Aditya Wagh

Website: http://adityamwagh.me

Email: adityamwagh@gmail.com
Mobile: +1-929-424-1931

Education

- **New York University** — **New York City, NY**
  Master of Science in Mechatronics and Robotics; GPA: 3.667/4 — Sep 2021 – May 2023
- **Birla Institute of Technology and Science, Pilani** — **Pilani, India**
  Bachelor of Engineering in Electronics and Instrumentation — Aug 2015 – May 2019

Experience

- **Central Electronics Engineering Research Institute** — **Pilani, India**
  Deep Learning Intern — Jul 2018 - Dec 2018
  - **Data Annotation**: Contributed to the pixel wise annotation of a novel data set consisting of 6000+ Infrared and RGB aerial images of power cables.
  - **Mask RCNN**: Fine-tuned a pretrained Mask RCNN model for instance segmentation of power cables on the new dataset and achieved an accuracy of approximately 70%

Technical Skills

- **Languages**: Python, C++, Bash, MATLAB, LaTeX
- **Frameworks**: PyTorch, Keras, TensorFlow, OpenCV, Open3D
- **Tools & Platforms**: VSCode, Vim, Git, GitHub, NYU HPC
- **Operating Systems**: Linux, MacOS, Windows

Relevant Coursework

- **Foundations of Robotics (F21)**: Kinematics, Dynamics, Open & closed loop control of a 7 revolute joint manipulator
- **Robot Perception (F21)**: Projective Geometry, Camera Calibration, SFM, SLAM, Bundle Adjustment, Optical Flow, Tracking
- **Robot Localisation and Navigation (S22)**: Bayes & Kalman Filter, Pose Estimation, Motion Field, EKF & Pose Graph SLAM
- **High Performance Machine Learning (S22)**: Performance Optimization in PyTorch, Distributed DL, CUDA
- **Deep Learning (S22)**: CNNs, RNNs, Transformers, GANs, BERT, Recommender Systems, Self-Supervised Learning, Deep RL
- **MOOCs**: Mathematics for ML, Introduction to Tensorflow Neural Networks & Deep Learning, CNNs in Tensorflow,

Projects

- **Fully Convolutional Networks for Post-Earthquake Damage Assessment**: (tensorflow)
  - **Introduction**: Developed FCNs for semantic segmentation of components and damage state of a damaged building.
  - **Model**: Designed symmetric convolutional encoder and decoder networks to pixel wise classification of building components and detect damaged components.
  - **Outcome**: Achieved a mean IoU of 83% over 5 component classes and mean IoU of 70% for 5 damage state classes.
- **Dimensionality Reduction using Convolutional Autoencoders**: (pytorch, scikit-learn)
  - **Introduction**: Developed a deep convolutional autoencoder to reduce the dimensions of 28x28 size Fashion MNIST images.
  - **Model**: Implemented an 10 layer convolutional autoencoder with a latent space of 64 dimensions.
  - **Visualisation**: Further reduced dimensions of the latent space to 2 dimensions using t-SNE, and visualised the result.
- **Marker based Augmented Reality**: (opencv, pyapriltag)
  - **Introduction**: Drew a AR 3D cube on the image of an AprilTag marker.
  - **Marker Detection**: Detected an AprilTag marker by computing corresponding corners and centers of the marker.
  - **Perspective Transformation**: Solved a PnP transform between the corners of a marker and a face of a cube.
  - **Cube Construction**: Computed and projected 8 points of the cube on the image and drew lines using OpenCV
- **Pose Estimation between two images**: (numpy, opencv, pyapriltag)
  - **Introduction**: Computed the relative pose between two images captured from different viewpoints.
  - **Camera Calibration**: Calibrated a camera using April Tag based calibration rig and removed distortion from the images.
  - **Fundamental Matrix Estimation**: Computed the fundamental matrix using the normalized 8 point algorithm.
  - **Pose Estimation**: Estimated the relative pose between two images by decomposing essential matrix.
- **Bag of Visual Words for finding similar images**: (opencv, scikit-learn)
  - **Introduction**: Developed a method to find similar query images from a database of 29,000 50x50 resolution images.
  - **Feature Extraction**: Computed SIFT features for each image in database using OpenCV's built-in SIFT feature extractor.
  - **Clustering**: Used the k-means clustering algorithm to compute 800 cluster centroids which serve as visual words.
  - **Histograms**: Computed histograms of computed visual words for all the query images and images in the database.
  - **Image Query**: Used the k-nearest neighbours algorithm to find the histogram that is most similar to query image.
- **3D Plane fitting in Point Cloud Data**: (open3d, numpy)
  - **Introduction**: Implemented the RANSAC algorithm from scratch on 3D point cloud data in pcd format.
  - **Point Selection**: Randomly selected 3 points in data and computed plane parameters using parametric equation of a plane
  - **Plane Parameters**: Computed the best plane parameters by minimizing perpendicular distance of each point in data from the plane.