**Computer Vision – Jonathan Hanbali**

In today's world, computer vision has become an integral part of our everyday life, revolutionizing tasks from object recognition to motion tracking with remarkable accuracy. Within this project, our aim is to harness the power of this revolutionary piece of technology and weave it into our overall workflow. The primary objective behind the computer vision segment of our Hoof IMU project centers on capturing drone footage of a horse in motion—walking or running—and subsequently analyzing the recorded video to **predict** potential issues. Our goal is to create a skeletal pose estimation and to superimpose the model onto the horse in the video, ultimately enabling us to calculate and compare the angles exhibited by a healthy horse versus one experiencing problems, especially laminitis. By utilizing this methodology, we seek to distinguish distinct differences between the skeletal structures and their respective angles, indicative of health or potential issues. Moreover, to deepen the integration within our project, we plan to correlate these computer vision predictions with data obtained from our inertial measurement device, enhancing our ability to forecast laminitis.

**METHODOLOGIES**

To achieve our computer vision goal, we explored various methodologies, encompassing established software/project, hardware solutions, and coding methodologies. However, our journey was rife with challenges. The path towards implementing these methods proved to be an ongoing process, demanding continual exploration to identify the most effective approach in achieving our objective.   
  
Thus far, we've experimented with three distinct methods in sequence:

1. Hardware Solution: NVIDIA Jetson Nano
2. Established Project: DeepLabCut
3. Coding: MATLAB Color Difference

Each of the methods mentioned above have its own sets of challenges and obstacles, which we will go over in each of the subsequent subsections below.

A close-up of a camera

Description automatically generated***Hardware Solution: NVIDIA Jetson Nano***

Figure 3 9- NVIDIA Jetson Nano Developer Kit (left) and 8MP CSI Camera - IMX219-130 (right)

Our initial strategy revolved around leveraging the capabilities of the NVIDIA Jetson Nano, an embedded computing board tailored for entry-level edge AI and computer vision applications. Jonathan personally purchased the NVDIA Jetson Nano development kit to assess whether Jetson Nano could serve as a pivotal tool to achieve our objectives. Our initial intention of utilizing Jetson Nano was to alleviate the computational load and streamline the process of our video capture and processing, particularly in the realm of AI integration. In our efforts to integrate Jetson Nano, we discovered that Jetson Nano comes with many problems, especially in software dependencies, as almost every Linux distribution has encountered. Linux often encounters software dependency issues due to its open-source nature and the diversity within its ecosystem. There are several factors that contribute to these issues, but mainly because it is possible to have more than one version of a software to coexist in Linux environments which could lead to a conflict, such that it was quite a challenge to ensure that we have the right version of the software required for a task. These challenges increased exponentially especially when we need to install more software, since now, the newer software that we need to install also have a dependencies to a certain version of software as “corequisite” or even a “prerequisite” in order for that software to run properly, such that, after installing multiple versions of software, we then stuck in the loop of trying to find a version of software that are needed in order for us to be able to do our task.

**Our milestone in utilizing Jetson Nano:**

* **JetPack Installation**: We successfully installed JetPack, the Ubuntu 22.04 Linux-based operating system for the Jetson platform.
* **Memory Swap Expansion:** We were able to increase the memory swap on the Jetson Nano to enable it to offload less critical data from RAM to our Micro SD Card on Jetson Nano, ensuring smoother operation and better memory management.
* **Software Setup:** We were successful in installing necessary dependencies and software essential for our machine learning task.
* **Camera Integration:** We were able to connect a CSI-based camera to analyze captured image utilizing Python Script running on Jupyter Notebook

**Challenges:**

* **Software Dependencies:** As mentioned above, we encounter a significant software dependencies issue which complicates our setup and slows down our progress.
* **Insufficient Beginner Resources:** We came in with a limited knowledge in Linux Operating System, Python programming language, and Machine Learning/Artificial Intelligence. With the lack of documentation and tutorials for beginners, it increases our initial learning curve and slows down our progress significantly.
* **Performance Issues with Image/Video Processing:** We experienced sluggish performance especially when processing the live video captured using our CSI camera setup.

Faced with the complexities and problems encountered during our hardware solution research, we made a resolute decision to pivot and explore alternative methodologies in our pursuit of achieving our computer vision objectives, however, if we would need to come up with a different solution that utilizes our hardware solution, Jetson Nano, I suppose that we would have to create a custom environment by using Docker containers or even Conda environment. Utilizing Docker containers would enable the development of a compact, self-sufficient software package containing all essential components for application operation, ensuring lightweight and portable execution, eliminating the issues of software dependencies.

A black background with a black square

Description automatically generated with medium confidence***Established Project: DeepLabCut***

Figure 10 - DeepLabCut Logo

After encountering many software dependencies problems when exploring our hardware options, Jetson Nano, we then pivoted and tried to see if other people already had done the same, estimating a skeletal body (pose estimation) on animals. One of the biggest hurdles that we can note is the fact that most of the machine learning models out there were focused on creating the pose estimation of the human body (even with the Microsoft Kinect, like what Matt tried during summer break), such that it was hard to find a training model for our purposes, since a training model is needed to train the computer’s neural network to be able to recognize different kinds of horses with different color and characteristics.

