به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

مبانی مکاترونیک

استاد: دکتر طالع ماسوله

مینی پروژه 1

فربد سیاه‌کلی

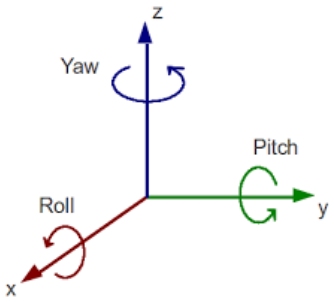
۸۱۰۱۹۸۵۱۰

فروردین ۱۴۰۱

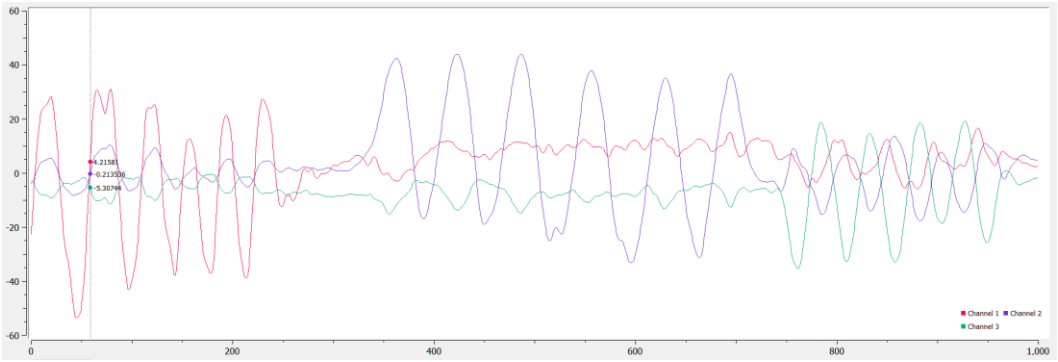
بخش اول:

نمایش داده‌های حسگر روی نمودار Serial Plot

حال با تست کردن بورد و جابه‌جایی آن در سوی سه محور x,y,z، و نگاه کردن خروجی در نرم‌افزار سریال پلات، می‌توان به جهت‌های سنسور پی برد. بعنوان مثال اگر پس از چرخش حول یک محور، مقدار roll افزایش یافت، در نتیجه آن محور x بوده است.



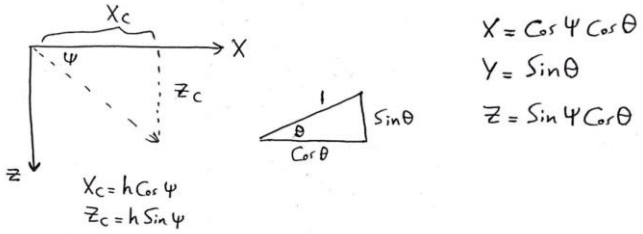
که در اینجا مثال گفته شده را به صورت عملی می‌بینیم:



در اینجا، برد در سه محور دچار دوارن شده که همانطور که می‌توان دید، محور اول چرخش ما محور x حسگر بوده چرا که کانال شماره یک که همان Roll است، دچار تغییرات شده. همچنین محور دوم باعث ایجاد تغییر در مقدار کانال دو شده که در نتیجه می‌توان فهمید این محور نیز محور y بوده است.

حال برای بدست آوردن محل جدید بردار x پس از چرخش‌ها خواهیم داشت که:

