Autonomous Vehicle Competition Project Report

Bailin Lu[1] Chusheng Ku[2] Ryan Cheng[3] Xiaolan Cai[4]

***Abstract*— This report will introduce the autonomous vehicle we designed and built that can operate itself in an unknown closed course. The techniques we use in building this car learned from class CSCI 5302 Advanced Robotics and open source online. To complete the task of the class, we need to make the car do some challenge. Challenge we select are list as following:**

- **Visual-inertial SLAM**
- **Sparse Map**
- **Stop at a stop sign**
- **Report on human-robot interaction with autonomous vehicles.**
- **Report on deep learning and its uses to improve robotic systems**

In this report, we will present the platform, system and algorithm we use and why we use them. Introduction

I. INTRODUCTION

The development of 'self-driving' or 'autonomous' car technologies have advanced to very promising levels in recent years and could become commercially available within a few years[1]. More and more company are working on this project including Tesla, BMW, Honda, Ford and so on. Self-driving car technologies could increase on-road safety, make transportation more efficient. Our job is to build a robot can drive itself in an unknown environment. This project helps us learn the algorithm and principle of a self-driving car.

The sensor of our robot are oCam camera,IMU and two IR sensors. Although the sensors are much cheaper and simpler than the sensor use in self-driving cars. We still can learn the control method and mathematical model by working on the tasks. Our whole system works on the Robot Operating System (ROS). ROS is a flexible open source framework for writing robot software. It provides lots of tools, libraries, and conventions that aim to simplify the task of creating robot behavior across a wide variety of robotic platforms. In ROS, data is broadcast over topics, while nodes are executable scripts that subscribes to their needed topic to receive the data. In our system each sensor has a node that transform the data into meaningful information and then publishes the data. The car motor and steering servo are sent input via a node that published the needed phase width modulation signal from the pololu servo controller.

*This work was not supported by any organization

[1]Bailin Lu, Mechanical Engineering, University of Colorado Boulder, balu9739@colorado.edu

[2]Chusheng Ku, Computer Science, University of Colorado Boulder, Chusheng.Ku@colorado.edu

[1]Ryan Cheng, , University of Colorado Boulder, rych6608@colorado.edu

[1]Xiaolan Cai, Computer Science, University of Colorado Boulder, xiaolan.cai@colorado.edu

As to the three challenge we finished. The first challenge is visual-inertial SLAM. SLAM (Simultaneous Localization And Mapping) has been a very active and almost ubiquitous problem in the field of mobile and autonomous robotics for over two decades[2].

It is the computational problem of updating a map of an unknown environment while simultaneously keeping track of features or keyframes location within it. Robot can locate itself by going in unknown environment look for features, such as corners, edges and contrast in color. The main algorithm is extended Kalman filter which is the nonlinear version of the Kalman filter. SLAM will always use several different types of sensors, and the powers and limits of various sensor types have been a major driver of new algorithms[3]. We combine 1042 PhidgetSpatial IMU with our oCam as our sensor. In the following section we will discuss visual-inertial SLAM in detail. Second challenge is stop at a stop sign. We use deep learning package train the camera then test it with printed stop on A4 paper. When we get high enough accuracy, we install the whole system in ROS and run with robot so that the robot can recognize the stop sign and stop immediately. Third challenge is power slide around turns. The basic idea is that by adjusting the PD parameter of the control system and speed of the car to make the robot drift. Last challenge is report about human-robot interaction with autonomous vehicles. You can read it in appendix.

II. VISUAL-INERTIAL SLAM

A. The Setting

As shown in Figure 1, we attached IMU on oCam to record the rosbag file. We finished taking the video by walking at normal speed around the racing route for two loops.

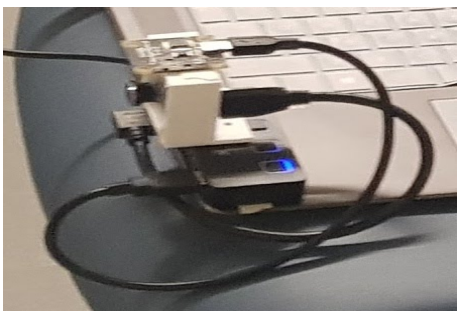

Fig. 1. The setting of camera and IMU

B. VINS-Mono

We used VINS-Mono, a robust monocular visual-inertial system, to finish VI-SLAM and create the sparse map. There

were three options for the configurations.

1) 0: Have an accurate extrinsic parameters
2) 1: Have an initial guess about extrinsic parameters
3) 2: Don't know anything about extrinsic parameters

We used option 2 in VINS-Mono, which makes the system calibrate automatically. To calibrate by itself, we need to rotate the camera and IMU in the beginning. It took around 20 seconds to finish calibration. After initializing successfully, we could start to move around the route. For the camera calibration parameters, we used ROS camera calibration package to get the values. After configurations are set correctly, VINS-Mono performs well on the video recorded along the racing route.

Trying to taking the videos many times, we met some issues and learned how to get a good result. First, we tried to record the rosbag file while running real-time VINS-Mono simultaneously. However, the system became unstable that it ran slowly and stopped working after a while. Then we figured out the real-time analysis consumed a lot of computing power. Therefore, we could only record the video and run it with VINS-Mono to check to result separately.

