Universitatea Tehnică "Gheorghe Asachi", Iași
Facultatea de Automatică și Calculatoare

Project 1: Lane Detection

The Intro to Computer Vision labs will be run in Google Colaboratory, a Jupyter notebook environment that runs entirely in the cloud, you don't need to download anything. To run these labs, you must have a Google account.

Step 1: click on the assignment invite link -> **Accept this assignment**. Refresh page -> individual repo for the specific assignment is created automatically

Project 1: 3 weeks
https://classroom.github.com/a/iM-Cnmc8

Step 2: Navigate to http://colab.research.google.com/github -> Click the **Include Private Repos** checkbox -> **select the correct repo** (SistemeDeVedereArtificiala/assignment_name-student_name) -> Click on the jupyter notebook of the current assignment

Step 3: [GitHub sign-in window] In the popup window, sign-in to your Github account and authorize Colab to read the private files.

Step 4: [in colab] **File** -> **Save a copy to GitHub**. Select the correct repository for the SPECIFIC assignment -> Click the **Include Colab Link** -> Click **OK**

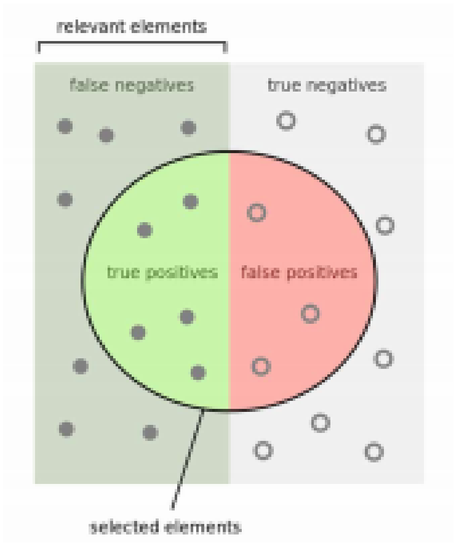
Step 5: [in colab] Navigate to the **Runtime** tab --> **Change runtime type**, under **Hardware accelerator** select **GPU/TPU** (tensor processing unit) according to your needs.

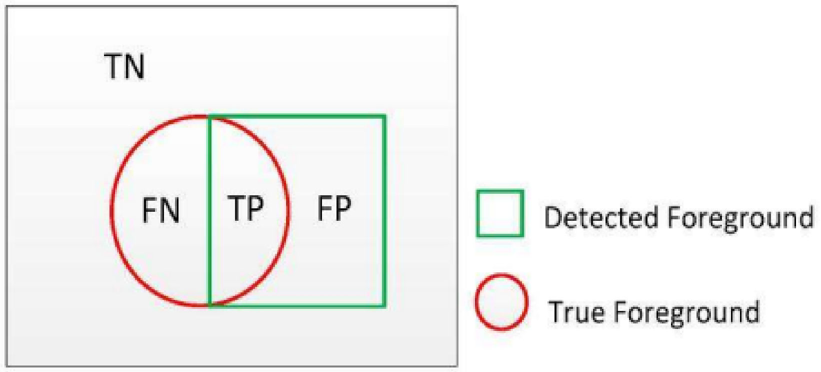
Read the suggestions and accomplish all tasks marked with **#TODO**.

!!! At the end of each laboratory **REPEAT step 4 in order to SAVE** the answers to your private repository (individual for each assignment)

Week 3: evaluate performance of our solution

Evaluation metrics:
When trying out different segmentation methods, how do you know which one is best? If you have a ground truth or gold standard segmentation, you can use various metrics to check how close each automated method comes to the truth.



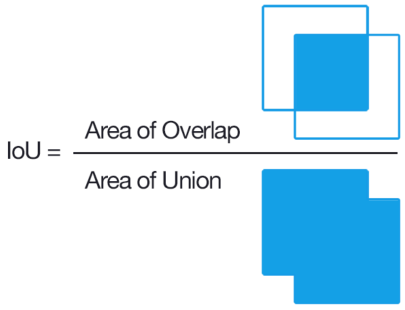


TP is the number of true positives
TN is the number of true negatives
FN is the number of false negatives
FP is the number of false positives

1. **Pixel accuracy -** the percent of pixels in the image that are classified correctly

$$Pixel_{accuracy} = \frac{\sum_i n_{ii}}{\sum_i t_i} = \frac{TP+TN}{TP+TN+FP+FN}$$

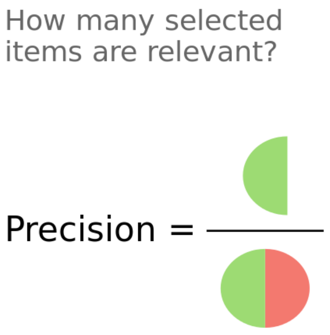
2. **Mean IoU (Mean Intersection over Union):** This metric ranges from 0–1 (0–100%) with 0 signifying no overlap and 1 signifying perfectly overlapping segmentation.



$$IoU = \frac{1}{n_{cl}} \bullet \frac{\sum_i n_{ii}}{t_i + \sum_j n_{ij} - n_{ii}} = \frac{TP}{TP+FN+FP}$$

where: nij – number of pixels of class i predicted to belong to class j, ncl – number of classes, ti – total number of pixels of class i

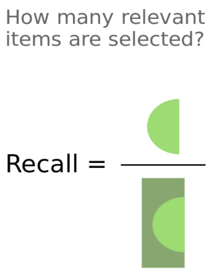
3. **Precision** - a metric of exactness or quality



$$Precision = \frac{TP}{TP+FP}$$

4. **Recall** - a metric of completeness or quantity

$$Recall = \frac{TP}{TP+FN}$$



The scores of precision and recall should be both high, but there is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other.