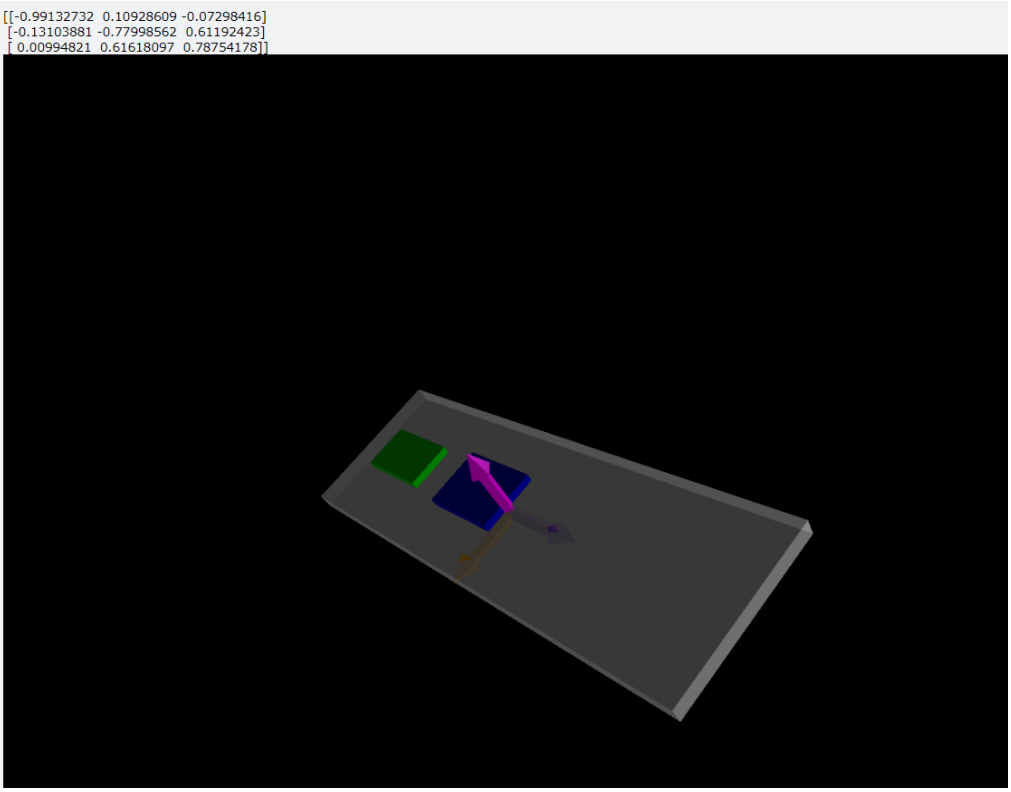
$$x_{new} = \cos(\psi) \times \cos(\theta)$$
$$y_{new} = \sin(\theta)$$
$$z_{new} = \sin(\psi) \times \cos(\theta)$$



و با استفاده از فرمول رودریگز vrot که همان z جسم خواهد بود بدست می‌آید:

$$\mathbf{v}_{\text{rot}} = \mathbf{v}\cos\theta + (\mathbf{k}\times\mathbf{v})\sin\theta + \mathbf{k}\,(\mathbf{k}\cdot\mathbf{v})(1-\cos\theta)\,.$$

نتیجه نهایی به صورت زیر شده است:



مشکل استفاده از این روش آن است که زاویه دوران بین زاویه صفر و 180 درجه تعریف می‌شوند، در نتیجه با گذر از این مقدار، جهت محور نیز تغییر می‌کند تا زاویه‌ای بین صفر تا 180 درجه را محاسبه کند. که این امر باعث تغییر ناگهانی جهت برد ما در شبیه سازی می‌شود. این مشکل در ویدیوی ضبط شده نیز به نمایش گذاشته شده است.

محاسبه ماتریس دوران نهایی به راحتی با ضرب همه ماتریس‌های دوران حول سه محور بدست می‌آید.

تابعی برای محاسبه این ماتریس نوشته شده.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_z(\psi)R_y(\theta)R_x(\phi)\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$
$$= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

کد چاپ ماتریس دوران با استفاده از roll, pitch yaw در Q1_1.py قرار گرفته است. همچنین در کد Q1_2.py شبیه سازی‌های جسم و آپدیت موقعیت آن انجام شده است.

