# South Dakota School of Mines and Technology

Deep Learning
Fall 2025
CSC 760
Project 2
Total Points: 200
**Due Date:** November 7, 2025, at 11:59 PM

**1. Objective**

This project belongs to the "Projects" category of assessments, as stated in the syllabus for this course. The objective of this project is to design and implement a computer vision program that detects and tracks the state of a train and the corresponding status of a traffic signal from a given video. This project provides practice in applying the algorithms and concepts learned in class, including YOLO for object detection and state machines implemented with ORB. The project also provides hands-on experience in analyzing dynamic visual scenes, detecting and tracking objects, recognizing changes in motion, and identifying events that occur over time based on visual information extracted from continuous frames. The program to be developed for this project will analyze a video where a traffic light and a train appear within the same frame. The program must determine and output the color of the traffic signal (red, yellow, or green) and detect and classify the train's state as not_present, just_entered, moving, or left_the_frame. Each of these states corresponds to a specific condition that must be identified from the video: when the train is not visible in the frame, when the train first appears, when it is in motion within the frame, and when it has completely exited the frame. The project provides an opportunity to apply computer vision techniques for the purpose of developing an intelligent monitoring system that can automatically identify critical states of objects in real time. Through this project, the application of YOLO demonstrates the modern deep learning approach to object detection, while the ORB-based state machine introduces the idea of recognizing changes in position or appearance over time to infer state transitions. By combining these ideas into one implementation, the project reinforces the use of vision-based computing for real-world automation scenarios such as railway safety systems and traffic monitoring.

**2. Background**

Automated surveillance systems form an essential part of intelligent infrastructure, particularly in transportation, where both visual awareness and timing are critical. Rail crossings and intersections often rely on synchronized signaling systems to prevent accidents and ensure safety. Developing computer vision systems that can detect and interpret both the presence of trains and the status of nearby signals serves as a foundation for intelligent traffic control. This project reproduces this type of problem in a controlled, instructional setting.

The project focuses on detecting and understanding two simultaneous visual entities: the train and the signal light. The primary goal is to ensure that the program can accurately identify the color of the signal while simultaneously detecting the position and motion of the train. YOLO (You Only Look Once) will be used as the main detection model due to its capability to identify objects in real time. This project also helps to improve our understanding beyond Haar cascades and the Viola–Jones algorithm. ORB (Oriented FAST and Rotated BRIEF), which has already been implemented in class for state-machine design, allows the program to analyze feature changes between consecutive frames and recognize transitions between distinct train states such as not_present, just_entered, moving, and left_the_frame. These states represent observable transitions that the program should be able to detect based on the combination of object presence, position, and movement patterns across time.

This project illustrates the connection between theory and application. The use of YOLO and ORB-based state machines mirrors how actual systems detect, classify, and track entities in continuous video streams. The scenario of detecting the train's presence and the signal's color at a railway crossing demonstrates how visual data can be interpreted algorithmically to represent contextual meaning. The ability to detect states automatically has significant implications for the design of intelligent alerting systems, safety monitoring platforms, and adaptive signal control systems.

**3. Overview**

The program to be written for this project must analyze a provided video (available on D2L) and output the status of both the traffic signal and the train as the video plays. The program must continuously process the video frame by frame, detect the relevant objects, determine their states, and display the corresponding results in a clearly readable textual format. The output must include the time at which each change occurs, the color of the signal at that time, and the train's state. Every change in either the signal or the train's state must be captured and displayed with its corresponding timestamp.

The expected format of the output is as follows:
t = 0.00s, signal = red, train = not_present
t = 2.00s, signal = red, train = just_entered
t = 2.05s, signal = red, train = moving
t = 9.00s, signal = red, train = left_the_frame

The detection of the traffic signal must be performed using YOLO, which will identify the bounding box of the traffic light in each frame. Once the bounding box of the signal has been detected, the program may determine the color of the signal, red, yellow, or green, by examining the pixels inside the detected region, using a color segmentation approach that operates directly on the BGR (Blue, Green, Red) values of the image without requiring conversion into other color spaces. A simple example of this type of color segmentation is shown below. This example isolates the dominant color in an image and displays the segmented region of that color. Color segmentation is just one of the approaches to solve this problem. You may implement your own approach (instead of color segmentation, if you wish to). There will be no deduction of points if your program doesn't use color segmentation, but an alternative approach to correctly detect the color of the traffic light from the video.

