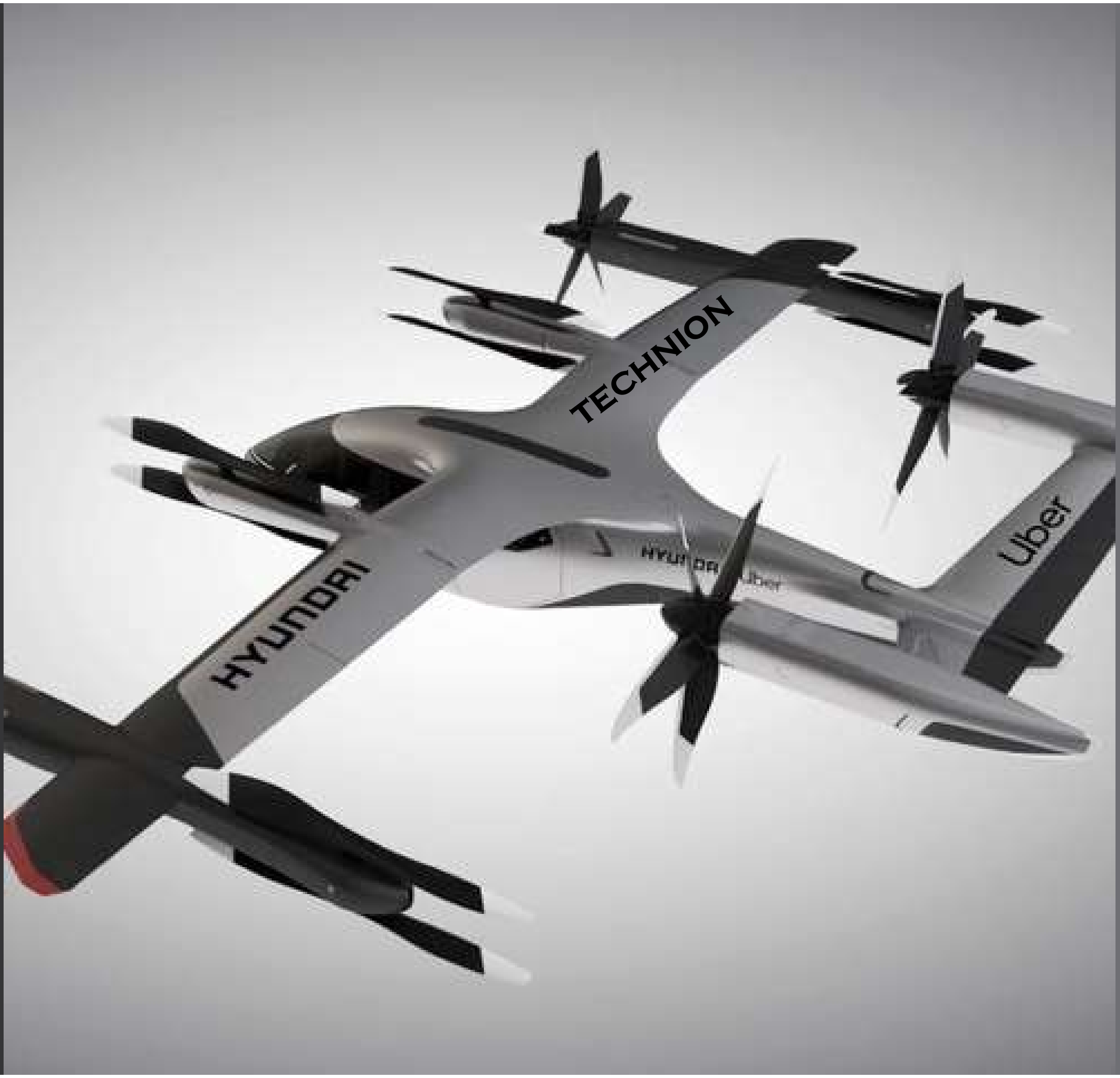
The Oracle

Determining Fitness-For-Landing Probabilities

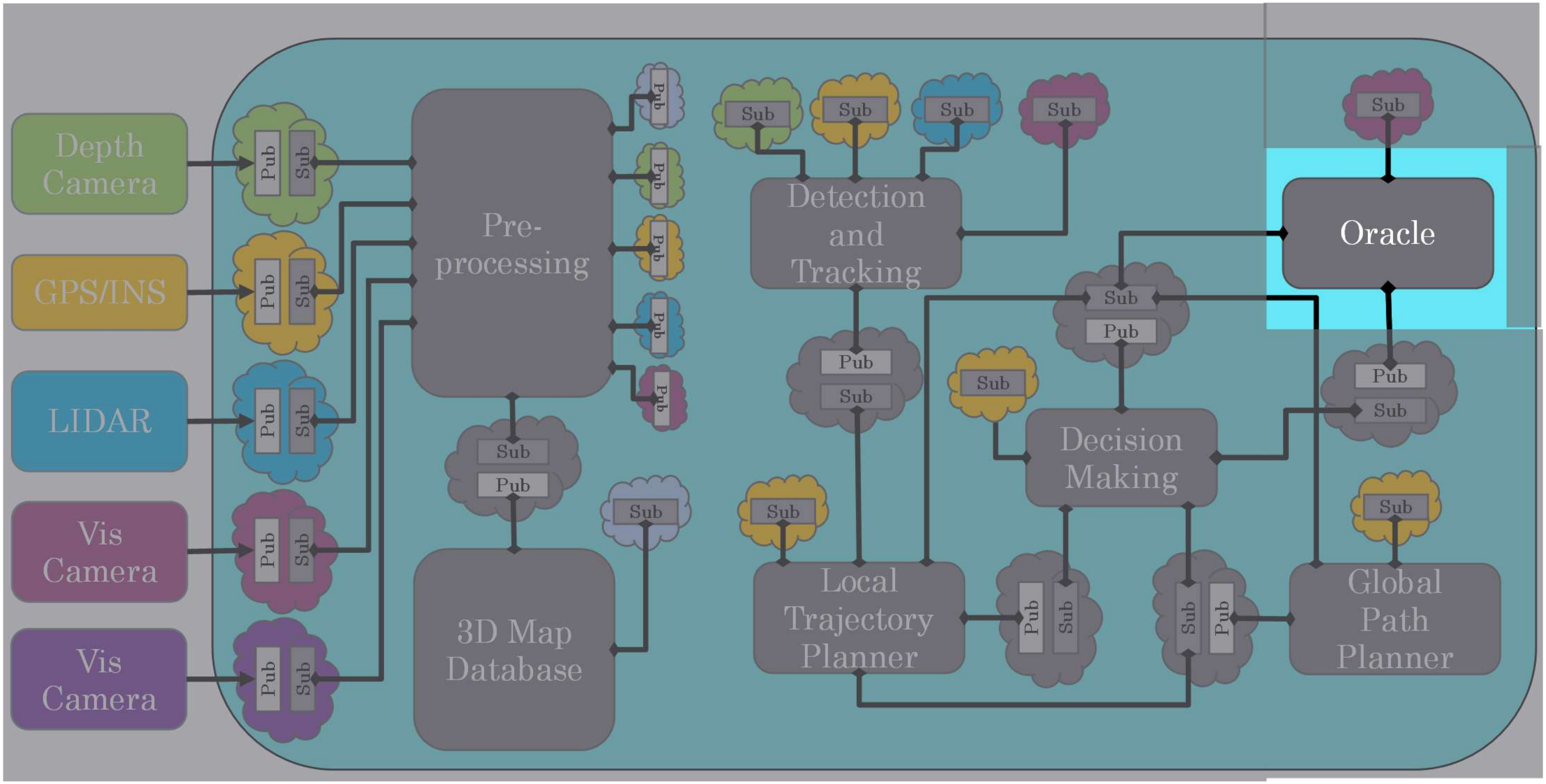


Overview: Autonomous Landing Project Objectives

Find a safe landing place for a drone in a crowded urban environment

- Do it fast
- Prevent collisions
- Provide performance guarantees (False positive/ False Negative)
- Implement and test in a simulation (phase 1) and a live demo (phase 2)