Second, based on the experience with ORB SLAM2, we walked slowly to make sure VINS-Mono would work well on the video. After several trials, we realized that slow motion did not make sense since IMU could get more useful inertial information from our move. If we turned faster at the corner, the inertial change helped VINS-Mono to estimate the poses.

Last, to get a loop closure, we walked around the route for multiple loops. Each time we moved back to the same place, the feature extractions matched the history points, we would get more accurate pose estimations, but the current points drifted away from history points, which made the sparse map useless as show in Figure 2

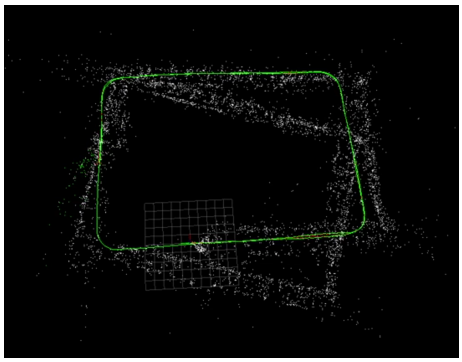

Fig. 2. The estimation by VINS-Mono

C. ORB SLAM2

ORB SLAM2 was another real-time SLAM library we used to finish the challenge. We could run it in real-time along the racing route, but it lost the track at the corners easily even we moved slowly. Without inertial information, we could not get the sparse map correctly with visual-only SLAM.

III. Control

The final control configuration utilized two PID controllers running in parallel with the state machine to dynamically switch between the control effort for steering car in either a corner state or straight state. One PID contoller was tuned with a gains setting that would allow for the vehicle to operate smoothly while going straight in the middle of the hallway. The second was a more aggressive gain setting to allow for rapid cornering when a corner was detected. That is, the proportional gain was set to a large value to make the rise time as quick as possible. Some differential gain was added so that the vehicle can mildly stabilize while cornering. The result was a vehicle that can come into a corner and take a racing line around a turn. Leading to our vehicle having a respectable time close to the leaders while going at a notably slower wheel speed.

The placement of the IR sensors made implementation of the PID controllers very simple. We placed the sensors at 45 degree angles pointing to each wall. This allowed us to make the set point to be the center of the hallway with the state being the distance from center. With the use of a PID loop that was made for the straight sections we were able to get a system that worked fairly well in the straight and corners. The placement of the IR sensors also gave the advantage for turning. By subtracting the right of the left hand distance the car would always have a preference to turn right when a corner was detected.

IV. State Machine

The final state machine configuration was a simplistic if/else script that had three states. At the start of tract, the state machine had a period of time of going straight to overcome the doorways at the beginning. The second state was looking for a large distance on the the right hand side as its conditional to publish the cornering control effort. It assumed at all other times the car would be going straight down a hallway, thus be publishing the straight-ways control effort. This was effective for patches of the track that did not contain a lot of doorways. However this simplistic state machine had short comings in the patches of track that had doorways. This was due to the metal foot trim causing the IR sensors to read a sporadic distance that varied from 3 meters and beyond. This led to the issue of the state machine switching to a cornering state part way when passing doors. This limited the aggressiveness of our control system since we needed to make a set of gains that would get the vehicle around the corner but also not so aggressive that car would dive into to the doorway. It was attempted over come these erroneous readings by adding complexity to the state machine by bringing in more data. That is the state machine script using the right and left distances as references for their conditionals. However, this proved to be more difficult than thought and resulted in a system that was not effective at all. The thought of using a series of "doorway counters", timing periods and reference features came when we thought of using a timer for the first state of the track however doing more than this was rejected due to the team's desire to have a more

generalized system. The team did not want to navigate the course by "dead reckoning" the track to get the best time. Opting to go for a vehicle that can handle a less strictly and specifically structured state machine. Then the idea of filtering the incoming data to prevent erroneous signals was attempted with if/else statements however this proved futile due to how the IR sensors behaved when detecting metal surfaces. We could not find a set of filtering conditionals to get past the doors without completely impairing the capabilities of the car to detect corners. In the end we fell back to our simplistic state machine with some added data filtering to remove extremely large distances. We spent many hours dialing in our gains settings to get the best performance with keeping repeat ability and robustness as a priority.

As a note, we first attempted to use ROS's SMACH package as a framework for the state machine so that complication can be easily added later. However it was implemented in a way that made it unreliable and would stop function shortly after launch. It was later learned, that issue was caused by the code creating multiple subscriber objects causing the ROS framework to shutdown. This error was discovered too late and the team had already switched to the if/else state machine.

V. Stop at a stop sign

One of the key aspects in building a fully autonomous self-driving car is to be able to track and detect objects around the vehicle. The challenge to stop at a stop sign lets us get a try of object detection on a small autonomous vehicle like ours.

There are several approaches to detect a stop sign with the camera, such as pattern recognition, image classification or even simpler way such as a shape detector to detect an octagon shape with the OpenCV FindContours and ApproPolyDP method to approximate contours in an image. We choose to use deep learning method to accomplish this task since we have been taught in class and this would be good learning opportunity for us.