Input file for Example program:

Example program for color segmentation:

```
import cv2
import numpy as np
img = cv2.imread("traffic_light.jpg")
B, G, R = cv2.split(img)
mask_red = (R > 120) & (R > G) & (R > B)
mask_green = (G > 120) & (G > R) & (G > B)
mask_yellow = (R > 120) & (G > 120) & (B < 100)
mask = np.zeros_like(R)
if np.sum(mask_red) > np.sum(mask_green) and np.sum(mask_red) > np.sum(mask_yellow):
    mask[mask_red] = 255
    color = "RED"
elif np.sum(mask_green) > np.sum(mask_yellow):
    mask[mask_green] = 255
    color = "GREEN"
else:
    mask[mask_yellow] = 255
    color = "YELLOW"
segmented = cv2.bitwise_and(img, img, mask=mask.astype(np.uint8))
print(f"The detected color is {color}.")
cv2.imshow("Segmented Color", segmented)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output from Example Program:
The detected color is RED.

[Note: this program and the input file it uses are also available on D2L. Program: Week 8 -> ColorSegmentation.txt and Input File: Week 8 -> traffic_light.jpg]

This example reads an image, checks the red, green, and blue channel values, and determines which color dominates in the frame. It then highlights only that region by displaying the segmented portion of the image corresponding to the detected color. When adapted for this project, the cropped portion of the traffic light detected by YOLO may be passed to this same logic to classify the signal color for each frame of the video. This approach is intentionally simple but effective; it does not rely on complex conversions or fine-tuning, yet it produces stable results when applied to video frames.

The detection of the train must also be handled through YOLO. The YOLO detection results will provide bounding boxes for the train in each frame, allowing the program to determine its presence and track its movement over time. Based on the information obtained from these detections, **the train's state must be reported as not_present, just_entered, moving, or left_the_frame.** These states are to be determined as follows: when no train is visible, the state is not_present; when a train first appears after a period of absence, the state is just_entered; when the train continues to move within the frame, the state is moving; and when the train is no longer detected for several consecutive frames, the state is left_the_frame.

The transitions between these states may be determined using ORB-based state machine logic, which compares features extracted from consecutive frames to recognize motion and confirm state changes. Edge detection may optionally be applied to enhance the contours of the train and make the detection of motion more robust, particularly when the train is entering or leaving the frame. ORB-based state machine logic is just one of the approaches to solve this problem. You may implement your own approach (instead of ORB-based state machine logic, if you wish to). There will be no deduction of points if your program doesn't use ORB-based state machine logic, but uses an alternative approach to detect the transitions between these states correctly. Your program must produce the correct sequence of transitions for both the signal and the train, along with the correct timestamps, maintaining accurate synchronization between detected visual states and timestamps.

**4. Submission Instructions**
The types of files you are allowed to submit on D2L are .txt, .doc, and .docx. Please submit only one file on D2L. Copy-paste your Python program into a .txt, .doc, or .docx file for submission on D2L. **Email submissions will not be accepted.**

**5. Evaluation Files**
The video provided on D2L with this project is for you to test your code. Two separate videos will be used for grading the submitted programs of Project 2.

1. The first evaluation video will include a looping scenario in which the train appears and departs from the frame multiple times. The number of times the train arrives and leaves will not be stated in advance. The purpose of this video is to evaluate whether the program can detect repeated events consistently and correctly identify the start, motion, and end of each occurrence. The output must record all transitions accurately for every cycle of appearance and disappearance.
2. In the second evaluation video, either the signal light may remain green throughout and the train still passes, OR the train may arrive a few seconds later than expected. This variation will test the flexibility of the program in dealing with unexpected delays, different lighting conditions, and/or signal colors.

**6. Expected Output and Grading**
Each program will be evaluated using both evaluation videos. The total possible score is 200 points, with 100 points assigned for each evaluation video. The grading will be based on the accuracy of detecting all train states and traffic signal colors and their corresponding timestamps.

| **Accuracy of Detection Across Both Evaluation Videos** | **Points Awarded** | **Interpretation** |
|---|---|---|
| All train states and signal colors correctly detected for both evaluation videos | 200 | The program performs all detections accurately and outputs correct timestamps for every state in both videos. |
| Exactly one train state missed or detected with an incorrect timestamp across both evaluation videos combined | 180 | The program demonstrates high accuracy and produces correct outputs except for one state that is either missed or timestamped incorrectly. |
| Exactly two train states missed or detected with incorrect timestamps across both evaluation videos combined | 160 | The program is accurate in most cases but has two errors related to state detection or timing. |
| Exactly three train states missed or detected with incorrect timestamps across both evaluation videos combined | 140 | The program functions correctly overall but contains three detection or timing errors across both videos. |
| Exactly four train states missed or detected with incorrect timestamps across both evaluation videos combined | 120 | The program runs correctly but exhibits four errors in detection or timing. |
