EECE 5644 Final Project Report
Machine Learning For Computer Vision

Chris Swagler, Tae Yang, Gary Lvov

Field Overview

Computer vision, a field deeply intertwined with machine learning, revolves around creating algorithms and systems that allow computers to interpret visual data from the physical world. It entails various techniques that aim to mimic human vision, extract meaningful information from images and videos, and enable machines to recognize objects or scenes. Although learning based techniques play an essential role in modern computer vision, many analytical techniques provide critical functionality that can be used independently or in conjunction with learning based techniques.

One fundamental aspect of computer vision are feature detectors and descriptors. These techniques include the Harris Corner Detector [1], which finds corners in an image based off of relative pixel intensities. The Harris Corner Detector is often utilized with a checkerboard of known dimensions to ascertain camera intrinsics. Similarly, the SURF[2] Feature Detector/Descriptor provides a unique representation of all detected features, thus enabling the computation of homography between two images to allow for stitching of images of the same scene. It identifies features that remain invariant to alterations in scale, rotation, and affine transformations, exhibiting similarities to other methods such as SIFT [2]. Another critical technique, the Canny Edge Detector [3], identifies edges in an image by optimizing the intensity gradients of blurred images and finalizing the edges via Hysteresis.

Complementing these detection techniques, determining optical flow[4] allows for analyzing feature movement. Optical flow assumes that neighboring pixels exhibit similar motion and that pixel intensity remains consistent from frame to frame. By calculating the relative motion of each pixel, these techniques can be applied for video stabilization and target tracking of features within an image.

Multi-View Geometry allows utilizing known camera intrinsics and extrinsics for multiple cameras to compute depth information from captured images. Techniques such as Stereo Vision use two cameras to generate images from slightly different viewpoints, allowing for the computation of disparities[5] and the creation of three-dimensional point cloud representations of a scene. Alternatively, sensors such as LiDAR can generate point clouds for point cloud processing techniques like Simultaneous Localization and Mapping[6] in robotics or autonomous driving.

Despite the efficacy of these classical techniques in capturing spatial-temporal information, they are limited in their inability to grasp high-level semantic understanding or extract complex contextual information. However, the advent of learning-based methods, such as Convolutional Neural Networks[7], Recurrent Neural Networks[8], and Transformer-based models[9], has revolutionized the field. These supervised learning techniques, trained on annotated images, have demonstrated remarkable success in various computer vision tasks, enhancing the field's ability to glean deeper, more meaningful information from visual data.

Object Detection in Autonomous Vehicles (AV)

Object Detection Overview

The first computer vision technology to investigate is object detection. It is a task of identifying and localizing objects within digital images or video frames which are basically a sequence of digital images.

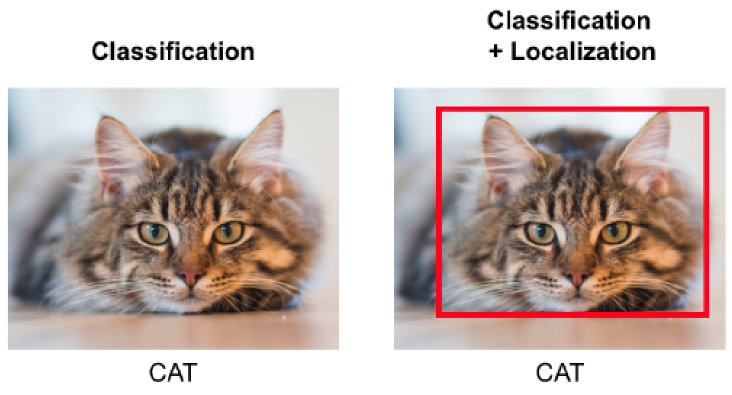


Figure 1: Single object classification and localization [10]

Figure 1 shows an example of classification and localization technology used in object detection on a single object. Classification is to assign a label or category to an input image. A machine learning model is trained using a labeled dataset and learns to extract meaningful features from the images and map them to the corresponding classes. During inference, an image is fed into the trained model which produces a probability distribution over the predefined set of classes. The class with the highest probability is then assigned as the predicted label for the input image. However, if there are multiple objects to be recognized in one image, exact regions of the objects should be located. This is where localization technique is utilized. Localization determines bounding boxes that tightly enclose the objects such as the red box in Figure 1.

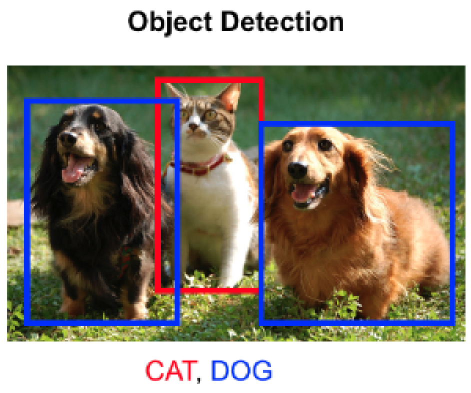


Figure 2: Multiple object detection [10]

Combining classification and localization, multiple objects in a single image can be detected with bounding boxes and corresponding classes as shown in Figure 2. To explore further details of object detection, localization for multiple objects can be divided into two steps [10]. The first step is to generate a set of potential object proposals or candidate bounding boxes through a region proposal network (RPN). These proposals act as potential regions of interest (ROIs) that may contain objects. Subsequently, the bounding box coordinates of the proposals are refined to tightly fit the objects present in the image. This is achieved through bounding box regression. The algorithm learns to predict adjustments or offsets to the candidate bounding box coordinates, adjusting their position, size, and aspect ratio to more accurately localize the objects. Finally, these refined proposals are fed into a classification network which works the same as in the single object detection in Figure 1.

Evolution of Object Detection

Until 2012, traditional techniques leveraged hand-crafted heuristics to extract visual features and manually tuned parameters to combine these features for inferences and decisions. These techniques are simple yet effective for many cases [11]. However, they are often not generalizable and difficult to configure. As explained in Field Overview, more powerful and accurate automated visual recognition techniques were enabled by deep neural networks that can handle high-dimensional data with the advent of the deep learning era. In particular, CNNs excel at image recognition and pattern identification due to its ability to capture space invariance of shapes in images.

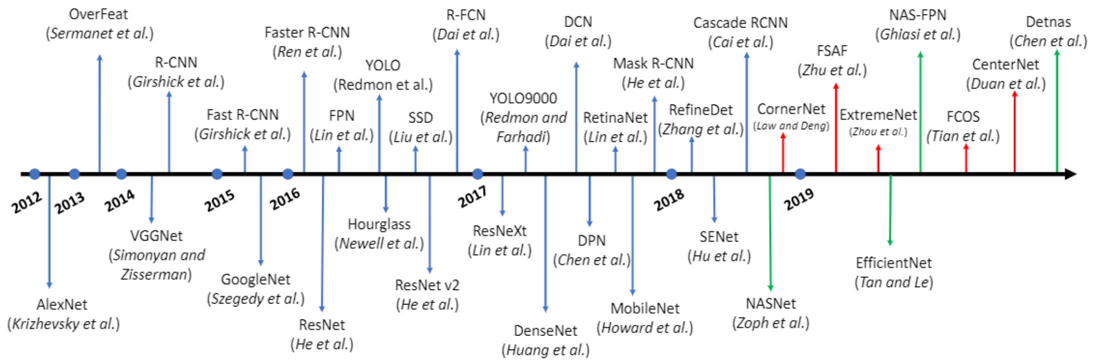


Figure 3: Evolution of deep learning based object detection algorithms [12]

There have been various object algorithms developed since 2012 as shown in Figure 3. Main purposes of the algorithms are focused on improving accuracy, speed, real-time performance, and the ability to detect objects at various scales [12].

Among these algorithms, Faster R-CNN, YOLO, SSD, and RetinaNet are dominant ones in AV [13]. In this application, accuracy and responsiveness are highly significant at the same time. AVs need to identify other vehicles, traffic agents and signs correctly to decide their next control and path. Furthermore, they need to perform the detecting task as soon as possible since driving environments are dynamic and rapid. That's why the four algorithms are mainly used for most of the AV companies' object detection models.

Taxonomy of Object Detectors

The object detectors can be classified based on the underlying network architecture they employ. There are two types of network which are two-stage detectors and single-stage detectors [13]. For example, R-CNN variants are two-stage detectors and YOLO, SSD, and RetinaNet are single-stage detectors.