We firstly tried several popular off-the-shelf pre-trained models, such as Google Tensorflow's Mobilenet Object Detection API[4], and Tiny-Yolo model[5] based on Darknet[6]. Both models are trained on Microsoft COCO dataset which has 80 real-world object categories including stop sign and sports ball. And both are open sourced light weight deep neural networks and can be adopted in micro-controller like a mobile device which has limited amount of computational power with competitive accuracy and FPS. Mobilenet and Tiny-YOLO are both being scaled based on their original models. Mobilenet has 28 convolutional and pooling layers, a FC layer, and softMax as the classifier while tiny-Yolo has total 9 conv-pool layers. Somehow, they are still too large for our Odroid XU4 platform and after experiments we found that with these models, near real-time detection was impossible with proper accuracy. Also training of these nural nets was difficult on a CPU laptop.

After consulting with Professor Heckman, we decided to create a small convolutional neural network model from scratch and train it on our own image data to build a binary image classifier which can classify a picture with stop sign or without a stop sign, using Python and Keras with Theano as backend.

Fig. 3. Sample data we collected using OCam

To train our stop sign detect deep learning model, we firstly need two sets of images: images containing a stop sign; images not containing a stop sign. We end up with taking pictures in the robotic lab with OCam as our training data due to following reasons: 1. OCam on Odroid can only take grey scale pictures with certain resolution, therefore we want the testing image be consistent with the training image to get accurate prediction. 2. It is challenging to find large set of images about stop signs.

We randomly collect 470 images in total of two classes Figure 3 and use these two sets of data to train our model as our first step.

The network we implement is LeNet(1998)[7], as shown in Figure 4. It's a simple network consisting of the input layer, 2 conv-pooling layers, FC layer and softMax as the classifier. It is a very simple neural network which used to classify hand written letters in greyscale images before. We use this network because it is fast. It can detect an image in real time and perform fairly well. We use ReLu as the activation function and SGD as the optimizer. We have also tuned the number of filters as well as the size of convolution

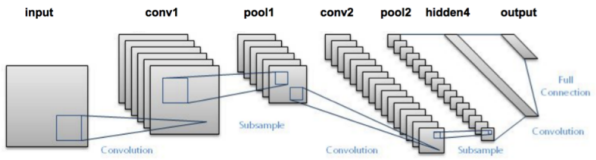

Fig. 4. LeNet neural network

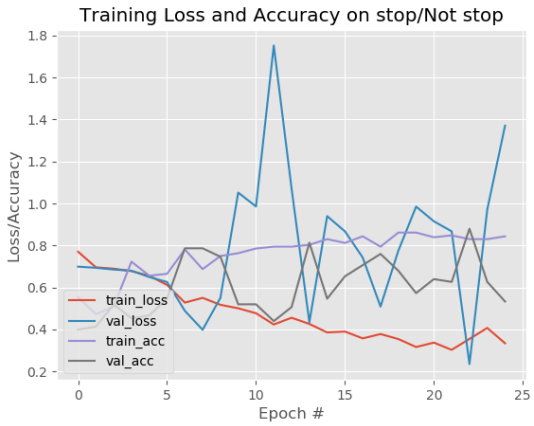

Fig. 5. training loss and accuracy

area in order to get interesting features from the image. The dropout rate is also important as to balance the overfitting issue. In Figure 5, we can see how the training loss and accuracy evolves as the training epoch iterates.

The challenges we met for this task reflect on both hardware part and software part. When it comes to the hardware, we found the OCam consumes the battery very fast. We can only test few rounds with fully charged battery before it goes too low to drive the camera. Regarding the software part, since we take the images by ourselves, all the training images looks very similar and we easily gets these images from same background.

VI. Electrical Components

A. ODROID XU-4

ODROID-XU4 board is a powerful, energy-efficient hardware made by Samsung. This is a good chioce since it offers open source support, the board can run various flavors of Linux, including Ubuntu 16.04 and Android 4.4 KitKat and 7.1 Nougat. Ubuntu 16.04 is the system we use. ROS can run in Ubuntu 16.04 stabily. Also, it has 2 USB 3.0 Host and 1 USB 2.0 Host can meet the requirements because our camera, IMU and Polulu all need USB interface.

It can boot from a MicroSD card or an eMMC module making use convenient. Our project need to work on Linux system with ROS and need to complete many tasks. Enough storage is important to us. ODROID-XU4 have 2 Gbyte DDR3 RAM is big enough for us to do all the challenges.

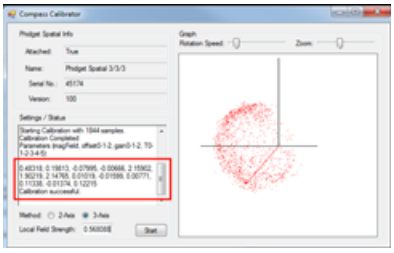

Fig. 6. IMU Calibration

B. IMU

The Phidget Spatial 3/3/3 IMU has a 3-axis compass, 3-axis gyroscope and 3-axis accelerometer. We combine IMU with oCam to do the visual-inertial SLAM. IMU was bonded to the camera so that we can combine space data with camera image. At first, the IMU need to be calibrated in order to get accurate compass information. We can calculate magnetic field value of your place on internet. Then put the magnetic field value into program that Phidget designed for us to do the calibration. With IMU connect to the computer and rotate the compass around such that red dots (Fig.6) being generated on the screen outline as much of a full sphere as possible. Then you can get parameter with calibrated data in it.