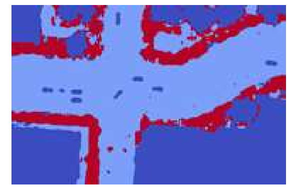
The Oracle within the system



The Oracle within the system

- Extracting information from input images for the other modules to use

Oracle Objective



- Given a single image of an urban environment
- Calculate **for each pixel** of the image its suitability for being part of a **landing spot**
- **Suitability** is measured in the probability sense: the closer to one, the more likely it is to be appropriate
- The Oracle is memoryless: assessment based on current image alone

Landing Site Fitness as a Classification Problem

- Objects that can be observe in a city are classified into some finite set of classes, each class is associated with an index from $0 \leq c \leq |C| - 1$:

$$C = \{sidewalk, road, car, dense\ vegetation, \ldots\}$$

- A $Score$, measuring how suitable is a given pixel to be part of a landing site, is attached to each class:

$$Score(c): C \rightarrow [0,1]$$

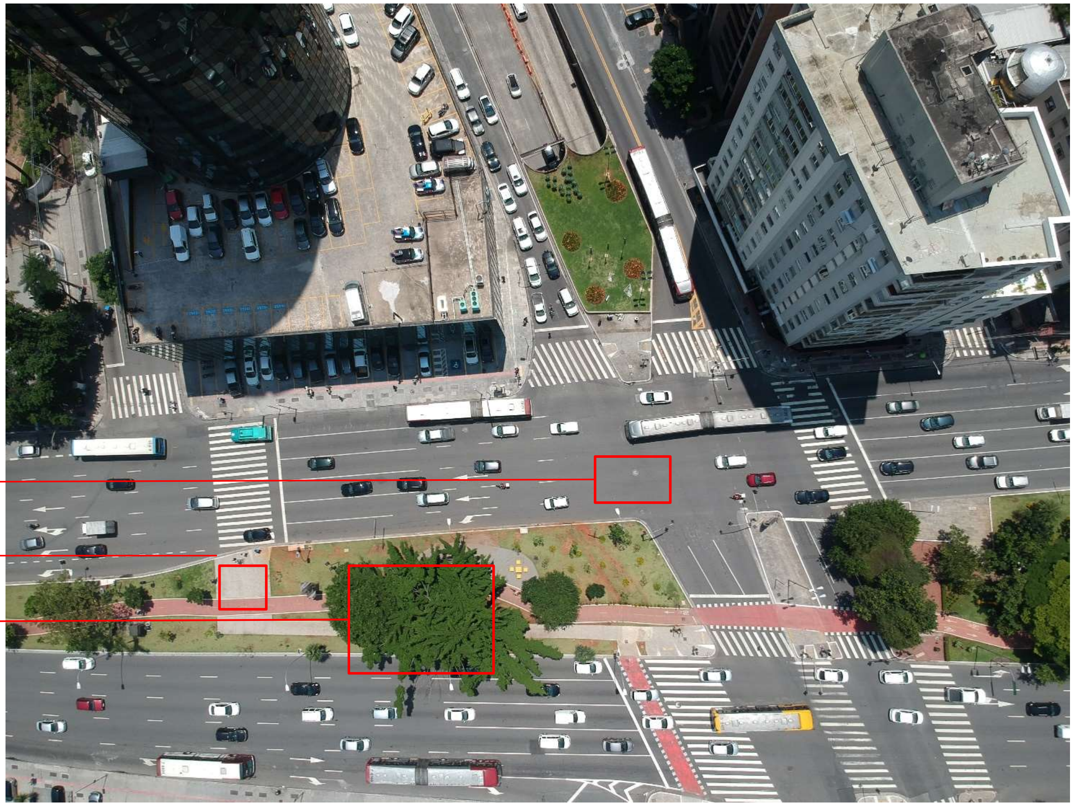
- This function is such that:
  - $Score(c) \approx 0 \Rightarrow$ the class is not suitable for landing
  - $Score(c) \approx 1 \Rightarrow$ the class is very suitable for landing