برای بدست آوردن ماتریس دوران به کمک مختصات کواترنینی، می‌توان از معادله زیر بهره گرفت:

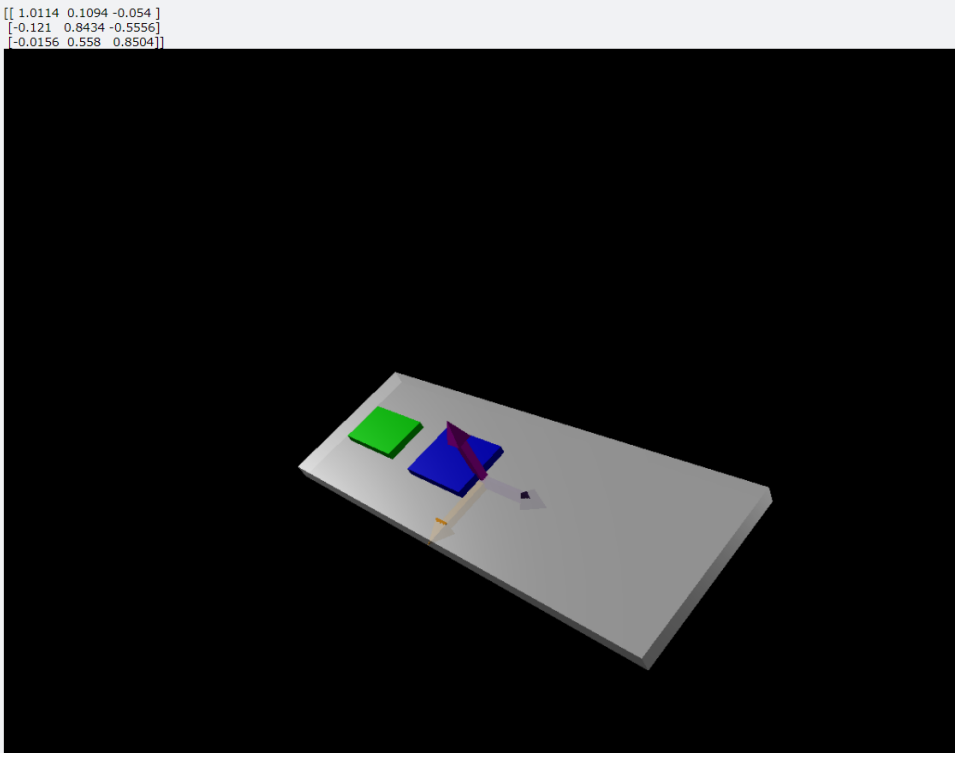
$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

حال می‌خواهیم که با مختصات کواترنینی اقدام به آپدیت جهت‌های جسم کنیم.

برای این کار ابتدا با استفاده از مختصات کواترنینی، roll، pitch و yaw را محاسبه کرده و سپس همانند قسمت قبل، بردارهای مورد نیاز را بدست می‌آوریم.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin(2(q_0q_2 - q_3q_1)) \\ \arctan\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix}$$

نتیجه به صورت زیر خواهد بود.



در کد Q1_3.py، با استفاده از مقادیر کواترنینی، ماتریس دوران محاسبه و چاپ شده و سپس در محیط vpython شبیه سازی آن انجام شده است.

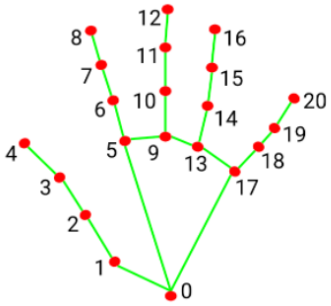
در این بخش به محدودیت‌های roll، pitch و yaw پی بردیم و دیدم که چطور با کواترنین‌ها می‌توان این محدودیت را برطرف کرد. حسگر استفاده شده از دقت نسبتا خوبی نیز برخوردار بود.

از کاربردهای این حسگر می‌توان به به دست آوردن موقعیت اشیا و سپس کنترل آن اشاره کرد که در صنایع زیادی کاربرد دارد.