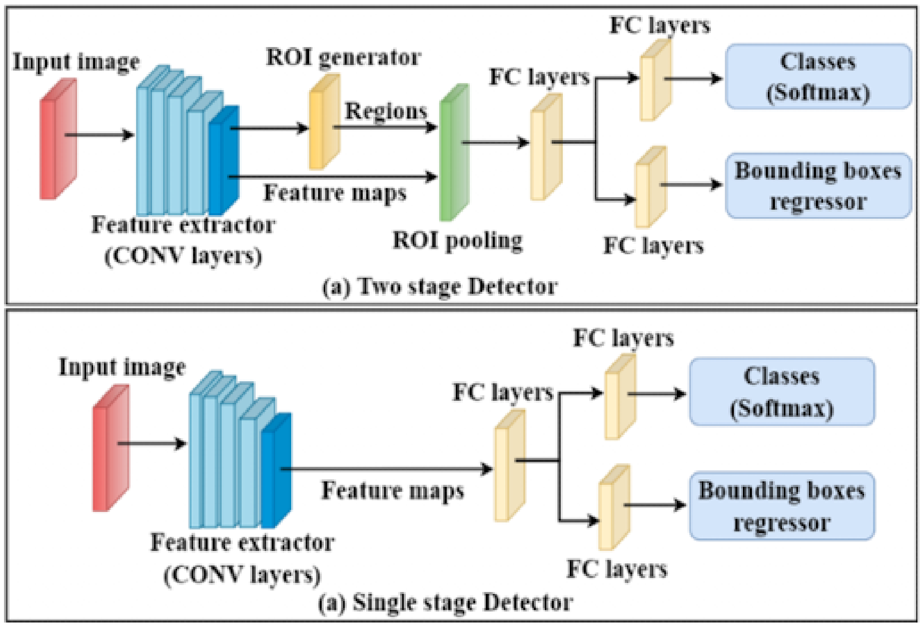


Figure 4: Two-stage vs Single stage object detector diagram [13]

Two-stage detectors use region proposal methods such as selective search or RPNs with image features from feature extractor using convolutional layers as shown in Figure 4-(a). The ROIs are collected by a ROI pooling whose purpose is to extract fixed-sized feature maps from variable-sized regions. The ROI pooling aligns and reshapes the regions proposal into a consistent format for subsequent classification and bounding box regression. Fully connected (FC) layers are the final layers that execute linear transformation with weights in the network architecture. They provided a flexible and trainable framework for capturing high-level representations and making object-level predictions based on the extracted ROIs. On the other hand, single-stage detectors directly predict class probabilities and bounding boxes in a single pass through the network as shown in Figure 4-(b). Instead of using the ROI generator, they operate on a dense grid of anchor boxes that cover the spatial locations and scales of potential objects in the input image. At every anchor box location, similar predictions as in the two-stage detectors are implemented. The anchor boxes act as priors in the FC layers and their coordinates are adjusted along with the class and bounding box predictions.

The single-stage detectors are generally faster than the two-stage detectors due to their direct prediction approach. However, the two-stage detectors often achieve higher accuracy, especially for small objects or in cases with complex scenes, due to their multi-stage nature and refined bounding box regression. Some of the detectors are further explored in detail to understand their principles better and tested with custom dataset in the following sections.

Detectors Performance Test

For the following testing, RetinaNet and YOLO were selected as they were fast and appropriate for object detection in AV. Vehicle dataset were used to train and test the detector models as if the models were for AV to detect different types of vehicles on the roads.

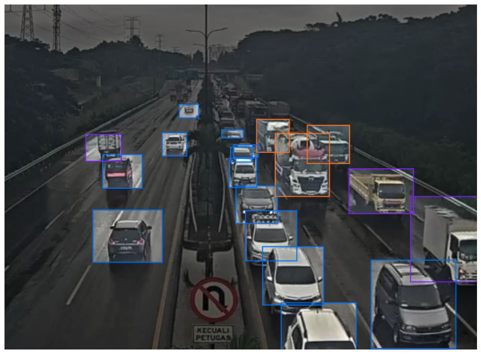


Figure 5: Example of vehicle dataset

The dataset provided pre-labeled images. There were 2634 train images, 966 valid images, and 458 test images. The vehicles in the images were classified as one of the 12 classes which were big bus, big truck, bus-l-, bus-s-, car, mid truck, small bus, small truck, truck-l-, truck-m-, truck-s-, and truck-xl-.

RetinaNet

RetinaNet is famous for its ability to address a class imbalance which refers to an uneven distribution of objects across different classes in the training dataset from YOLO and SSD when detecting small objects which are common challenges for the single-stage detectors. The small objects have less features extracted, so RetinaNet uses a focal loss function during training and a separate network for classification and bounding box regression. The focal loss functions apply a modulating term to the cross entropy loss in order to focus learning on hard misclassified examples such as the small objects.

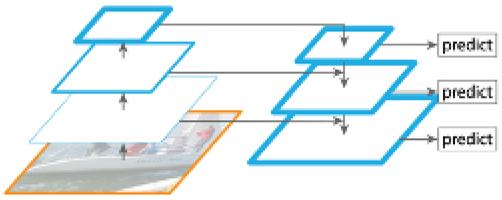
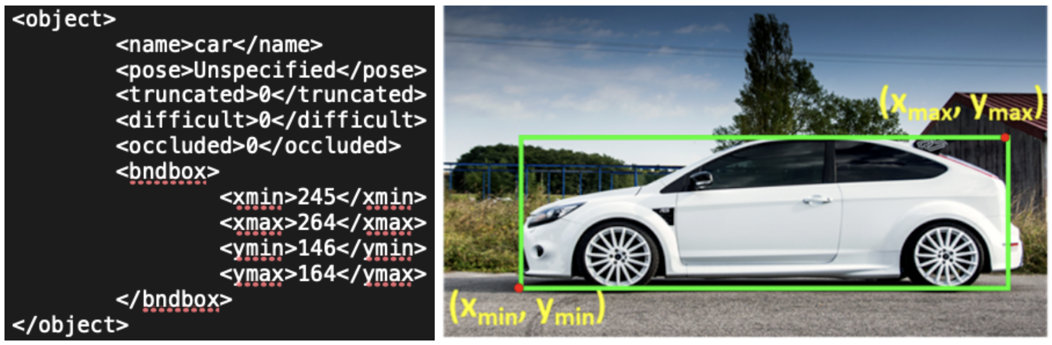


Figure 6: RetinaNet diagram [14]

The architecture of RetinaNet has a rich multi-scale feature pyramid from an input image and performs prediction at each scale as shown in Figure 6. In that way, it can be semantically strong at all scales.

For the RetinaNet model training, Pytorch RetinaNet and Pascal VOC dataset format were used.



```
<object>
        <name>car</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
                <xmin>245</xmin>
                <xmax>264</xmax>
                <ymin>146</ymin>
                <ymax>164</ymax>
        </bndbox>
</object>
```

Figure 7: Example of Pascal VOC dataset

Pascal VOC datasets provide xml label files as shown in Figure 7. The label files include classes and coordinates of bounding boxes for each object in the images. 50 epochs were performed for the training. One epoch is when all the training data is used at once. The number of epochs is relevant to a convergence of weights of the models.