5. **F-measure**, also known as Sørensen–Dice coefficient or Dice similarity coefficient

$$F - measure = \frac{2*Recall*Precision}{Recall+Precision}$$

The F-Measure which is a harmonic mean of precision and recall can be viewed as a compromise between precision and recall. It balances the precision and recall with equal weights, and it is high only when both recall and precision are high. A higher score of F-Measure means that the performance of the algorithm is better.

Computation of TP, FP, FN, TN in C++ using OpenCV:

```
struct result_t {             int TP;
                              int FP;
                              int FN;
                              int TN; };

 result_t conf_mat_2c(cv::Mat1b truth, cv::Mat1b detections) {  CV_Assert(truth.size ==
                                      detections.size);
       result_t result = { 0 };
       cv::Mat inv_truth(~truth);
       cv::Mat inv_detections(~detections);
       cv::Mat temp;
       cv::bitwise_and(detections, truth, temp);
       result.TP = cv::countNonZero(temp);
       cv::bitwise_and(detections, inv_truth, temp);
       result.FP = cv::countNonZero(temp);
       cv::bitwise_and(inv_detections, truth, temp);
       result.FN = cv::countNonZero(temp);
       cv::bitwise_and(inv_detections, inv_truth, temp);
       result.TN = cv::countNonZero(temp);
       return result;  }
```

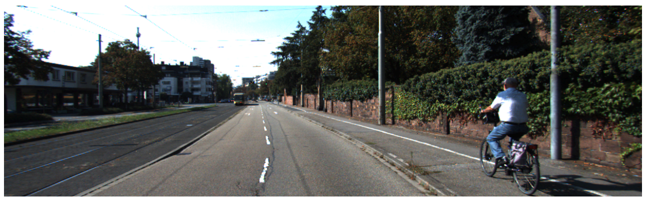
TODO: evaluate performance

Given the Ground Truth (golden model) and our detected lane compute:

1. TP, TN, FN, FP
2. Accuracy, Mean IoU, Precision, Recall, F-measure
   for Kitti Lande Detection dataset annotated images
   (http://www.cvlibs.net/datasets/kitti/eval_road.php)

- images available at
  https://drive.google.com/drive/folders/1yuq8kJaVY6TKRU93ahVbsz-I7aT-Azuy?usp=sharing
- annotation available at
  https://drive.google.com/drive/folders/1fmWB21Nv6yeyPSIFqnU_T6956bpCfk52?usp=sharing

Figure: input image and ground truth (lane in magenta)



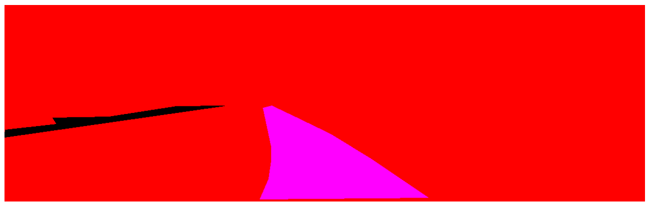


Figure: detected lane using our pipeline

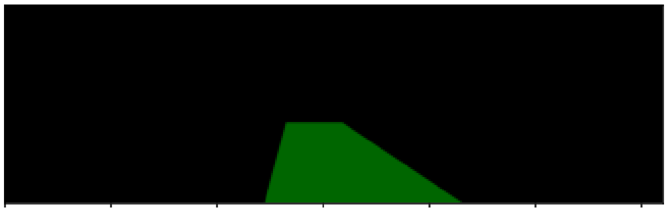
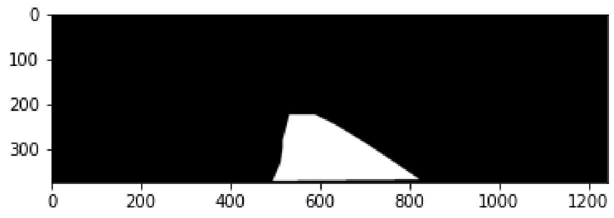
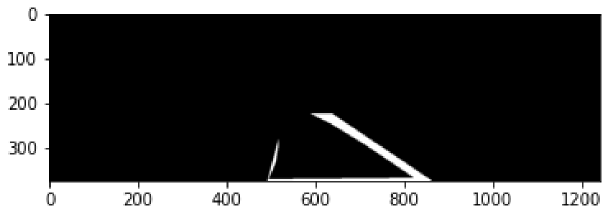
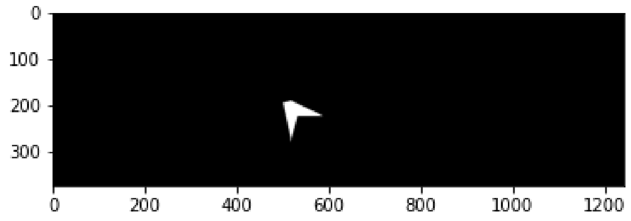
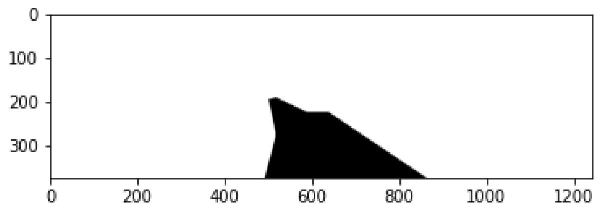


Figure: | True Positives | True Negatives |
| False Negatives | False Positives |


0
100
200
300
0
200
400
600
800
1000
1200


0
100
200
300
0
200
400
600
800
1000
1200


0
100
200
300
0
200
400
600
800
1000
1200


0
100
200
300
0
200
400
600
800
1000
1200