بخش دوم

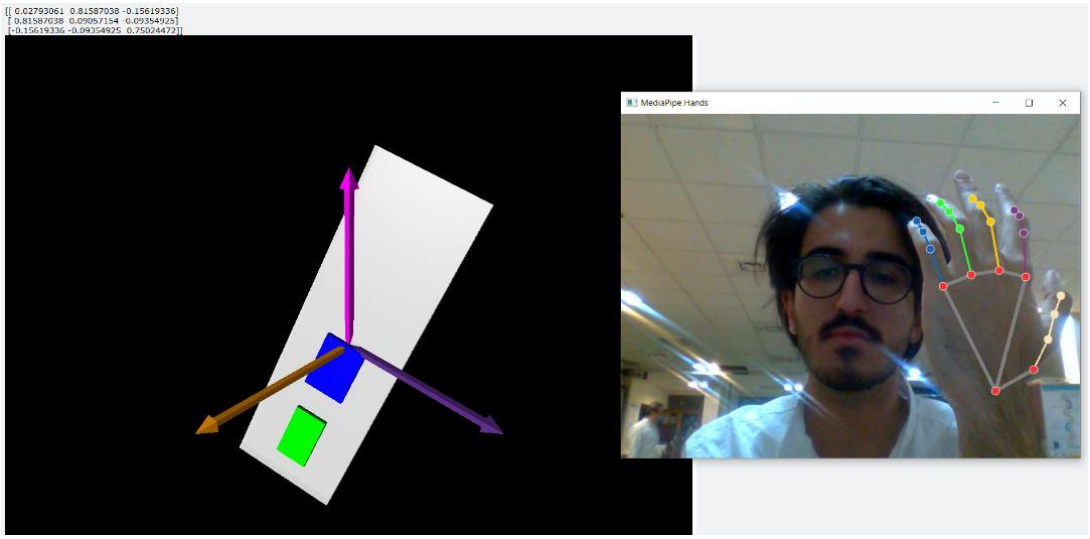
حال ابتدا با استفاده از شبکه عصبی تشخیص دست mediapipe، نقاطی بر روی دست خود را می‌یابیم.

نقاط detect شده توسط این مدل به صورت زیر شماره گذاری شده است:



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

حال با استفاده از نقاط صفر، پنج و هفده، سعی در بدست آوردن دو بردار x و z، به جهت کنترل کردن تخته می‌کنیم. بردار x را به گونه‌ای تعریف می‌کنیم که برابر با حاصل جمع دو بردار صفر به پنج و صفر به هفده باشد. در نتیجه توقع خواهیم داشت که با عمود نگه داشتن دستمان، تخته به صورت عمودی قرار گیرد. از طرفی بردار z را نیز به صورت ضرب خارجی دو بردار یاد شده می‌نویسیم و با استفاده از این دو بردار، تخته را کنترل می‌کنیم.



از آنجایی که تشخیص دست توسط شبکه عصبی دچار نویزهای بسیاری بوده و حتی در حالاتی که دست به صورت افقی قرار گرفته، نقاط به اشتباه تشخیص داده می‌شود، استفاده از حسگر شتاب سنج نتایج بهتری نسبت به این روش داشته و به طور کلی دارای اطمینان بیشتری است.

در حالتی استفاده از شبکه عصبی گزینه بهتری بود که از دو دوربین برای عمق سنجی استفاده شود.

(امتیازی)

برای محاسبه ماتریس دوران ابتدا مختصات اولین تشخیص دست توسط شبکه را بعنوان مختصات مبدا در نظر گرفته و سپس سعی در بدست آوردن ماتریس دوران مختصات جدید نسبت به مختصات اولیه داریم. در ویدیوی تست ضبط شده، ماتریس دوران به ازای هر دویست نمونه برداری محاسبه شده و به جهت صحت سنجی دترمینان آن نیز محاسبه شده است.

این بار یک مجموعه نقاط جدید شامل نقاط صفر و چهار و هشت را انتخاب کرده و با استفاده از جمع و ضرب خارجی بردارها، جسم را آپدیت می‌کنیم. این بار X و Z را به گونه‌ای تعریف کردیم تا با عمود بودن دست، جسم به صورت افقی قرار گیرد.

نتیجه به صورت زیر می‌شود.