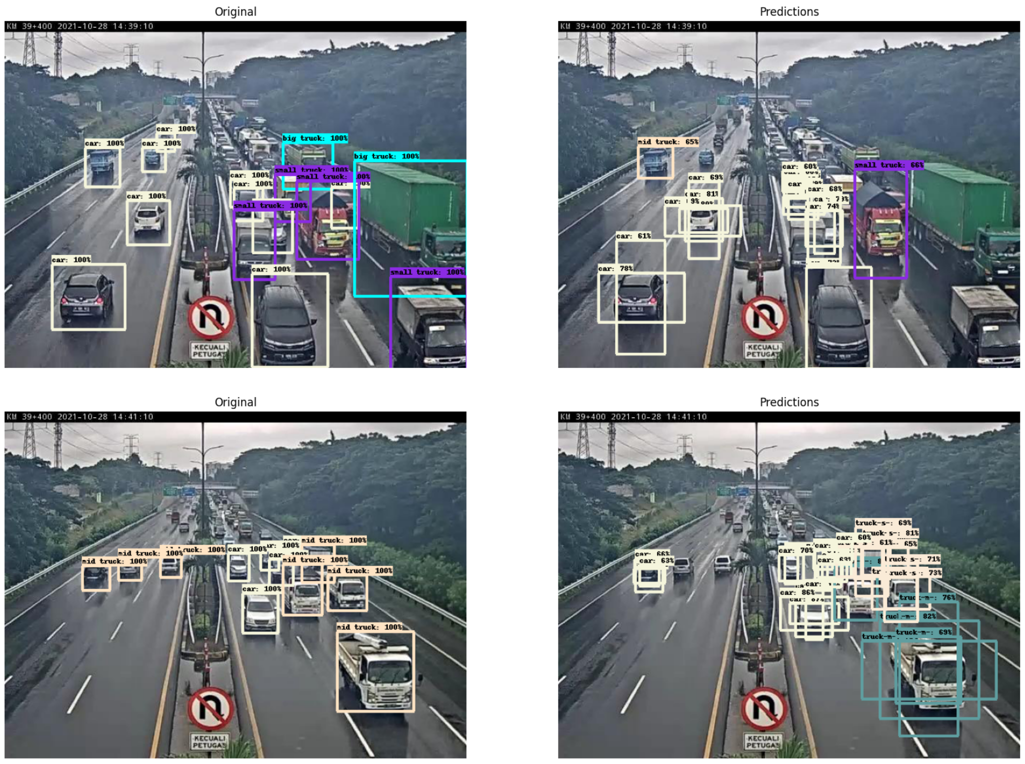


Figure 8: Examples of testing results of RetinaNet

The training result showed a classification loss of 0.112, box regression loss of 0.0385, and validation loss of 0.515. The classification and box regression were trained satisfactorily, but the validation loss was high which meant there was an overfitting during the training. Thus, the training results in Figure 8 demonstrated some misclassification and duplicate bounding boxes on a single object.

YOLOv5

YOLO stands for You Only Look Once to emphasize its speed in object detection. It has many variants such as v1, v2, v3, and so forth. For the testing, YOLOv5 was selected since it was one of the latest versions and proved its performance on many datasets throughout open source communities. YOLOv5 is famous for proposing further data augmentation and loss calculation improvements. Data augmentation is deliberately transforming training images by cropping, resizing, and flipping to increase the diversity of the dataset. This improves the model's generalization ability and robustness. Additionally, Auto-learning bounding box anchors are featured in the detector to adapt to a given dataset. Anchor boxes are usually fixed boxes for calculating bounding boxes. Auto-learning bounding box anchors are adaptive to datasets in order to enable fast converging bounding box calculation.

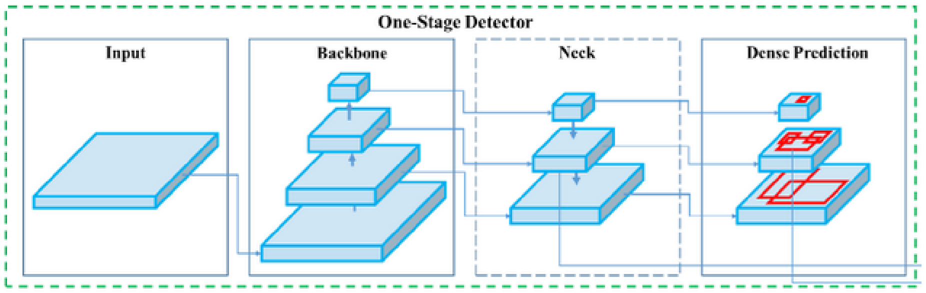
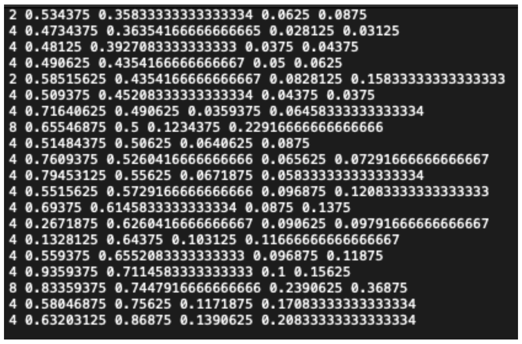


Figure 9: YOLOv5 diagram

The architecture of the detector is similar to that of RetinaNet having a rich feature pyramid to generalize well to objects on different sizes and scales and predicting classes and bounding boxes at each neck layer in Figure 9.

For the YOLOv5 model training, Ultralytics YOLOv5 and TXT annotations dataset format were used.



```
2 0.534375 0.35833333333333334 0.0625 0.0875
4 0.4734375 0.36354166666666665 0.028125 0.03125
4 0.48125 0.39270833333333333 0.0375 0.04375
4 0.490625 0.43541666666666667 0.05 0.0625
2 0.58515625 0.43541666666666667 0.0828125 0.15833333333333333
4 0.509375 0.45208333333333334 0.04375 0.0375
4 0.71640625 0.490625 0.0359375 0.06458333333333334
8 0.65546875 0.5 0.1234375 0.22916666666666666
4 0.51484375 0.50625 0.0640625 0.0875
4 0.7609375 0.52604166666666666 0.065625 0.07291666666666667
4 0.79453125 0.55625 0.0671875 0.05833333333333334
4 0.5515625 0.57291666666666666 0.096875 0.12083333333333333
4 0.69375 0.61458333333333334 0.0875 0.1375
4 0.2671875 0.62604166666666667 0.090625 0.09791666666666667
4 0.1328125 0.64375 0.103125 0.11666666666666667
4 0.559375 0.65520833333333333 0.096875 0.11875
4 0.9359375 0.71145833333333333 0.1 0.15625
8 0.83359375 0.74479166666666666 0.2390625 0.36875
4 0.58046875 0.75625 0.1171875 0.17083333333333334
4 0.63203125 0.86875 0.1390625 0.20833333333333334
```

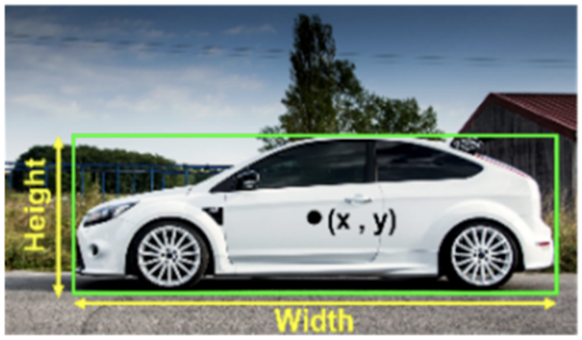


Figure 10: Example of TXT annotation dataset

Unlike the Pascal VOC format, TXT annotation format includes class types and center coordinates of bounding boxes with their widths and heights. Again 50 epochs were executed for the same dataset.

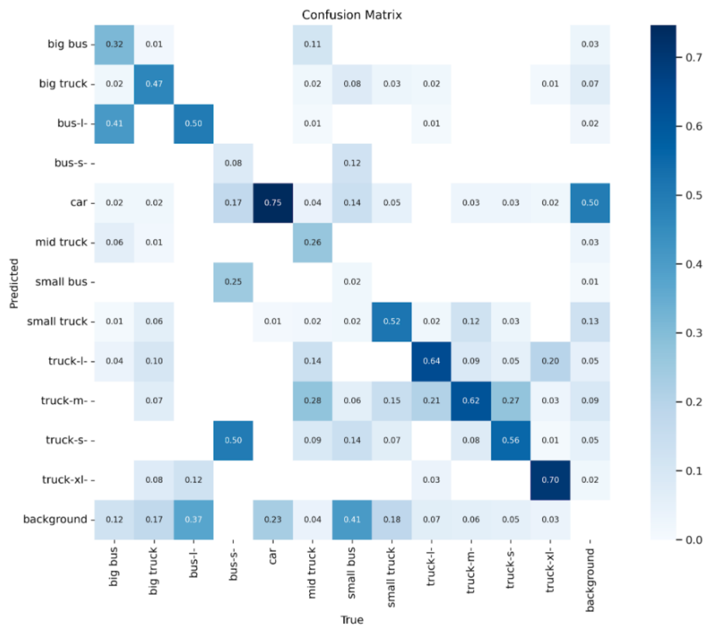


Figure 11: Confusion matrix of YOLOv5 on vehicle dataset

YOLOv5 provided not only training losses, but also various training results such as the confusion matrix in Figure 11. It was able to make correct detections on most of the classes with a classification loss of 0.0069 and box regression of 0.032. Its validation loss was 0.036.