C. IR Sensor

IR sensors chosen are a cheap sensor to detect the distance to the wall. We set two sensors in front of our robot. One facing slightly to the left the other one facing slightly to the right. By computing the difference distance to left and right wall, we can set the plant set point to zero, the car will keep running in the middle of the corridor. When the robot getting to the corner, the difference will immediately change, we set the threshold as 200cm which means distance to left minus to the right is bigger than 200 or less than -200 means the car is in the corner. Then the robot will change to turning mode.

The IR sensor we use is GP2Y0A710K0F. It can measure distance between 100 to 550cm. Analog output type. One problem is that the sensor's output is not linear. Distance measurement characteristics is shown in Fig.7 We test the output depends on the distance and separate distance into three segments. Each segment has corresponding slope. In this way, we can get more accurate data. As you can see the distance it can measure is more than 1 meter. When the distance is less than 1 meter, the data confuse controller because it is same as data more than 1 meter.

D. Polulu

The Mini Maestro can directly connect to ODROID via USB. It has 18 channels, each channel can be used as general-purpose digital outputs, analog or digital inputs (channels 0-11 can be analog inputs; channel 12+ can be digital inputs). Polulu connects to the car chasis steering servo and motor ESC (electronic speed control) and makes controls of these units very simple. It sends a PWM signal to the ESC and servo controller from inputs received from the ODroid.

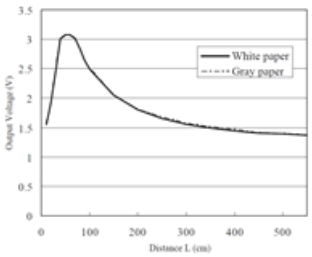

Fig. 7. IR sensor distance measurement characteristics

E. Voltage Divider

The battery we use are two 7.2V 6 cell Ni-MH. The Odroid, polulu and IR sensor all need 5V power. Drok voltage regulator can regulate voltage from 3 40V to 1.23 37V. We set the output to 5V power for our robot. The Output current can be up to 3A. XU4 board requires a 5V/4A DC power source which is more current than voltage divider can produce. This provided another layer of challenge when working off of battery power. The odroid would shut processes down to protect itself thus if the batteries were not above 7.8 volts reliability of the entire system would go down or simply not work.

VII. Mechanical Part

A. Bumper

We made bumper to protect our car from high speed impacts with the wall during testing. Since we wanted to be competitive in the race we needed to risk running at faster speeds. It was hard for us to keep up with car and catch it before it slammed into the wall. With a bumper in front of the car damage to the electronic device and chasis was kept to a minimum. IMU and IR sensor are easily damaged. We have broken one of our IR when testing the car in the corner. The bumper can save lots of work repair the robot. The construction is of an acrylic board with foam tube taped together as seend in Fig.8. Holes were drilled in the stock car bumper so that we can use screw to screw the bumper.

Fig. 8. Bumper

B. Platform

All of our electric board except ESC are putting on the acrylic board cut by laser. We reserve some big holes at front and back of our robot so we can make the wire connect to ODROID can go downside of our platform and go up again near the XU4. This can make our robot looks concise.

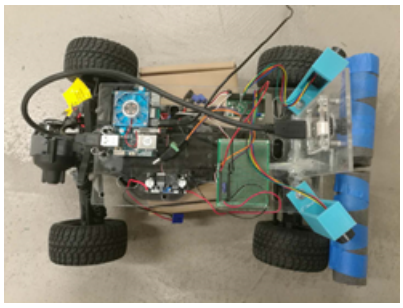

Fig. 9. Platform

C. IR Stand

At beginning of our project. We used a simple and quickly printed IR stand made from PLA, but we found that the position of screw hole at side of stand was not stable. It would turn a little bit when the robot hits the wall. Then we drop this plan and made a new one with screw hole at back of stand shown in Fig.10. The new stand is much more stable than the old design. We use 3D printing to print out the stand with PLA.

D. Camera Stand

The camera stand is more complex than the IR stand. Because the camera is more sensitive than IR. Good quality of image is important for us to get better result with visual SLAM. If the stand is too high, it will have more violent shaking at top. If the stand is too low, the camera will get less features from the image because nearly half of the image is ground. The color of ground doesn't change too much. Also, there is less corner and line on the ground. We set the height as 16cm so that we can get good image features without too much shaking. To decrease the shaking, we add rib to fix the stand. We use hot glue combine rib and stand with our platform.

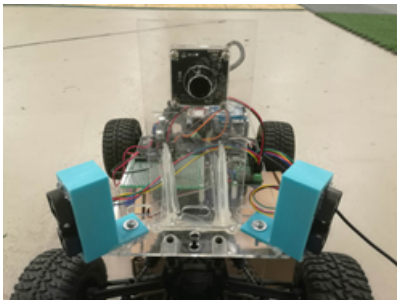

Fig. 10. IR and oCam stand

E. 3D printing

The most commonly used types of personal 3D printing are ABS and PLA. We build two IR sensor and Camera holder with PLA. Because PLA have less shrink than ABS. When ABS cools after printing it shrinks approximately 8%.

PLA shrinks approximately 2%. It is not as much as ABS but big enough have influence on our IR stand. We first drawing the stand with Solidworks use parameter exactly same as IR sensor. After we print out the stand we found that we can not put IR sensor into it. Then we resize the stand and print it again get the one can fit IR sensor in it. Counting on the shrinking in the design part can improve the success rate of 3D printing and waste less material and time.