Basically, We want to land on the classes with the high scores

$Score(road) = 0$

$Score(sidewalk) = 1$

$Score(dense\ vegetation) = 0$

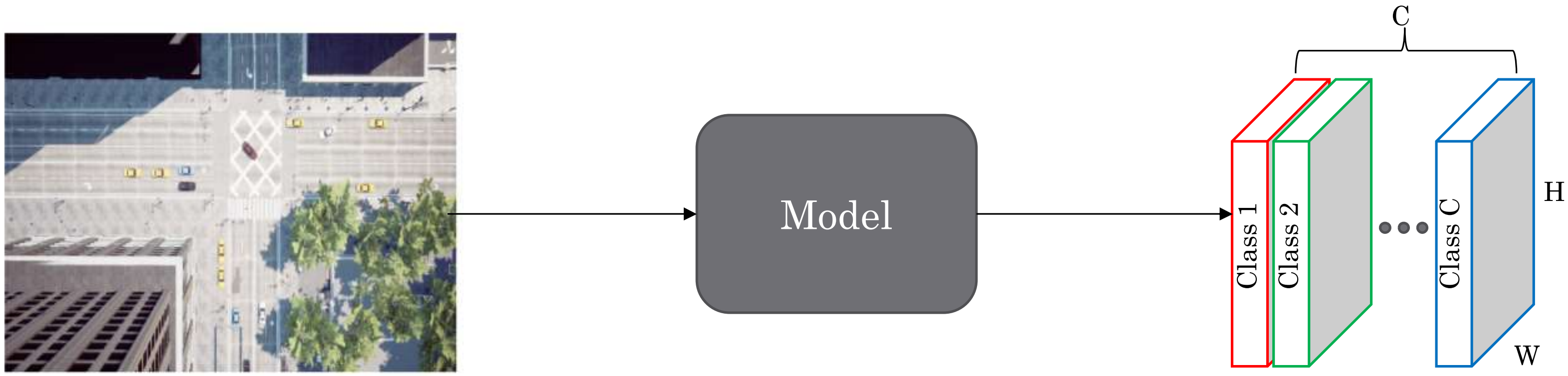


Semantic Segmentation

- **Semantic Segmentation** is the task of classifying each pixel in an image with the corresponding class it belongs to.

- As opposed to standard classification, the output of a semantic segmentation algorithm is not a single class label, but the probability for each of the classes, i.e. for each pixel $[i, j]$ we get the vector:

$$(p_0([i,j]\ is\ class\ 0), p_1([i,j]\ is\ class\ 1), \dots)$$

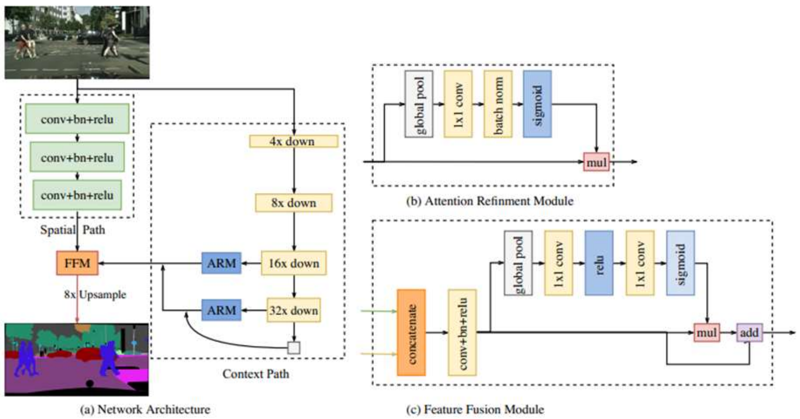


Selected Algorithms

- An extensive literature review was performed
- Two different semantic segmentation algorithms were found to be adequate to solve the fitness problem, due to their relative high performance and low runtime:
  - Bilateral Segmentation Network for Real-time Semantic Segmentation
  - Fast Semantic Segmentation Network
- These two algorithms are briefly reviewed next

BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation

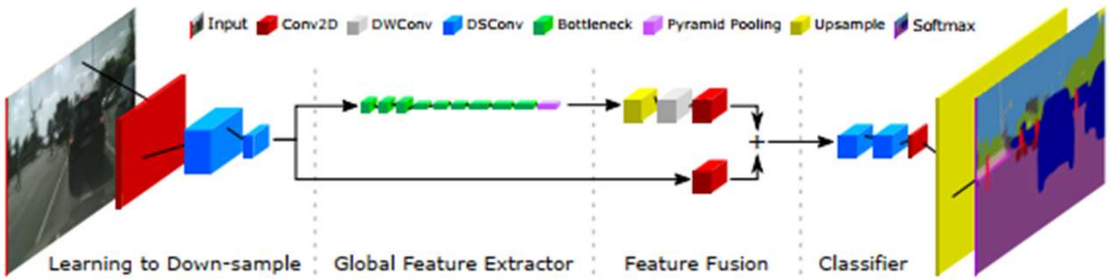
This model containing 2 paths - one with spatial information, where each convolution is affected by a small number of inputs, and with moderate down sampling, and a context path, with aggressive down sampling. Then they fuse the 2 paths for the prediction.



Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 325-341).

Fast-SCNN: Fast Semantic Segmentation Network

This model first down sample the image to 1:8 resolution feature map. Then using the two-branch approach. One path for feature extraction using depth wise separable convolutions and inverse residual blocks. The second path use simple convolution and then aggregated with the first path.



Poudel, R. P., Liwicki, S., & Cipolla, R. (2019). Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*.

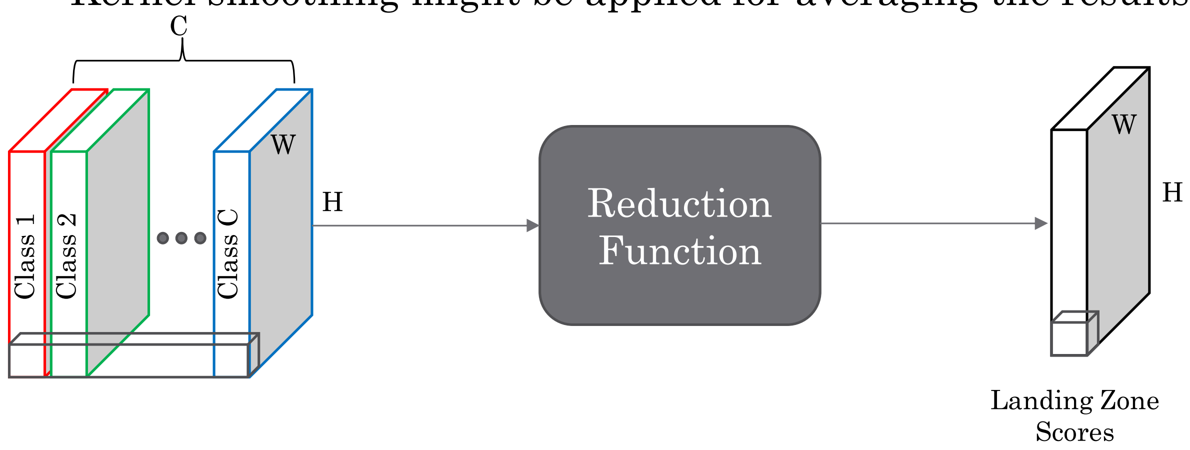
Reduction Function

- The reduction function converts the class probabilities(=output of the semantic segmentation) to safety scores.

$$R\colon\ [0,1]^{|C|} \rightarrow [0,1]$$

- There are two methods for the reduction function: **threshold** and **weighted average**

- Kernel smoothing might be applied for averaging the results.



