 - Dot product of every pair of rows is 0
- If the third row is the axis of rotation, then
 - Z becomes the axis of rotation!

Axis Angle (IV)

- Step 1: normalize the axis of rotation
 - Write the normalized axis as $w = (w_x, w_y, w_z)$
- Step 2: pick any axis M not parallel to W
 - Heuristic: pick the smallest term in w , set it to 1 and renormalize to create m
- Step 3: create $U = W \times M$
- Step 4: pick an axis v perpendicular to w & u
 - $V = W \times U$ (or $U \times W$)

Axis Angle (V)

- Now put those together in a rotation matrix:

$$R_{\omega} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Axis Angle (VI)

- To rotate by θ around ω :

$$P' = \left(R_{\omega}^{-1} R_{Z_{\theta}} R_{\omega} \right) P$$

- R_{ω} is from the last slide
- $R_{Z_{\theta}}$ is the rotation matrix about Z, by amount θ
- What about R^{-1} ?

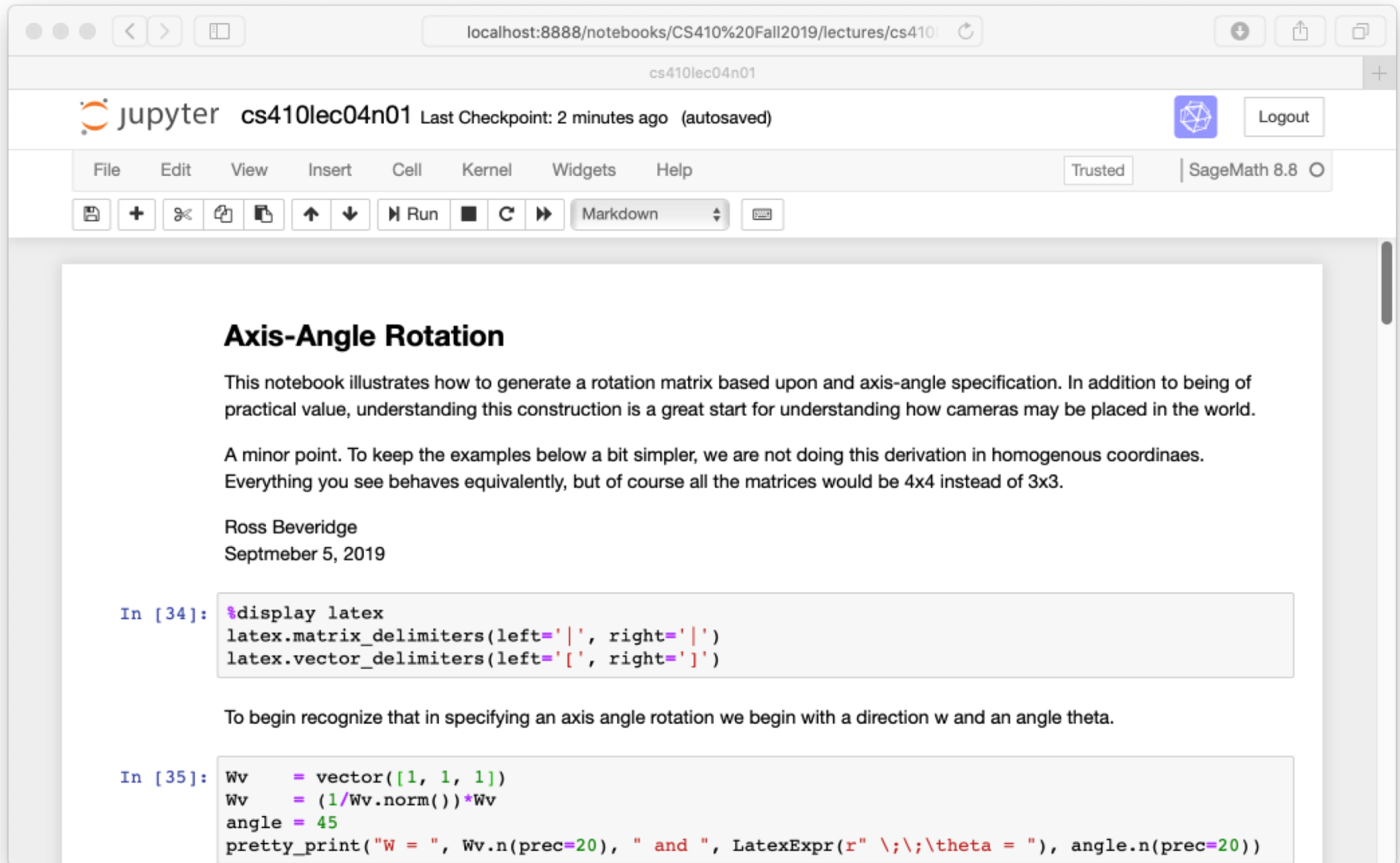
Axis Angle (V)

- Useful math fact: the inverse of an orthonormal matrix is its transpose

$$P' = \left(R_{\omega}^T R_{Z_{\theta}} R_{\omega} \right) P$$

- That's how you implement axis-angle rotation!

Practice with Axis-Angle



The screenshot shows a Jupyter Notebook interface in a browser window. The address bar shows the URL `localhost:8888/notebooks/CS410%20Fall2019/lectures/cs410/`. The notebook title is `cs410lec04n01`. The Jupyter logo and name are visible, along with the text `cs410lec04n01` and `Last Checkpoint: 2 minutes ago (autosaved)`. A `Logout` button is present. The menu bar includes `File`, `Edit`, `View`, `Insert`, `Cell`, `Kernel`, `Widgets`, and `Help`. The toolbar contains icons for file operations, a `Run` button, and a `Markdown` dropdown menu. The notebook content includes a title, introductory text, a code cell with LaTeX delimiters, and another code cell with vector calculations.

Axis-Angle Rotation

This notebook illustrates how to generate a rotation matrix based upon an axis-angle specification. In addition to being of practical value, understanding this construction is a great start for understanding how cameras may be placed in the world.

A minor point. To keep the examples below a bit simpler, we are not doing this derivation in homogeneous coordinates. Everything you see behaves equivalently, but of course all the matrices would be 4x4 instead of 3x3.

Ross Beveridge
September 5, 2019

```
In [34]: %display latex
         latex.matrix_delimiters(left='|', right='|')
         latex.vector_delimiters(left='[', right=']')
```

To begin recognize that in specifying an axis angle rotation we begin with a direction w and an angle θ .

```
In [35]: Wv = vector([1, 1, 1])
         Wv = (1/Wv.norm())*Wv
         angle = 45
         pretty_print("W = ", Wv.n(prec=20), " and ", LatexExpr(r" \;\;\;\theta = "), angle.n(prec=20))
```