A screenshot of a computer

Description automatically generated

Figure 11 – Pose Estimation Search Results from Kaggle.com

We then tried to search far and wide, reading many papers about pose estimation, and found DeepLabCut which was developed by the Mathis Group and Mathis Lab at EPFL, Swiss Federal Institute of Technology Lausanne. Borrowing the word from their creator, [1]**DeepLabCut™ is an efficient [open source] method for 2D and 3D [marker-less] pose estimation based on transfer learning with deep neural networks that achieves excellent results (i.e. you can match human labeling accuracy) with minimal training data (typically 50-200 frames). T**here are several publications that utilized DeepLabCut, and many of them revolved around biomechanics of mainly animals, and pertaining to the utilization of DeepLabCut in predicting behavior. For more information about the publications involving DeepLabCut, see appendix \_\_**H**\_\_\_

As we mentioned before, that more often than not, Linux distributions often encounter software dependencies. DeepLabCut is no different. Since it was built on Python programming language, it also needed custom prerequisites software for the application to run smoothly, such that they utilized Conda environments, in which in itself is a directory that contains **specific collection** of Conda packages. Notice how they also would have to create a custom environment for their program to function. In our effort to utilize our established project solution, DeepLabCut, we were able to reach some milestones, while also facing some challenges, mostly hardware.

**Our milestone in utilizing DeepLabCut:**

* **Anaconda Installation**: We successfully installed on a windows laptop, Anaconda navigator, a distribution of the Python and R programming languages along with a collection of open-source packages used for scientific computing, data science, and machine learning tasks. This application contains Conda, the package and environment management system that were required for us to install DeepLabCut and provides a range of **pre-built** packages and libraries such as NumPy, Pandas, TensorFlow, scikit-learn and many others which are essential for machine learning purposes.

A screenshot of a computer

Description automatically generated

Figure 12 - Anaconda Navigator

* **DeepLabCut Installation:** We were able to install DeepLabCut successfully, and tried the software for a period until we encountered a different challenge; hardware.

A computer screen shot of a mouse

Description automatically generated

Figure 13 - DeepLabCut Graphical User Interface

**Challenges:**

* **Hardware Limitation:** As mentioned above, we encountered some hardware limitations. Jonathan’s laptop, where we installed DeepLabCut was an older computer, such that it was quite a demanding task to run the DeepLabCut on his computer.
* **Time Constraint:** We also have time constraints such that we don’t have time to fully understand and learn how to utilize DeepLabCut to achieve our objectives.

Despite the challenges that we encounter, we found some valuable resources that could be utilized in the future exploration of the computer vision section of the overall project, training datasets. Alexander Mathis and his group introduced Horse-10, a training dataset that [2] **comprises 30 diverse Thoroughbred horses, for which 22 body parts were labeled by an expert in 8,114 frames (animal pose estimation).** With this dataset, we would be able to train a machine learning model in the future if we choose to go back and utilize Jetson Nano.

**DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT**

***Coding: MATLAB Color Difference***

After the failed attempts of using Jetson Nano and DeepLabCut, Professor Zeke advised us to try to use MATLAB to achieve our goal, especially utilizing their Image Processing and Computer Vision applications, such that we then try this path. We are utilizing the Color Thresholder application to create a mask for our video. First, we would have to capture our video, to achieve this, we recorded the video of a horse using DJI drone with Professor Faramarzi. After doing several rounds of walking with different horses (some of them have a different color), we then grabbed the video from the SD card inside of the drone. Ultimately, we got around 10 usable videos of the horse movement. After that, we then these following methods:

* Taking a screenshot of the video to create a mask. We tried to create masks using several different screenshots from different tries, and we like the results that we get from Video #6. We then named the mask file as createMask6.
* Using the color thresholder toolbox to “select” the color that we want to “isolate”
* Using several MATLAB FUNCTION to modify the binary image
* Show the skeleton on the top of the original video

By utilizing the method above, we were able to achieve a rough pose estimation. However we noticed that some items were also being “skeletonized” since they contain the same color with the color masking that we choose, such that, we then need to utilize blob analysis to be able to “differentiate” or to basically eliminate the smaller blob of images, such that we will be able to only isolate the “biggest” blob of the binary image.

Figure xxx will show the video before the blob analysis

Figure xxy will show the video after the blob analysis

We then tried to see if our method actually works, we create a different mask (in this case, we took a blue color mask, and then apply it to our video, we we’re able to isolate the blue color from the jeans of Professor Farmanzi. We then let the video pass several frames, and then we noticed that after professor farmanzi goes outside of the frame, we got another “skeleton overlay” over some other things that have the same dark blue color. This indicates that our blob analysis is working perfectly. The idea behind blob analysis is to be able to select the bigges blob, and we were able to achieve this.

Figure xxx will show the video with blue color of blob analysis

Figure xxy will show the video with blue color of blob analysis after professor farmanzi going out of frame

The biggest problem from this method is to be able to determine whether the sekeleton is a part of the front leg or the back leg. Since the color difference method is only masking a certain color that we choose, such that sometimes they “attach” or connect the skeleton to the wrong part of the body. We also noticed that by using this method, we will not be able to differentiate clearly or draw the skeleton clearly, since we’re just using the default MATLAB function, such that there are still artifacts like the “branch” on the skeleton that does not even represent anything. In order to combat this issue, we then tried a method to “remove” shorter skeleton, but yet we found out that the center section of the horse skeleton consisted of many dashes or shorter lines, such that, those supposedly long lines is actually the shorter ones. With that, our effort to remove the shorter lines does not yield the result that we envisioned.

Looking forward to the improvement that we can do in the future, we are aiming to do a marker-ed pose estimation instead of a marker-less pose estimation. I think for our use case, it would be great to have a marker, such that we would be able to draw the skeletal overlay accurately. We are planning to integrate a cyan colored marker so we can differentiate it better (since typically, horse color will be closer to red-brown variation). After achieving that goal, we then can use the cosine law to calculate and estimate the angle between the thigh and the leg of the horse and compare it to the measurement that we gathered from our IMU device.

* + MATLAB Skeleton
    - Tangible Result
    - Faster to create
    - Hard to determine whether a skeleton is a part of front leg or back leg
* Progression
* Possibility of Enhancement
* What works what does not work

**DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT** **DRAFT**