AMAN SEHGAL

MACHINE LEARNING / COMPUTER VISION / PERCEPTION / ROBOTICS / EMBEDDED SYSTEMS / MECHATRONICS

aman.1.sehgal@gmail.com 215-834-9732 Philadelphia, PA-19104
https://www.linkedin.com/in/amansegal

SYNOPSIS

Firmware Development and Design • C/C++11 • Python • MATLAB/SIMULINK to design filters and control systems and prototyping algorithms • Machine Learning with Python in SciKit-Learn, SciKit-Image, OpenCV, TensorFlow, NumPy/SciPy • Computer vision with MATLAB, PIL, OpenCV • Image Processing • DSP • OpenGL / GLSL shaders • multi-threading and concurrent programming • RTOS

VISION / PERCEPTION / GRAPHICS

- Edge detection pipeline: (x,y) Gradients → energy → non-max suppression → edge linking for the edge map.
- Image morphing pipeline using 1)Delaunay triangulation+linear interpolation and 2) thin plate spline interpolation.
- Image stitching pipeline: shi-tomasi corners feature → adaptive non-max suppression → feature patch descriptors → descriptor matching → outlier rejection using RANSAC to pick best matches → estimate homography → Transform image into final frame.
- Image down sizing using seam carving → Energy matrix → Dynamic programming to carve low cost seam.
- Image processing using Gabor functions, DCT, FFT, CNNs, gaussian and laplacian kernels.
- Image Segmentation using GMMs and K-means.
- Homography estimation to implant perspective correct images.
- Edge and Blob detection using SIFT based pyramid at different scales.
- Structure from 2 view geometry using two images with matched features.
- Pose estimation for a quad rotor using an estimated homography from images of April tags taken from a camera mounted on the quadrotor.
- Implemented visual inertial odometry by fusing data from a camera and an IMU. Velocity and orientation from camera were used along with the same from an IMU and fused using an EKF to estimate the robot state vector.
- Made a manipulable box humanoid using OpenGL and GLSL. Implemented the geometric transforms for the humanoid as a rooted tree with some nodes that had drawable geometries. A flat shader was used to color the humanoid.
- Wrote a graphics renderer implemented as rasterization. A 3D model view was projected into pixel space using a camera model with view and projection matrices. Vertex and fragment properties (color, z-coord, surface normals, etc.) were determined by barycentric interpolation and Z buffering was used to determine which fragment to display.
- Used OpenGL to create surface shaders (Blinn-Phong, lambertian, flat, gradient, madcap and a custom shader that deformed the object and its colors with time.)
- Used OpenGL to create post-process shaders: Sobel with convolution, gaussian with convolution, greyscale with a vignette effect, binary shader with luminance thresholding, worley noise shader, brownian noise shader and a shader that mimics a poor quality VHS tape.

- Wrote a GUI application to load and parse a OBJ file with a triangulated closed mesh representation of a 3D model. The data was stored in a half-edge mesh data structure. Additional features were added to highlight the face, vertex and edges on the final rendered image using Qt's observer pattern (signal-slot mechanism).
- Implemented catmull-clark subdivision to recursively interpolate B-spline surfaces on a given topological mesh. Rendering was done using OpenGL. Selected faces could be triangulated upon selection from the GUI.
- Implemented shader based deformation using skeleton skinning. The transformation is applied in the a shader program to deform the skeleton of the mesh and then rendered using the openGL pipeline.
- As part of a game development project, implemented a rudimentary physics engine, used L-Systems to generate trees, caves and rivers on a terrain that was generated based on fractal brownian motion. The terrain and assets were generated in real time using multiple threads.

MACHINE LEARNING

- Generated learning curves for decision trees to evaluate performance of trees with different levels. Also created a decision boundary plot for decision trees to visualize shape of the said boundary.
- Implemented back propagation to train a neural net that was used for image classification on the MNIST data set. Plotted the the weight matrices of the layers as a grid of pixels to visualize the learned features.
- Implemented univariate and multivariate linear regression using Gradient descent and compared it to the closed form solution. Also implemented polynomial regression and generated a learning curve with different regularization polynomial degree parameter values.
- Implemented naive Bayes as batch learning and online learning for text classification of documents from multiple categories.
- Implemented Logistic weighted linear regression for the generating EMG signals corresponding to finger movement from the EEG data. Pre processed the data using Low-pass filtering and post processed the data using adaptive least-mean square filter and Savitsky-Golay filtering to achieve high correlation between labels and predictions.
- Used LASSO and polynomial regression to regress on EEG data to generate the future samples of finger movement neuronal activation.
- Used HoG features extracted from images to train an SVM to classify faces in images.