| Exactly five train states missed or detected with incorrect timestamps across both evaluation videos combined | 100 | The program executes successfully but contains five errors in detection or timing accuracy. |
| More than five train states missed or detected with incorrect timestamps across both evaluation videos combined, or fails to produce valid outputs for the evaluation videos | 0 | The program does not meet the accuracy or functionality requirements for successful evaluation. |

**7. Academic Integrity:**
South Dakota Mines is committed to academic honesty and scholarly integrity. The South Dakota Board of Regents (BOR) Policy 2.9.2 provides a comprehensive definition of "Academic Dishonesty", which includes cheating and plagiarism. All Instructors at South Dakota Mines are required to report allegations of academic misconduct to the Student Conduct Officer. BOR Policy 3.4.1 provides detailed information regarding key definitions, policy information, prohibited conduct, and the Student Conduct process adhered to at South Dakota Mines. Any student suspected of violating academic integrity standards will be reported in accordance with the process outlined on the South Dakota Mines website.

**8. Use of artificial intelligence**
The use of artificial intelligence tools is permitted for this assignment. However, consistent with the publication policy of the IEEE Robotics and Automation Society, you must provide a clear disclosure of the scope of such use. Specifically, you must explicitly state in the first few lines of your program (as comments) which portions of the work were completed independently by your team and which portions were produced or refined with the assistance of an AI tool. Your statement must be precise so that it is clear which sections of code originated from your own effort and which sections received AI assistance.
Examples of acceptable disclosure include, but are not limited to:

- Data preprocessing functions were designed and implemented by the team; AI was used only to draft the initial helpers for converting Fahrenheit to Celsius and stripping percent signs from $SpO_2$, which the team verified and integrated.
- The team developed the DNN architecture and the training/evaluation code; AI was used to generate ideas for preprocessing the 'gender' and 'ventilation' fields, which the team reviewed and finalized.
- AI tools may be used as aids, but they cannot be used to complete the entire assignment. Your program must represent your own design, implementation, and understanding. Failure to disclose AI involvement, or reliance on AI to produce the complete solution, will be treated as a violation of academic integrity. You remain fully responsible for understanding and being able to explain every part of your program, regardless of whether AI assistance was used in its development.

**9. Collaboration Policy**

Collaboration is allowed for this Project but not required. So, if you prefer, you may work individually. The team size is limited to 4 students, and every member of the team must be a student enrolled in CSC 760 this semester. The first few lines of your program should be written as comments, clearly identifying the full names of all team members along with the specific roles assigned to each member. Next to each of the following task items, DNN Design, Data Preprocessing, DNN Implementation, Experimentation and Evaluation, and Results Analysis, the relevant team member names must be listed. These task items are fixed and must remain constant across all teams.
Example
# Team Members: A, B, C, and D # Roles:
# DNN Design: A, B, and C
# DNN Implementation: A, C, and D # Data Preprocessing: B, C, and D
# Experimentation and Evaluation: A, C, D # Results Analysis: A, B, and D
Teams are allowed to work collaboratively on the same code, and it is acceptable for all team members to submit the same or similar code if the entire team has worked together on that implementation. However, each individual must still upload their own copy of the program to D2L. Within each uploaded program to D2L, the team member roles listed at the top must be identical across all team submissions. It is the responsibility of every team member to ensure that their role is correctly stated. Any team with a member whose role as listed reflects minimal (or no) involvement may receive a deduction of points. Such a deduction will apply only to the individual whose contribution is insufficient, not to the entire team.

**Note:**

- You are permitted to use any number of Python libraries to complete this Project. However, all libraries used must be ones that the instructor or the teaching assistant can install and run within PyCharm without special access or private distribution. In other words, the libraries you use must be publicly available. If you have developed your own Python package for research purposes and it is not publicly available, you may not use it for this Project. The expectation is that your submission can be executed in its entirety using only libraries that can be readily downloaded and installed in a standard Python environment.
- The policies stated in the "Late/Make-up Assignment Policy" section of the Syllabus of this course apply to this assignment.
- When you copy-paste your Python code into a .txt, .doc, or .docx file for submission on D2L, please carefully check if the indentation from your code is maintained.
- Your program must be written using Python. If your program is written in any other programming language, your submission will not be graded.
- Write only one Python program for this assignment. If you submit multiple programs on D2L, only the most recent submission will be graded.
- You are not required to present the output or submit any screenshots of the output when you upload your program to D2L.
- A photo/scan of handwritten code will not be graded.
- You are NOT allowed to post any ideas/hints provided for this Project and/or your solution for this Project on a public repository such as GitHub.
- You are NOT allowed to upload the problem statement of this Project in part or in full on any website on the internet, including but not limited to Chegg, Course Hero, and Stackoverflow.