Reduction Function

- Threshold mode:
  - For each pixel we take the class with the highest probability as predicted by the model. If the probability is greater or equal to a given threshold, we substitute it with the corresponding landing score.

$$pmax = \max\{p([i,j]\ is\ class\ c\}, cmax\ - the\ class\ c\ that\ gives\ pmax$$

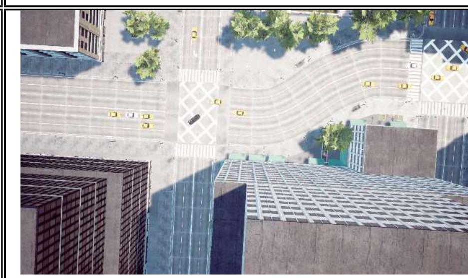
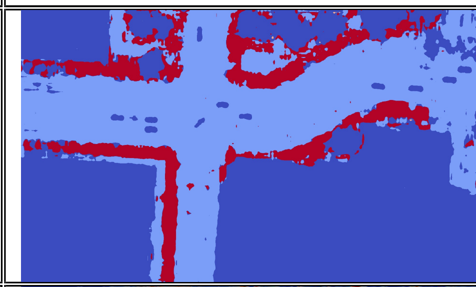
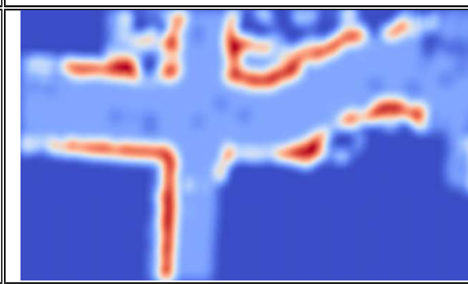
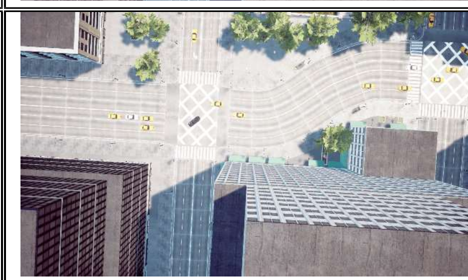
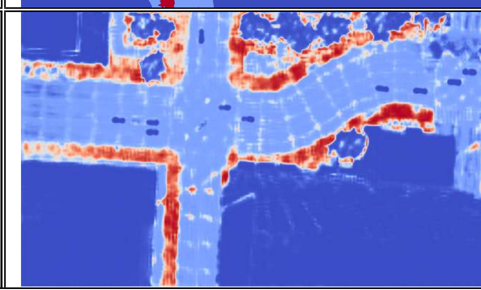
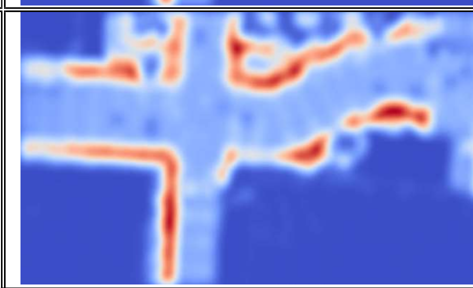
$$Score = \begin{cases} Score(c) & pmax \geq threshold \\ 0 & pmax < threshold \end{cases}$$