VIII. Discussion

The car performed well even though the implementation was not very complex. The race results ended with an average time of approximately 26 seconds over three runs. One run failed due to a cameraman being in a position that interfered with the IR sensor during cornering. The state machine worked reasonably well but can be considered rudimentary. Added complexity to the state machine would lead to a much better performance and robustness of the system. This can occur through the use of added sensor information such as IMU data and discrete IR sensor readings to determine the state of the car. Filtering of the IR sensor data can possibly be done through the use of an extended Kalman filter to attenuate the erroneous readings. One short coming of the system was that the PID controls were very sensitive to the velocity of the car. And that in itself varied due to the voltage charge on the batteries. Meaning pre-run checking of the velocity needed to occur to ensure we were operating in velocity range that the PID controllers can accommodate for. Another issue for the PID controllers was that by their nature they do not tend to the most optimal path. Implementation of an alternative control scheme such as model predictive control could resolve this, further increasing performance.

What also hindered our system is the placement of our IR sensors. Even though their placement made tracking to the center extremely easy it also impeded our ability to detect corners accurately. Their placement on the car made doorways look like corners thus greatly hampered our ability to improve our time. Switching the IR sensors to a single forwarded and one to 45 degree to the right position would have allowed us better control of our system due to having better conditionals to switch to corner states.

The stop sign detection challenge ended successfully. The car accomplished the goal of detecting a stop sign and stop, however improvements can be made. The detector had issues with background that contained a lot of features. For example, when pointed to an area with a lot of features it would detect a stop sign when none was present. This could be due to the neural network not beginning trained with photos with backgrounds that had a lot of features. Another issue was that the camera runs at 45FPS, but detecting rate is about 3FPS, which will result in accumulating of the messages in the queue and detector only detects the delayed image captured by the camera. We handle this by detecting once every five images. But this will also delay the car stop when the stop sign is already shown.

VINS-Mono was so robust that we did not need to get the extrinsic parameters between IMU and camera to make it work. If we could get the accurate values on our own, the performance should be better than what we have. Besides, the drifted spare map was useless, we should be able to fix this issue by further improvement.

IX. Code

The github repo for the code is: `https://github.com/AdvancedRoboticsCUBoulder/asurada`

X. CONCLUSIONS

Our robot can complete the track in approximately 26 seconds reliably. Our visual inertial-SLAM works well with camera and IMU together on laptop. We can make the car get 3D information of the environment around it. State machine can make the car turn smooth during the corner with different PID, however adding complexity would greatly increase performance and robustness of the system. Stop sign challenge is not quite stable because the training dataset is not big enough, plus the data is monotonous since we collected them by ourselves in the lab with same background and same lighting. In the future, we can improve the performance of the model by training with more datasets. The speed also need to be slow enough to keep the successful rate. Camera stand and bumper works fine. But IR stand easily breaks when hitting the wall. Overall the ability of the car to meet the goals of the challenges was a success due to the interdisciplinary teamwork of the team.

XI. ACKNOWLEDGEMENT

We want to express our thankfulness to Prof.Heckman and for his patience, support, advice and encouragement. Also we need to say thanks to our TA Jeff for his continuing help to us.

APPENDIX

References

[1] Miller, John (19 August 2014). "Self-Driving Car Technology's Benefits, Potential Risks, and Solutions". theenergycollective.com. Retrieved 4 June 2015.

[2] Sünderhauf, Niko. Robust optimization for simultaneous localization and mapping. Diss. Technischen Universitat Chemnitz, 2012.

[3] Magnabosco, M., Breckon, T.P. (February 2013). "Cross-Spectral Visual Simultaneous Localization And Mapping (SLAM) with Sensor Handover" (PDF). Robotics and Autonomous Systems. 63 (2): 195–208. doi:10.1016/j.robot.2012.09.023. Retrieved 5 November 2013.

[4] Andrew G. Howard ,Menglong Zhu ,Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". CoRR: abs/1704.04861.2017. URL:http://arxiv.org/abs/1704.04861.

[5] Joseph Redmon,Santosh Kumar Divvala,Ross B. Girshick,Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". CoRR:abs/1506.02640. 2015. URL:https://dblp.org/rec/bib/journals/corr/RedmonF16.

[6] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016

[7] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998d). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

[8] VINS-Mono, `https://github.com/HKUST-Aerial-Robotics/VINS-Mono`

[9] ORB SLAM2, `https://github.com/raulmur/ORB_SLAM2`

Compare Different Deep learning algorithm in traffic sign detection system

***Abstract*— In this report, we will introduce different ways of deep learning algorithm working on traffic sign detection and recognition. Including k-d trees and random forests, multi-scale convolutional networks, haar cascade and deep learning confusion method and deep neural network.**

I. INTRODUCTION

Most of us have experience with driving. With or without Google map, we have to see the traffic sign on the road. Such as stop sign, one way road sign, speed limiting sign and so on. Miss one of them may bring you serious accident. Sometimes we are looking for the street we want to turn in. When we can see the road sign clearly, it may too close to the crossroad so we have no time to change line. Missed the road is not big deal. But we can see it is not safe to drive with focusing on find traffic signs or road signs. Drivers are less likely to notice pedestrian signs while focusing speed limit signs. This can be danger to pedestrians. If we can build a new system help people detect traffic sign and tell driver or show driver by head-up display will improve the safety. Driving will be easier for driver.