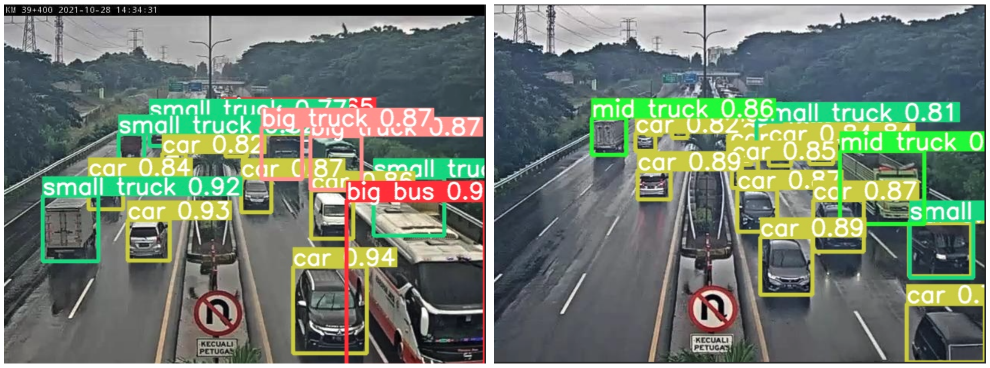
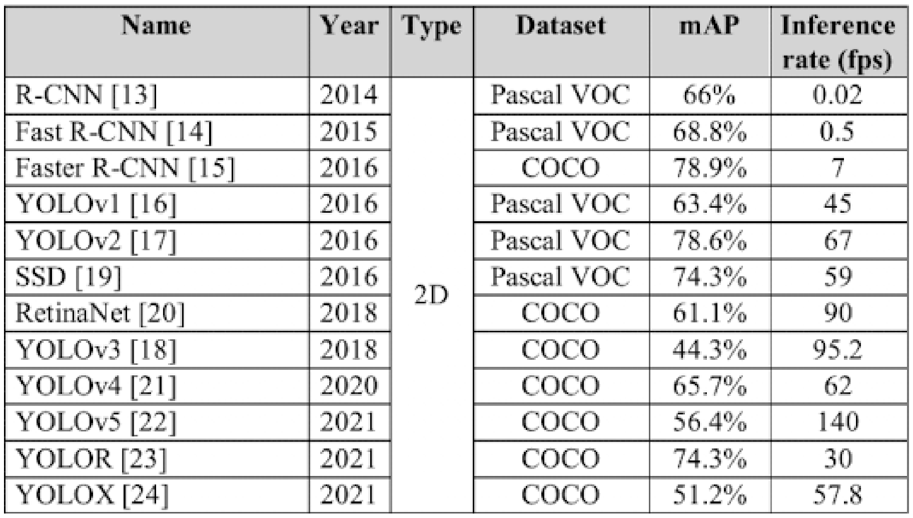


Figure 12: Examples of testing results of YOLOv5

YOLOv5 detected objects very accurately on the testing images as shown in Figure 12. Most of the vehicles were classified correctly and bounding boxes were fitting in them tightly. Even some vehicles that were partly covered by other vehicles were detected showing the powerful inference of the detector.

Detectors Evaluation

Given the limited time and computing resources, only two algorithms were tested. In order to provide an extensive performance evaluation for object detectors in AV, a test result from a paper is borrowed as shown in Figure 13 below.



Name	Year	Type	Dataset	mAP	Inference rate (fps)
R-CNN [13]	2014	2D	Pascal VOC	66%	0.02
Fast R-CNN [14]	2015		Pascal VOC	68.8%	0.5
Faster R-CNN [15]	2016		COCO	78.9%	7
YOLOv1 [16]	2016		Pascal VOC	63.4%	45
YOLOv2 [17]	2016		Pascal VOC	78.6%	67
SSD [19]	2016		Pascal VOC	74.3%	59
RetinaNet [20]	2018		COCO	61.1%	90
YOLOv3 [18]	2018		COCO	44.3%	95.2
YOLOv4 [21]	2020		COCO	65.7%	62
YOLOv5 [22]	2021		COCO	56.4%	140
YOLOR [23]	2021		COCO	74.3%	30
YOLOX [24]	2021		COCO	51.2%	57.8

Figure 13: Object detector test results on vehicle dataset [13]

The main performance indicators are mAP (mean Average Precision) and inference rate. mAP is a commonly used metric as it combines the concepts of precision and recall to assess the accuracy and effectiveness of an object detector. It measures how well the detector localizes and classifies objects across different classes and different levels of confidence thresholds. A higher mAP indicates better object detection performance. Inference rate is literally how many images a detector can process in a second. Reasonably, the two-stage detectors showed good mAP results but very low inference rates compared to those of the single-stage detectors. With the advantage of achieving fast inference rate, the single-stage detectors have made great improvements on accuracy as well and the result shows that mAPs of the single-stage detectors are fairly as high as those of the two-stage detectors.

Gesture Recognition

Gesture Recognition Overview

A gesture can be defined as any meaningful movement of part of the body. Gestures can be performed by the entire body when jumping up and down for joy or they can be more acute and be limited to just two fingers when making a peace sign. Gestures can also be expressed on one's face to demonstrate an emotional response. Not only can gestures be categorized by the region of the body, but they also can be classified as static or dynamic. For example, a static gesture could be a thumbs up or a dynamic gesture could be a flick of the hand to the right.

When considering gestures as they relate to computer vision, the topic of gesture recognition arises: the idea that a meaningful movement can be understood via an image or video. It is useful to have computer techniques to interpret gestures for a variety of reasons. For instance, a photo application could interpret a thumbs up as a cue to take a picture, allowing users to position themselves beyond the reach of a device to capture an image. Or in the setting of a vehicle, having gesture recognition on a flick of the hand to the right could skip the song playing on the sound system without requiring the user to divert their attention from the road to press a button. As a whole, gesture recognition can be applied as an input mechanism to a computer system. Additionally, gesture recognition has use cases in medical operations as alternatives to peripheral devices like a mouse and keyboard when zooming or rotating medical scans and images [21]. Gesture interaction can also be used for gaming, as popularized by the Microsoft Kinect Xbox, through hand or body movement.

VR/AR Gesture Recognition

One of the applications of gesture recognition that has gained popularity with recent technological advances is with Virtual Reality (VR) or Augmented Reality (AR) devices. VR is a "computer simulation system that can create and simulate virtual worlds" [22] that provides an immersive experience for users, fully occluding the user from viewing the real world. Instead, users are provided an experience in which their interactions are given feedback through a virtual scene, as if they are immersed. Augmented reality follows a similar principle but instead of fully immersing users in a virtual world, virtual elements are augmented onto a real-world scene. This real-world scene can either be truly "seen" through a transparent medium like glass with virtual elements displayed on the glass, or the real-world can be transmitted to the user on a screen through a camera with the virtual elements displayed on the screen as well.

Devices like the Meta (Oculus) Quest specialize primarily in the VR space but also have AR (passthrough) capabilities for a relatively low cost. The Quest headset comes with handheld controllers but also can be controlled via gesture recognition with hands. Other devices like Apple's Vision Pro are designed primarily for AR and are primarily controlled with gesture recognition. Both of these devices are head-mounted displays (HMDs), meaning the user wears the device on their head and their visual experience comes from the screen covering their eyes. Among other sensors and other proprietary differences, both of these headsets are equipped with cameras on the underside of the front of the device, positioned so that when the user's head is looking forward, the cameras can capture the user's hands. The distinction between VR and AR-specific headsets on the topic of gesture recognition is not incredibly important since the distinction is primarily concerned with the visuals transmitted to the user. The main observation from these HMDs is that they use cameras to capture hand gestures performed by the user from which the device can use computer vision to interpret and process these gestures.