MECHATRONICS / EMBEDDED SYSTEMS

- PD position control of 2 phase DC motor with optical quadrature encoder as feedback.
- Implemented a PID controller to track a helical trajectory with minimal error for a quad rotor where the position was generated using a cubic polynomial.
- Programmed forward kinematics of of a RRR simulated kinematic chain manipulator using DH convention.
- Solved inverse kinematics of the Puma robot and programmed the final equations in MATLAB to control the end effector position in cartesian space by turning the joint motors through the appropriate angles.
- Designed and Developed a sensor PCBA using the ATSAMW25 as the compute platform for local climate sensing and reporting.
    - Schematic layout and PCBA design was done using Altium. BOM was maintained on Octopart and part selection was done using DigiKey and Mouser.
    - Implemented a single pole LPF to smooth temperature and RH readings.
    - Other functionalities include boot loading, remote actuation using commands over MQTT, and CLI over USART.
    - The board also supported OTAFU over HTTP which was secured using symmetric key cryptography.
    - CRC32 was used to verify Firmware packets.
- Designed a PID Linear regulator for tracking a toy train coach automatically based on heading-distance and feedback of velocity from hall effect sensors attached to the wheel. The code was implemented in C on an Arduino Uno board. the speed was controlled using open loop PWM control of a 2-phase servo.

- Performed system identification of a ball mounter on a thin metal strip in order to facilitate open loop control using a notch filter with the notch at the system's natural frequency. Input command was given using a joystick interface potentiometer that was sampled from an ADC channel. This prevented oscillations of the system in response to a step function.
- Created a model of the Human Heart in SIMULINK/STATEFLOW and simulated various conditions such as bradycardia and tachycardia. Did the same using a timed automaton model and verified using CTL queries in UPPAAL.

EMPLOYMENT

Penn Center for Neuroengineering and Therapeutics
Research Staff - *May 2017 to August 2017, Philadelphia, PA*

Apple Inc.
Firmware Engineering Intern - *May 2016 to August 2016, Cupertino, CA*

FARO Technologies
Firmware Engineer - *Oct 2014 to Apr 2015 - Exton, PA*

Lutron Electronics Co. Inc.
Senior Project Embedded Engineer - *Feb 2012 to Oct 2014 - Coopersburg, PA*

Comcast Cable
Systems Test Engineer - *Jun 2011 to Feb 2012 - Downingtown, PA*

CoVal Systems
Embedded Software Engineering Intern - *May 2010 to Aug 2010 - Ardmore, PA*

EDUCATION

University of Pennsylvania
M.Sc. Engineering - Robotics, 2018

University of Pennsylvania
M.Sc. Engineering - Embedded Systems, 2017

Villanova University
B.Sc. - Electrical & Computer Engineering, *Magna Cum Laude*, 2011

SKILLS

MACHINE LEARNING:
TENSORFLOW, SCIKIT-LEARN, SCIKIT-IMAGE
NUMPY/SCIPY, WEKA, OpenCV

PROGRAMMING LANGUAGES:
C/C++ (including STL), Java, Python,
MATLAB, PHP, JavaScript, VHDL

IDES:
Eclipse, NetBeans, IAR Workbench, Code Warrior,
Visual DSP++, Code Composer, Xilinx SDK, Keil
uVision, ARM DS, Eclipse PyDev, VSCode

BUILD TOOLS:
CMake, Make, ANT

SOURCE CONTROL:
GIT, MKS Integrity

AGILE TOOLS:
Version One, Atlassian JIRA, HPQC

CONTINUOUS INTEGRATION TOOLS:
Jenkins, Atlassian Bamboo, GitLab CI

TDD UNIT TESTING:
JUnit, CppUtest

PCB DESIGN:
Altium Designer, P-CAD, Eagle CAD

EMBEDDED PLATFORMS:
ST ARM M0/M3, NXP ARM M3, TI ARM M4,
Xilinx ZYNQ, STM 8, ADI SHARC,
Freescale iMX6, Motorola ColdFire

RTOS:
freeRTOS, Keil RTX-RL, uC/OS-II, pSOS+, Embedded Linux

DEBUGGING:
Oscilloscopes, DMMs, Function Generators,
Standard JTAG ICEs/ICDs, Logic Analyzers

FORMAL VERIFICATION:
UPPAAL, PC-LINT, UML, PYLINT
MATLAB STATEFLOW