Autonomous vehicles are developing very fast today. We can see more and more news about self-driving car testing on the road. We believe in the future autonomous vehicle will be part of our life. More and more company are working on study self-driving car, including Tesla, BMW, Honda, Ford and so on. Traffic sign recognition system is also important to those vehicles. Self-driving car need to obey those traffic sign like human driving vehicle.

The accuracy of detect the traffic sign should be very high to keep the car safely driving on the road. At least is should be higher than human performance. According to [3] human accuracy with speed limit is 97.63%. Danger sign is 98.67%. In the next four section will show four different deep learning methods result.

In many recent object recognition systems, feature extraction stages are generally composed of a filter bank, a non-linear transformation, and some sort of feature pooling layer[1]. Following systems use one or two of the feature stages.

II. MULTI-COLUMN DEEP NEURAL NETWORK ALGORITHM

Deep neural network(DNN) consists of a succession of convolutional and max-pooling layers, and each layer only receives connections from its previous layer[4]. Hierarchical features extractor will map pixel intensities of the input image into a feature vector to be calssified by 2 connected layers. Each convolutional layer performs a 2D convolution

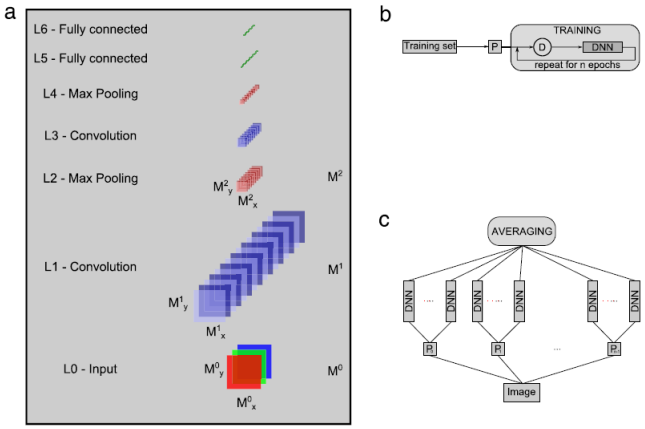

Fig. 1. DNN architecture

of its multiply input maps with a filter. The biggest architectural defference between DNN and the CNN is the use of max-pooling layer given by maximum activation over non-overlapping rectangular. Kernel size of convolutional filters and max-pooling rectangles are chosen such that either the output maps of the last convolutional layer are down-sampled to 1 pixel per map[4]. Training DNN need a given dataset. The dataset then be continually distorted during training. At last, they form an MCDNN by averaging the output activations of several DNN columns. Various columns can be trained on the same inputs, or on inputs preprocessed in different ways[4].

III. K-D TREES AND RANDOM FORESTS

[9] evaluate the performance of K-d trees and Random Forest for traffic sign classification using different size Histogram of Oriented Gradients(HOG) descriptors and Distance Transforms. A K-d tree is a binary search tree organizing k-dimensional data points. The algorithm is like directed-acyclic-graph that can build hyperplane by features of test image. One side of subtree contain data smaller than the feature. The other one subtree contain data larger than feature. The division is repeated until each data point is represented by a leaf[9]. K-d tree is a nearest-neighbor-base search tree that is more suitable for challenge dataset due to the large variation in the number of training samples of the different classes.

Random forest is formed by random trees. They randomly chose subset number of training samples from original training set. Then repeating this process to get different random

Traffic Signs	Test	Recall	Precision
Speed limit	127	99.21 %	99.13 %
Danger signs	119	99.16 %	99.17 %
Unique signs	50	98.89 %	99.04 %
Mandatory signs	135	98.84%	98.95%
Derestriction signs	120	98.82%	98.92%
Derestriction signs	140	99.12%	99.06%
Other prohibitory signs	115	99.15%	99.08%

Fig. 2. Recall and precision results for trafic sign detection

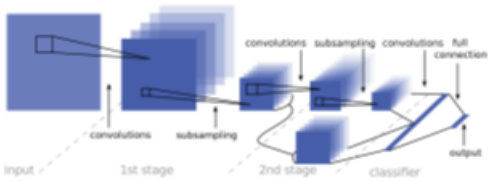

Fig. 3. A 2-stage Convolutional networks architecture

trees. The advantage is that the process is fast, easy to implement in a distributed computing environment. Random forests are less sensitive to variations in the background than the K-d tress because of the random selection of the variables[9].

With test 43 types of traffic signs. The k-d tree achieves a classification result of 92.9% accuracy and random forest get 97.2% accuracy.

IV. Haar cascade and deep learning confusion

Paper[2] introduce a new algorithm that begins with Haar cascade detection algorithms in the early stages of analysis, then brings in deep learning neural networks in the final stages. Haar cascade detection algorithm is fast, it can quickly detect many traffic signs in one images, but can bad at distinguish between traffic signs and similar-looking objects in some cases. Machine learning algorithms can handle complex pattern recognition, but work slowly in real-time road signs detection. As shown in [2], they separate the detection process into two steps. First, using scanning window with Haar cascade detector to find features possible be traffic signs to reduce the computational region in hypothesis generation step. Then, use deep learning classification for verification to improve the accuracy. They trained deep learning classifier in the form of convolutional neural networks with approximately 10,000 identities with 160 hidden identity features in the top layer[2].