Major State of the Art Algorithms

Gesture recognition, especially as it applies to VR/AR devices, poses some unique challenges within computer vision. For real-time gesture recognition from video streams some challenges include identifying the actual start and end of a performed gesture and ensuring that the performed gesture is only recognized once [17]. Additionally, it's essential to have gesture recognition algorithms that are highly efficient and responsive. With HMDs that occlude the user from viewing the real world, any latency or lag in displaying gesture responses can be frustrating or even disorienting for users. To tackle some of these problems, several advances in efficient and accurate gesture recognition algorithms have surfaced, including Support Vector Machines, Random Forests, and K-Nearest Neighbors.

Support Vector Machines

A popular approach to gesture recognition algorithms is the Support Vector Machine (SVM). SVM is a supervised learning algorithm used in classification or regression problems [17]. The premise of this algorithm is to find a multi-dimensional hyperplane separating classes. Many superplanes exist that can divide a dataset, but the objective of SVM is to find the hyperplane that maximizes the margin between classes. This margin is defined by the nearest data points of different classes to the plane. For non-linear classification, kernels can also be applied.

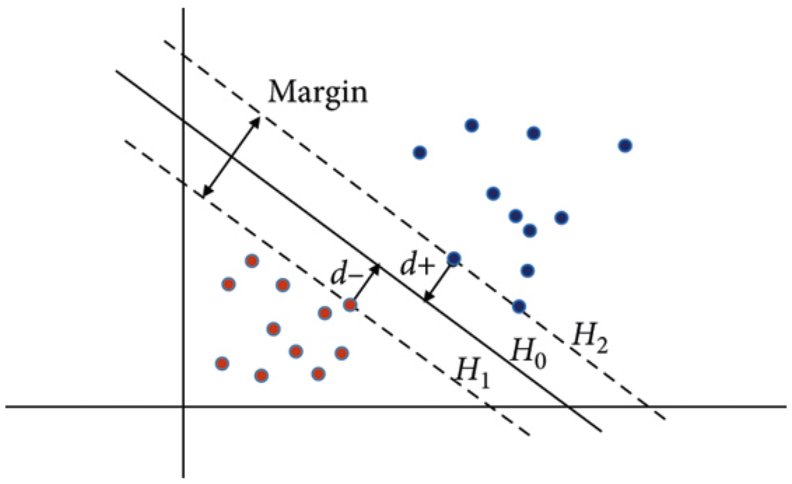


Figure 14: An example of a hyperplane maximizing the distance, d, between two classes [17]

Random Forests

Random Forests (RF) are another popular gesture recognition algorithm. RF is an ensemble learning algorithm combining multiple decision trees, leading to a high precision statistical modeling technique for regression and classification [17]. Naive decision trees are generally prone to overfitting and inaccuracy in high-dimensional modeling problems, but RF can correct these problems. The algorithm creates a specified number of trees, each trained on a different subset of the training data. Each tree assigns a class based on their own model, and the final prediction produced by RF aggregates the predictions from the trees.

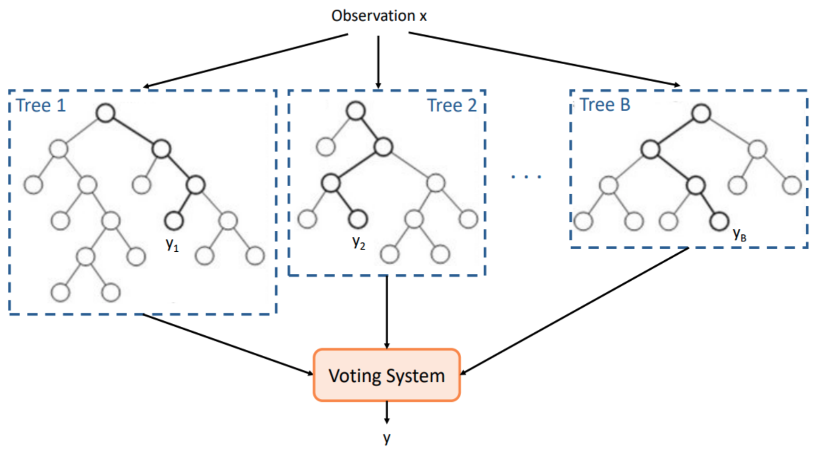


Figure 15: A visualization of the Random Forest algorithm [17]

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is also an effective algorithm for gesture recognition. Unlike the aforementioned algorithms that have a training phase, this algorithm simply stores the training dataset and finds the K nearest neighbors to a given data point. The nearest neighbors are determined by calculating the distance between feature vectors. The number of nearest neighbors, K, is a parameter that can be specified. The final prediction is the class label appearing most frequently among the nearest neighbors. It's a simple and easy algorithm to use and is strong with small datasets with a well-defined feature space [20].

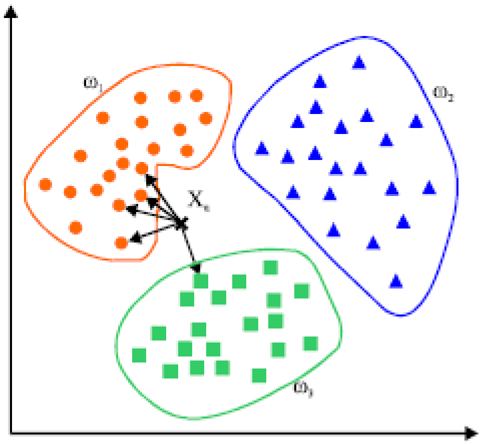


Figure 16: A visual representation of KNN for a given point, X, with three classes [20]

Standard Test Dataset

In order to evaluate these proposed algorithms, a standard test dataset was used. A multitude of labeled gesture recognition datasets exist but the uniqueness of analyzing gesture recognition for VR/AR applications requires an egocentric dataset, where the user's head is behind the camera and the captured feed is either first-person or pointed downwards.

The dataset used is the Virtual Reality Gesture Recognition Dataset [21]. The dataset uses snapshots of individuals making different hand gestures, and the features are reported as specific distances between the fingers and the palm, including distances between two fingers.

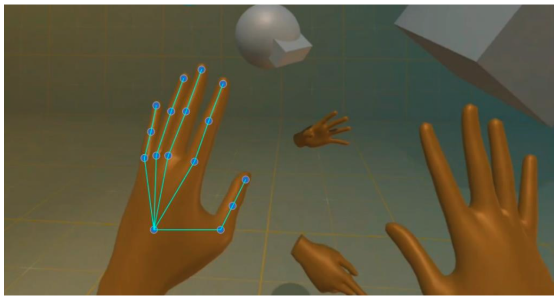


Figure 17: An image from the Virtual Reality Gesture Recognition Dataset [21]

The dataset consists of 9 features, which are the distances, and 6 classes representing various gestures. The overall dataset includes 447 samples, and the Virtual Reality Gesture Recognition Dataset separates the samples into 3 datasets based on the probabilities listed in the figure below.

Dataset	Gesture 1	Gesture 2	Gesture 3	Gesture 4	Gesture 5	Gesture 6
Dataset 1	70%	40%	20%	30%	40%	30%
Dataset 2	20%	30%	20%	60%	20%	30%
Dataset 3	10%	30%	40%	10%	40%	40%

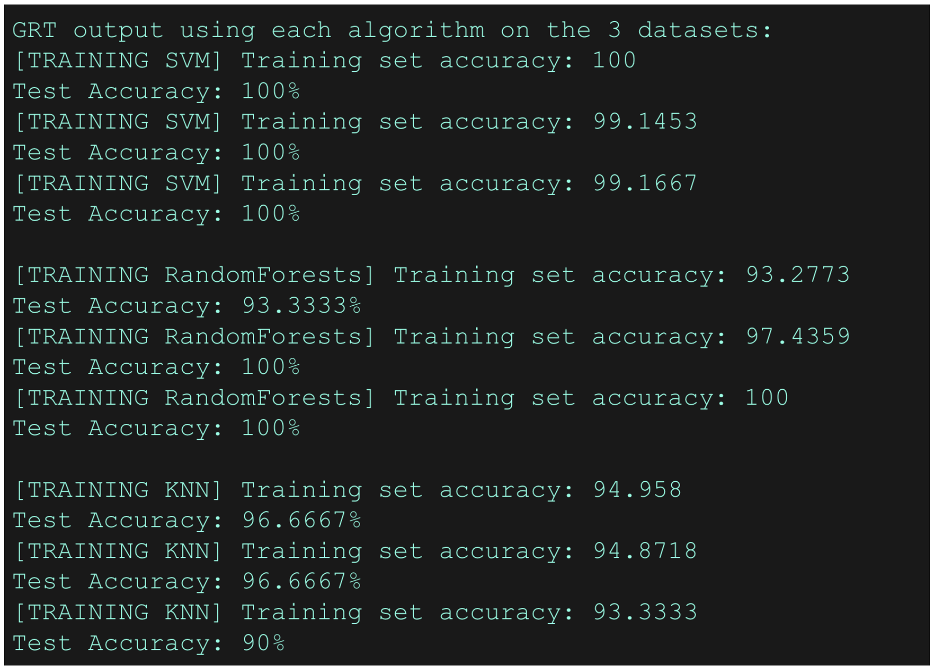
Figure 18: The data distribution of the 3 smaller datasets used in the Virtual Reality Gesture Recognition Dataset [21]

For the Evaluation of Algorithms section, only this dataset was used in the evaluation process. However, it is worth mentioning another dataset, given more time and resources to continue researching this topic, would be used. The EgoGesture Dataset is a significantly larger dataset that is described as a “multi-modal large scale dataset for egocentric hand gesture recognition” [22]. It contains 2,081 RGB-D videos, 24,161 gesture samples, and 2,953,224 frames from 50 distinct subjects with 83 classes of static and dynamic gestures. The advantages of using this dataset over the previously mentioned is clear since it is a much more diverse and comprehensive dataset. However, the sheer size of the dataset, consisting of 46 GB of videos and 32 GB of images, proved to be a challenge to download on the computer used for running the algorithms. If given access to a computer with more storage capacity to handle the dataset and higher processing power to speed up the unarchiving process, this dataset would be the clear choice. Unfortunately, the limitations of storage and processing power led the Virtual Reality Gesture Recognition Dataset to be the dataset used in evaluation.

Evaluation of Algorithms

In order to evaluate the SVM, RF, and KNN algorithms, the Gesture Recognition Toolkit (GRT) was used. GRT is a cross-platform, open-source, C++ machine learning library specifically designed for real-time gesture recognition. It has a multitude of tools available including classification algorithms, clustering algorithms, and regression modules, to name a few. It's organized into an object-oriented modular architecture, which supports functions like predict, train, save, load, reset, and clear for nearly all of the GRT classes. To evaluate the algorithms, the SVM, RandomForests, and KNN classes from this toolkit were used.

The overall process for evaluating the algorithms was relatively standardized, having small deviations to tune parameters of the individual algorithms. Each program took in an argument to specify the filepath of the dataset, which would be to one of the three CSVs from the Virtual Reality Gesture Recognition Dataset. The dataset would then be loaded into a ClassificationData object, used for the training data. The toolkit also provided a method for the ClassificationData class to split the dataset, and so for each program 20% of the training dataset was reserved for testing. Then within each algorithm's evaluation program, an object was created for the classifier and parameters were set. For SVM, the classifier was initialized as a linear kernel and set to enable scaling of the training and prediction data. For RF, the classifier was set to have a forest size of 10, a maximum tree depth of 10, and minimum number of samples per node of 10. For KNN, the K value was initialized to 6 and scaling was also enabled. Once the classifier objects were instantiated, the classifier was trained on the training data and the model was saved to a file. Using the generated model, the predict method was used on the classifier, given an input feature vector from the test dataset. The accuracy of the predicted versus actual class was determined using the test dataset and reported. For each of the classification algorithms, the training set accuracy was reported along with the resulting test accuracy for the 3 datasets. The results are found in the figure below.



```
GRT output using each algorithm on the 3 datasets:
[TRAINING SVM] Training set accuracy: 100
Test Accuracy: 100%
[TRAINING SVM] Training set accuracy: 99.1453
Test Accuracy: 100%
[TRAINING SVM] Training set accuracy: 99.1667
Test Accuracy: 100%

[TRAINING RandomForests] Training set accuracy: 93.2773
Test Accuracy: 93.3333%
[TRAINING RandomForests] Training set accuracy: 97.4359
Test Accuracy: 100%
[TRAINING RandomForests] Training set accuracy: 100
Test Accuracy: 100%

[TRAINING KNN] Training set accuracy: 94.958
Test Accuracy: 96.6667%
[TRAINING KNN] Training set accuracy: 94.8718
Test Accuracy: 96.6667%
[TRAINING KNN] Training set accuracy: 93.3333
Test Accuracy: 90%
```

Figure 19: Results using the GRT SVM, RF, and KNN classifiers on the 3 datasets from the Virtual Reality Gesture Recognition Dataset

The results were determined by the percentage of correctly predicted class labels over the total number of samples in the test dataset. There are other metrics to evaluate classifier performance but this approach is a simple and a good indicator. From the above results, the SVM classifier performed the strongest. This is likely due to the nature of SVM having strength in handling high-dimensional feature spaces. The success of these algorithms on this dataset could be attributed to a number of factors, including the size of the dataset in terms of number of samples, features, classes, etc. If the hardware limitations were resolved, evaluating these classifiers on the much larger EgoGesture dataset would be a worthwhile exercise to truly evaluate the performance of these algorithms.

Other Algorithm Evaluations

Other researchers have done more through evaluations of the aforementioned algorithms and also different promising algorithms. For instance, the SVM algorithm incorporated in histogram of oriented gradient (HOG) was demonstrated to be a strong gesture recognition algorithm in its application to robotics systems, with accuracy improved up to 99% [23]. This proposed algorithm, though thorough, was still evaluated on a small dataset and should be evaluated on a larger, more comprehensive dataset. Another algorithmic approach to tackling gesture recognition is using convolutional neural networks (CNNs). Researchers developed an architecture consisting of two models: the first which is a lightweight CNN gesture detector, and the second which is a deep CNN to classify the detected gestures [21]. This model was used as a classifier on the EgoGesture and NVIDIA Gesture datasets, achieving 94.04% and 83.82%

accuracy, respectively, on these large datasets, proving to be another effective algorithmic approach.

Challenges

Numerous challenges persist as advancements are made to gesture recognition in VR/AR devices. The first is developing efficient and lightweight algorithms. Many older HMDs were tethered to a computer, with much of the processing power on that external computer. However, this age of HMDs is shifting towards having the computer onboard. To maximize comfort and wearability, these headsets need to be lightweight and should not have spatially large compute units. A specific challenge to gesture recognition is the interpretation of both static and dynamic gestures, and being able to interpret gestures that appear similar. These distinctions can be addressed by having more training data but can otherwise be avoided by designers by assigning distinct, yet intuitive gestures for tasks.

Other computer vision challenges that apply to gesture recognition with VR/AR are a camera's limited field of view (FOV) and lighting limitations. Only what the camera sees can be processed, and so gestures performed outside of the camera's FOV, including out of frame or blocked by other objects, might not get captured. Additionally, gestures performed in dark lighting prove to be difficult to interpret. Both of these challenges are often mitigated by using more sensors on the HMDs, either by positioning more cameras on the display to extend FOV or by using other types of sensors like LIDAR.

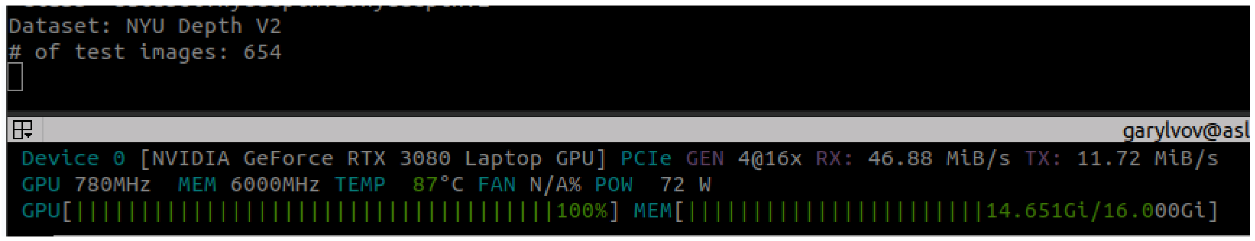
Monocular Depth Estimation

Motivation

Although it is possible to obtain depth information from stereo cameras with classical techniques, a monocular approach could reduce the price of equipment needed by only relying on one camera as opposed to several. Even systems that employ multiple cameras could be augmented with the additional information. Furthermore, most cameras are monocular, as well as most video/photos that are available, due to the popularity of monocular cameras. The human ability to perceive a monocular video or image as realistic implies the human ability to perform monocular depth estimation. If one were to try to navigate a scene with one eye closed, they would be able to perform some depth estimation, albeit perhaps not as well as with two eyes open. As computer vision attempts to mimic a human's ability to perceive, this predicates monocular depth estimation as a fundamental problem to computer vision. Collecting labeled datasets for monocular depth estimation is convenient as they can be autonomously labeled by recording point cloud data from a stereo system or LiDAR concurrently with a monocular feed. However, monocular depth estimation presents a significant technical challenge. Unlike stereo vision, which calculates depth through triangulation from two different viewpoints, monocular depth estimation must rely on other cues. These cues may include perspective (the size of known objects), texture gradient (the detail of textures), or shading, among others. The ability to algorithmically interpret these cues in a manner similar to human perception is a complex task and forms the crux of the problem.