V. Multi-Scale Convolutional Networks

Convolutional networks are biologically-inspired multi-stage architectures that automatically learn hierarchies of invariant features using a combination of supervised and unsupervised learning(see Fig.3)[8]. In traditional convolutional networks, only the last stage of output is fed to a classifier. In paper [8] they fed the outputs of all stages into classifier to improve the accuracy. This allows the classifier to use not only high-level features but also low-level features, which tend to be more local, less invariant. Convolutional networks can be run at very high speed at low-cost, small form parallel hardware based on GPUs. The results they got is 98.97% with full trained supervision on the color vision. And 99.17% accuracy with additional grayscale established.

VI. Conclusion

K-D trees and random forests algorithm have the lowest accuracy. It is lower than human performance. Other three algorithm get result better than human. Haar Cascade and deep learning confusion algorithm get highest accuracy and because it use Haar Cascade algorithm. The computation is much less than only use deep learning algorithm. The speed of detect traffic signs is fast.

References

[1] Jarrett, Kevin, Koray Kavukcuoglu, and Yann LeCun. "What is the best multi-stage architecture for object recognition?." Computer Vision, 2009 IEEE 12th International Conference on. IEEE, 2009.

[2] Abdi, Lotfi, and Aref Meddeb. "Deep learning traffic sign detection, recognition and augmentation." Proceedings of the Symposium on Applied Computing. ACM, 2017.

[3] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs.computer: Benchmarking machine learning algorithms for trafic sign recognition. Neural networks, 32:323-332, 2012.

[4] CireAn, Dan, et al. "Multi-column deep neural network for traffic sign classification." Neural networks 32 (2012): 333-338.

[5] K. Lenc and A. Vedaldi. R-cnn minus r. arXiv preprint arXiv:1506.06981, 2015.

[6] D. Shinar, Traffic Safety and Human Behaviour. Bingley, U.K.: Emerald, 2007.

[7] Mogelmose, Andreas, Mohan Manubhai Trivedi, and Thomas B. Moeslund. "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey." IEEE Transactions on Intelligent Transportation Systems 13.4 (2012): 1484-1497.

[8] Sermanet, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale convolutional networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[9] Zaklouta, Fatin, Bogdan Stanciulescu, and Omar Hamdoun. "Traffic sign classification using kd trees and random forests." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[10] Fan, Jialue, et al. "Human tracking using convolutional neural networks." IEEE Transactions on Neural Networks 21.10 (2010): 1610-1623.

Human-robot interaction with autonomous vehicles

***Abstract*— In this report, we will analysis human-robot interaction(HRI) system in three aspects, developer-robot interaction, user-self-driving cars(SDC) interaction and pedestrian-SDC interaction.**

I. INTRODUCTION

Improvements of SDC technology in the last decade makes autonomous driving gone from maybe possible to definitely possible. These new capabilities will have profound global impacts that could markedly change society, not to mention the significant improvements they bring to the overall efficiency, convenience, and safety of our roadways and transportation systems[1]. To take place of normal car on the road, autonomous vehicles should be able to do all the things a normal car with driver can do. A normal car with driver on the road have lots of interactive with people. According to facial expression and hand gesture, people want to cross the road will understand if it is safe to cross now. Passenger knows where they are going, why the car stops now, how long it takes to arrive destination by communicate with driver. Autonomous vehicles do not have driver. It is driver itself. So SDC should have same communication with people like driver can do. Thats why we need HRI system.

In this report, we separate people HRI need to communicate with into three groups to analysis HRI system. The first one is developers. It is important to feedback vehicles status to developers so that they can improve performance of the autonomous vehicles. Second group is user, people sitting in the car had to believe the car they are taking can successfully send them to their destination. HRI is to help passengers believe it is safe in this vehicle. Third one is pedestrian. HRI also need to let pedestrian and other vehicles know what the car coming is going to do.

II. DEVELOPER-ROBOT INTERACTION

In traditional car manufacturing industry, test drivers assess vehicle performance for manufacturers. In order to make SDC perform better, therefore, they need to be able to take advantage of human skills and to benefit from human advice and expertise. We can use collaborative control system for teleoperation. Instead of a supervisor dictating to a subordinate, the human and the robot engage dialogue to exchange ideas, to ask questions, and to resolve differences[2]. Take Fig.1 as example. The figure shows a robot module that detect motion from a stream of camera images. As the robot see the image difference, it will notify user and wait for user. Then take action as the user response.

To build a collaborative control system, the robot must have self-awareness[2]. introduce four kinds of features SDC should have. First, it is capable of detecting limitations,

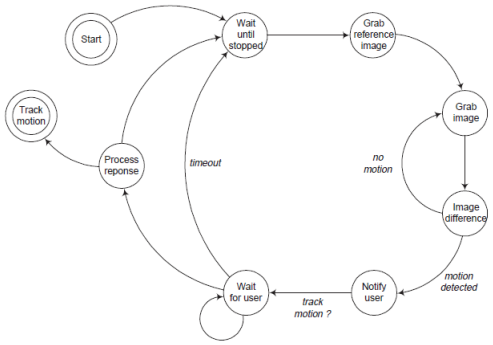

Fig. 1. Motion Detector state machine[3]

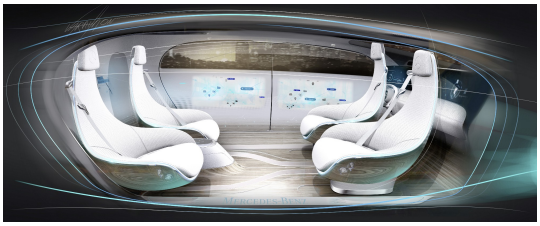

Fig. 2. An interior concept created by Mercedes-Benz

determining if it should ask for help, and recognizing when it has to solve problems on its own. Second, the robot must be self-reliant which means it can avoiding hazards, monitoring its health, taking action to be safe when necessary when there is no developer provide accurate information. Third, the system must support dialogue. Developer can communicate effectively with SDC. SDC must be able to ask questions and to judge the quality of responses received. At last, the system must be adaptive. The robot has to be able to adjust its behavior as needed, e.g., asking different questions when it needs designer to answer.