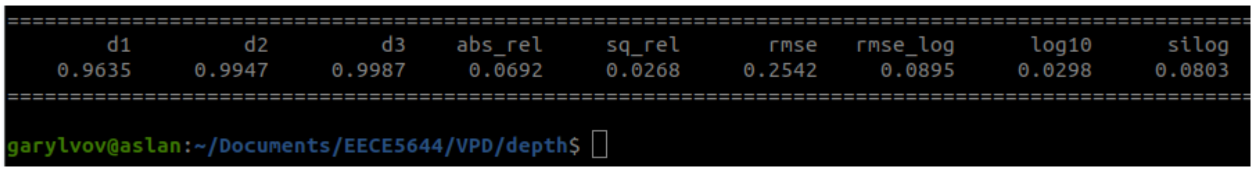
Evaluation

Currently, the two most popular datasets for monocular depth estimation are the KITTI Eigen split [24], and NYU-Depth V2 [25]. The KITTI dataset includes scenes outdoors on roads, and the NYU-Depth V2 dataset includes indoor scenes. As benchmarked by Papers with Code, SwinV2-L 1K-MIM[20] exhibits the lowest error on the KITTI Eigen split, and VPD [17] exhibits the lowest error on NYU-Depth V2. These techniques are selected for further investigation due to their performance, and their principles of these approaches are described in subsequent sections. Running these models proved difficult on an Nvidia GeForce RTX 3080 Laptop GPU with 16gb VRAM, due to their high computational demands. The VPD repository suggests utilizing 8 Nvidia V100 GPUs, which could cost over a hundred thousand dollars.



```
Dataset: NYU Depth V2
# of test images: 654

                                                                           garylvov@asl
 Device 0 [NVIDIA GeForce RTX 3080 Laptop GPU] PCIe GEN 4@16x RX: 46.88 MiB/s TX: 11.72 MiB/s
 GPU 780MHz  MEM 6000MHz TEMP  87°C FAN N/A% POW  72 W
 GPU[||||||||||||||||||||||||||||||||||||||100%] MEM[||||||||||||||||||||||14.651Gi/16.000Gi]
```



```
==============================================================================================
       d1         d2         d3     abs_rel      sq_rel       rmse   rmse_log      log10      silog
   0.9635     0.9947     0.9987     0.0692      0.0268     0.2542     0.0895     0.0298     0.0803
==============================================================================================

garylvov@aslan:~/Documents/EECE5644/VPD/depth$
```

Figure 20: GPU consumption by VPD on NYU depth v2, and the reported errors output of the testing set.

This gives the insight that it is unlikely for such models to run locally on more inexpensive consumer hardware (as it took over an hour to estimate depth for 654 images on a higher specification gaming laptop) until GPU processing power improves, or such models can be produced in more lightweight variants.

On the NYU Depth V2 dataset, VPD had an RMSE of .2542, while SwinV2-L 1K-MIM had an RMSE of .2871. It is important to note that despite SwinV2-L 1K-MIM's higher RMSE, it does not require context labels like VPD. As a result of requiring context labels, VPD is not yet compatible with KITTI, where SwinV2-L 1K-MIM had an RMSE of 1.966.

Vision Transformer

Vision Transformers (ViT) were first introduced in the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [27]. This groundbreaking model offers a unique approach to image recognition tasks, by dividing images into non-overlapping patches and treating these patches as if they were words in a sentence - a process equivalent to the attention layer used in the influential NLP paper, "Attention Is All You Need" [22]. This allows the model to determine the importance or attention of different parts of the image. In other words, 'attention' allows the model to focus more computational resources on important parts of the image, while reducing the emphasis on less relevant areas. This is especially beneficial in tasks like depth estimation, where the importance of different image regions varies greatly. By focusing on regions with higher relevance for depth cues, the model can achieve more accurate depth estimation. Recently, this attention-based approach has shown immense promise in the field of monocular depth estimation, as it is adept at recognizing patterns and correlations between different elements of the image. This sophisticated level of understanding allows Vision Transformers to accurately estimate depth from a single image, pushing the boundaries of what is possible in monocular depth estimation.

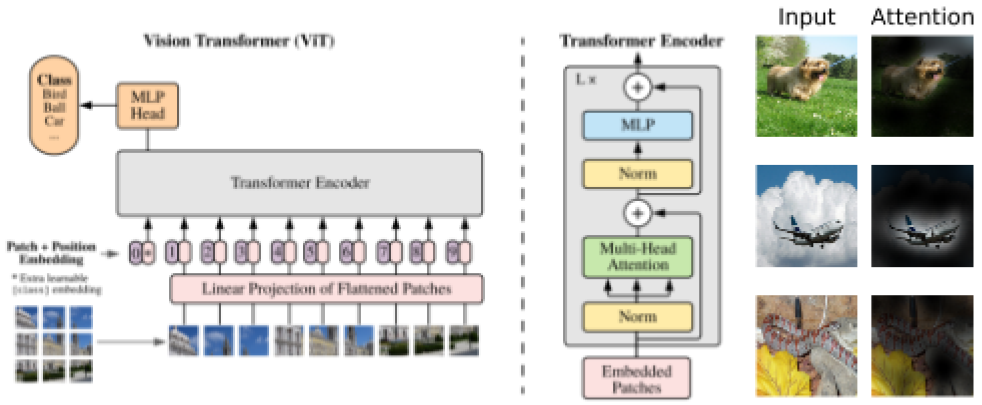


Figure 21: Visual Transformer architecture and visualization of Attention [28].

SwinV2-L 1K-MIM

SwinV2-L 1K-MIM is a cutting-edge approach to depth estimation introduced in the paper "Revealing the Dark Secrets of Masked Image Modeling" [26]. It boasts the lowest Root Mean Square Error (RMSE) in the Monocular Depth Estimation on KITTI Eigen split, and ranks fifth lowest on the NYU-Depth V2 benchmark as benchmarked by Papers With Code. This approach leverages the concept of Masked Image Modeling (MIM), which involves the use of images with omitted portions for training transformers, a strategy that is also applicable for object detection and segmentation. MIM involves the use of images with deliberately omitted, or "masked," portions. These masked portions provide the model with a kind of 'blind spot' during training. The task of the model is then to predict and fill in these masked sections, much like filling in the blanks in a sentence. This method significantly enhances the model's depth estimation capabilities by forcing it to infer information from the visible sections of an image to accurately predict the obscured parts. The model, in essence, learns to understand and replicate the inherent structural and contextual patterns within the image.

The model specifically uses the Swin Transformer block, where the standard multi-head self-attention mechanism is replaced with a shifted windows module. This innovative method has shown improved performance in various computer vision tasks. Unlike traditional vision transformers which typically input an RGB image and output a class label, SwinV2-L 1K-MIM operates differently. It takes an RGB image with a missing patch as its input and instead of a class label, the model outputs a depth map. A depth map is a two-dimensional representation that encapsulates the three-dimensional structure of the scene. Each pixel in the depth map represents the estimated distance from the camera to the corresponding point in the real world, allowing for a detailed understanding of the scene's spatial properties. This advanced application of vision transformers sets a new bar in monocular depth estimation.