III. USER-SDC INTERACTION

Some experts predict that we can save millions of lives and grant mobility to all just by removing humans from the drivers seat. But the difference between theory and practice comes down to this: People are downright scared of robot cars. In fact, a recent AAA study found that 75 percent of Americans are afraid to ride in self-driving cars[4]. Intel invited some consumers who had no previous driverless car experience to study their feeling of five actions including requesting a vehicle, starting a trip, making changes to the trip, handling errors and emergencies, and pulling over and exiting. In the report, they identified seven areas of tension.

Fig. 3. smiling self-driving car(Jeremy Hsu Sep 2016)

We will take some of them as example to introduce how HRI reduce those anxiety. First, people were concerned about the ability of autonomous vehicle handle nuanced situations. Second, passenger feeling some of the alerts and communications might be bothersome. Third, understanding how the technology functions works. Fourth, people wondered if they could use their own voice to communicate with the car, especially if needing to make a detour, change destination, or take off right now. As to the first, third, and fourth problem people concerned. We can show Normans seven stages of interaction[6] to passengers. These seven stages are iterated until the intention and goal achieved or the user decides that the intention or goal has to be modified[7]. With those information, people will understand how the car works. With understanding how car works, they feel more comfortable with the autonomous car. For example, when slowing down or accelerating, show in the LED screen there is a stop sign ahead so we slow down now or we are entering the highway so we accelerate.

Self-driving car will have totally new seats arrangement. The screen in the car will be much bigger than traditional car. Like Fig.2 shows, people will face to each other in the car. There will be more room for designer to decide what should be shown to user and where the information should show up depends on how important the information is.

IV. Pedestrian-SDC interaction

There is a huge amount of unspoken interactions between drivers and the world around them. Gesturing to a pedestrian to cross the street, horn to warning other driver get into one-way street in wrong direction. Its important for SDC to engage in that interactions as well. Drive.ai is working on LED signs on the vehicle that use text and emoji-like pictures to communicate. They also working on auditory feedback too. The team think horns do not change direction or volume which is bad. They are working on an advanced version of its auditory feedback, allowing the car to ”see the context” of the situation and emit a ”more socially appropriate” honk. In [8], Google reveals that it’s been training the cars to honk at the absent-minded humans around them when the situation demands. If another vehicle is slowly reversing towards us, we might sound two short, quieter pips as a friendly heads up to let the driver know we’re behind. However, if there’s a situation that requires more urgency, we’ll use one loud sustained honk[8]. As in the Fig.3 shows. SEMCON working on The Smiling Car system when the self-driving cars sensors detect a pedestrian, a signal is sent to a display at the front and a smile lights up that confirms that the car will stop for the pedestrian. The team use cameras with systems for eye tracking technology, so SDC read a gaze directed toward the car then provide immediate feedback through a smile lighting up on the grille.

V. Conclusion

HRI have different systems for different people. When interact with designer, the system should have ability to show the problem vehicle met and execute designers command. Interact with pedestrian just need to show people what the car is doing and going to do. User is more complicate, the system need to be able to execute users command like designer do, also, the system should show lot of information about the car and road situation to let user know all the things is in control.

References

[1] M. Daily, S. Medasani, R. Behringer and M. Trivedi, ”Self-Driving Cars,” in Computer, vol. 50, no. 12, pp. 18-23, 2017.

[2] Fong, Terrence, Charles Thorpe, and Charles Baur. ”Collaboration, dialogue, human-robot interaction.” Robotics Research. Springer, Berlin, Heidelberg, 2003. 255-266.

[3] Fong, Terrence, Charles Thorpe, and Charles Baur. Collaborative control: A robot-centric model for vehicle teleoperation. Vol. 1. Carnegie Mellon University, The Robotics Institute, 2001.

[4] https://newsroom.intel.com/editorials/when-will-you-be-ready-get-driverless-car/ August 24, 2017

[5] Fong T (2001) Collaborative control: a robot-centric model for vehicle teleoperation. PhD thesis, Carnegie Mellon University, Pittsburgh. CMU-RI-TR-01-34.

[6] Norman, D. 1986. Cognitive Engineering in Donald Norman and Stephen Draper (Eds) User-centered design: new perspectives on human-computer interaction, Erlbaum Associates: Hillsdale, N.J, 31-62.

[7] Scholtz, Jean. ”Theory and evaluation of human robot interactions.” System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on. IEEE, 2003.

[8] Google Self-Driving Car Project Monthly Report May 2016 https://static.googleusercontent.com/media/www.google.com/en//selfdrivingcar/files/reports/report-0516.pdf