VPD

In the study "Unleashing Text-to-Image Diffusion Models for Visual Perception", a novel framework known as Visual Perception with a Pre-trained Diffusion model (VPD) is proposed. This framework exploits the high-level knowledge embedded within a pre-trained text-to-image diffusion model, specifically aiming to harness this knowledge for downstream visual perception tasks such as depth estimation. Diffusion models, trained on extensive image-text pairs, carry substantial high-level information obtained from natural language supervision during pre-training. They are capable of reconstructing the data distribution by learning the reverse process of a diffusion process. VPD goes a step further by "prompting" the model with appropriate textual inputs and refining these text features with an adapter. This strategy results in better alignment to the pre-trained stage and promotes a productive interaction between the visual contents and text prompts.

A key aspect of the VPD framework involves the construction of hierarchical feature maps using the text-to-image diffusion model. These maps are derived from an image or conditioning inputs and then decoded for the specific downstream perception task—in this case, depth estimation. By establishing a connection between the task-specific label and natural language, the learned semantic information can be efficiently extracted and utilized. The effectiveness of the VPD framework, specifically for depth estimation, is demonstrated through impressive results. Notably, VPD outperforms methods that use robust visual backbones pre-trained on masked image modeling such as SwinV2-L 1K-MIM on some datasets, establishing a new record in the field. These findings underline that large-scale text-to-image pre-training can be highly competitive in downstream visual perception tasks, even compared to dedicated visual pre-training methods, reinforcing the efficacy of the VPD approach.

Discussion

Monocular depth estimation, though a complex problem, is fundamental to advancing the capabilities of computer vision systems. Its potential applications are far-reaching, from enabling cost-effective autonomous navigation to enhancing virtual reality experiences. The discussed techniques, SwinV2-L 1K-MIM and VPD, represent promising approaches in this field, harnessing the power of advanced techniques like Vision Transformers and pre-trained diffusion models. As GPU processing power continues to improve and more lightweight models emerge, we can expect these technologies to become more accessible and widely used. However, significant challenges remain, such as the high computational demand of these models, indicating a path for future research and development.

Challenges in Computer Vision

Computer vision, as a research area, brings forth numerous challenges that test the capabilities of existing machine learning techniques and technologies. Real-time image processing with machine learning stands as a significant hurdle. It requires handling high-dimensional data instantaneously and efficiently, which demands substantial computational power and intricate algorithms.

Training deeper models introduces another complication, needing copious amounts of accurately labeled data. Gathering and labeling such large datasets, particularly for complex images, is labor-intensive, time-consuming, and costly. Environmental variability also adds to the difficulty. Photos taken under different lighting, from varying angles, and in diverse settings introduce inconsistencies that can confound models, affecting their ability to interpret and analyze images correctly. Moreover, understanding context in an image is a daunting task. It's not just about recognizing the objects within the image; it's also about understanding their interrelationships, the roles they play within the scene, and the overall narrative of the image. Additional challenges lie in handling occlusions, changes in object appearance, scale variation, and 3D perception. Furthermore, developing systems that can learn continually without human intervention and making sense of dynamic and rapidly changing scenes also pose significant challenges.

References

[1] C. Harris, M. Stephens, and others, “A combined corner and edge detector,” in Alvey vision conference, 1988, vol. 15, no. 50, pp. 10–5244.
[2] D. Mistry and A. Banerjee, “Comparison of feature detection and matching approaches: SIFT and SURF,” GRD Journals-Global Research and Development Journal for Engineering, vol. 2, no. 4, pp. 7–13, 2017.
[3] J. Canny, “A computational approach to edge detection,” IEEE Transactions on pattern analysis and machine intelligence, no. 6, pp. 679–698, 1986
[4] B. K. Horn and B. G. Schunck, “Determining optical flow,” Artificial intelligence, vol. 17, no. 1–3, pp. 185–203, 1981.
[5] R. A. Hamzah and H. Ibrahim, “Literature survey on stereo vision disparity map algorithms,” Journal of Sensors, vol. 2016, 2016.
[6] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” IEEE robotics & automation magazine, vol. 13, no. 2, pp. 99–110, 2006.
[7] M. Z. Alom et al., “The history began from alexnet: A comprehensive survey on deep learning approaches,” arXiv preprint arXiv:1803.01164, 2018.
[8] L. R. Medsker and L. Jain, “Recurrent neural networks,” Design and Applications, vol. 5, pp. 64–67, 2001.
[9] B. Wu et al., “Visual transformers: Token-based image representation and processing for computer vision,” arXiv preprint arXiv:2006.03677, 2020.
[10] R. Shanmugamani, “Deep Learning for Computer Vision,” Packt Publishing, 2018
[11] Y. Wang et al., “A comprehensive review of modern object segmentation approaches,” arXiv preprint arXiv:2301.07499v1
[12] R. Sagar, “How the deep learning approach for object detection evolved over the years,” Analyticsindiamag.com, https://analyticsindiamag.com/how-the-deep-learning-approach-for-object-detection-evolved-ove r-the-years/ (accessed Jun. 20, 2023)
[13] A. Balasubramaniam and S. Pasricha, “Object Detection in Autonomous Vehicles: Status and Open Challenges,” arXiv preprint arXiv:2201.07706
[14] A. Karaka, “Object Detection with RetinaNet,” wanda.ai, https://wandb.ai/site/articles/object-detection-with-retinanet (accessed Jun. 22, 2023)
[15] M. Oudah, A. Al-Naji, and J. Chahl, “Hand gesture recognition based on Computer Vision: A review of techniques,” Journal of Imaging, vol. 6, no. 8, p. 73, 2020. doi:10.3390/jimaging6080073
[16] Y. LI, J. HUANG, F. TIAN, H.-A. WANG, and G.-Z. DAI, “Gesture interaction in virtual reality,” Virtual Reality &amp;amp; Intelligent Hardware, vol. 1, no. 1, pp. 84–112, 2019. doi:10.3724/sp.j.2096-5796.2018.0006
[17] O. Kopuklu, A. Gunduz, N. Kose, and G. Rigoll, “Real-time hand gesture detection and classification using convolutional Neural Networks,” 2019 14th IEEE International Conference

on Automatic Face &amp;amp; Gesture Recognition (FG 2019), 2019. doi:10.1109/fg.2019.8756576

[18] P. N. Huu and T. Phung Ngoc, “Hand gesture recognition algorithm using SVM and Hog Model for control of Robotic System,” Journal of Robotics, vol. 2021, pp. 1–13, 2021. doi:10.1155/2021/3986497

[19] N. Bargellesi, M. Carletti, A. Cenedese, G. A. Susto, and M. Terzi, “A random forest-based approach for hand gesture recognition with wireless wearable motion capture sensors,” IFAC-PapersOnLine, vol. 52, no. 11, pp. 128–133, 2019. doi:10.1016/j.ifacol.2019.09.129

[20] M. Z. Alksasbeh et al., “Smart hand gestures recognition using k-NN based algorithm for video annotation purposes,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 21, no. 1, p. 242, 2021. doi:10.11591/ijeecs.v21.i1.pp242-252

[21] Dag Eklund, Ilias Siniosoglou, Anna Triantafyllou, Athanasios Liatifis, Dimitrios Pliatsios, Thomas Lagkas, Vasileios Argyriou, Panagiotis Sarigiannidis, May 30, 2023, "Virtual Reality Gesture Recognition Dataset", IEEE Dataport, doi: https://dx.doi.org/10.21227/kyzx-m451

[22] Y. Zhang, C. Cao, J. Cheng and H. Lu, "EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition," IEEE Transactions on Multimedia (T-MM), Vol. 20, No. 5, pp. 1038-1050, 2018

[23] W. Zhao, Y. Rao, Z. Liu, B. Liu, J. Zhou, and J. Lu, “Unleashing text-to-image diffusion models for visual perception,” arXiv preprint arXiv:2303.02153, 2023.

[24] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in 2012 IEEE conference on computer vision and pattern recognition, 2012, pp. 3354–3361.

[25] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgbd images.,” ECCV (5), vol. 7576, pp. 746–760, 2012.

[26] Z. Xie, Z. Geng, J. Hu, Z. Zhang, H. Hu, and Y. Cao, “Revealing the dark secrets of masked image modeling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 14475–14485.

[27] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” arXiv preprint arXiv:2010.11929, 2020.

[28] A. Vaswani et al., “Attention is all you need,” Advances in neural information processing systems, vol. 30, 2017.