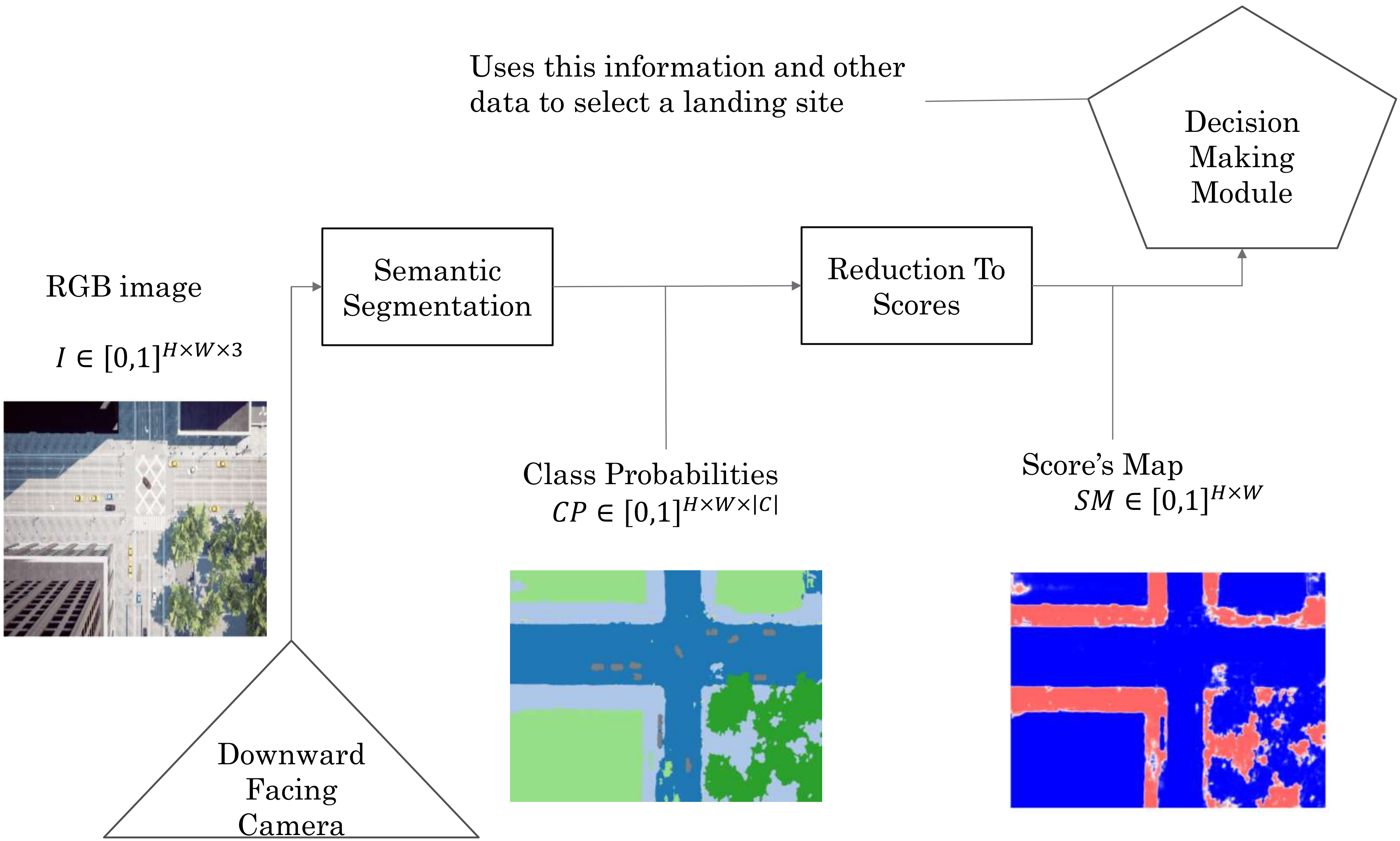
- Weighted Average mode:
  - For each pixel we compute the weighted average of the class probability distribution with the given classes scores

$$Score = \frac{1}{|C|} \sum p([i,j]\ is\ class\ c) \cdot Score(c)$$

- Kernel Smoothing:
  - Using a blurring kernel each landing score is computed as the weighted average of neighboring scores.

Reduction Function

	Input Image	Reduction	Smoothing
Threshold			
Weighted Average			


Uses this information and other data to select a landing site
Decision Making Module
RGB image
$I \in [0,1]^{H \times W \times 3}$
Semantic Segmentation
Reduction To Scores
Class Probabilities
$CP \in [0,1]^{H \times W \times |C|}$
Score's Map
$SM \in [0,1]^{H \times W}$
Downward Facing Camera

Experiments & Results

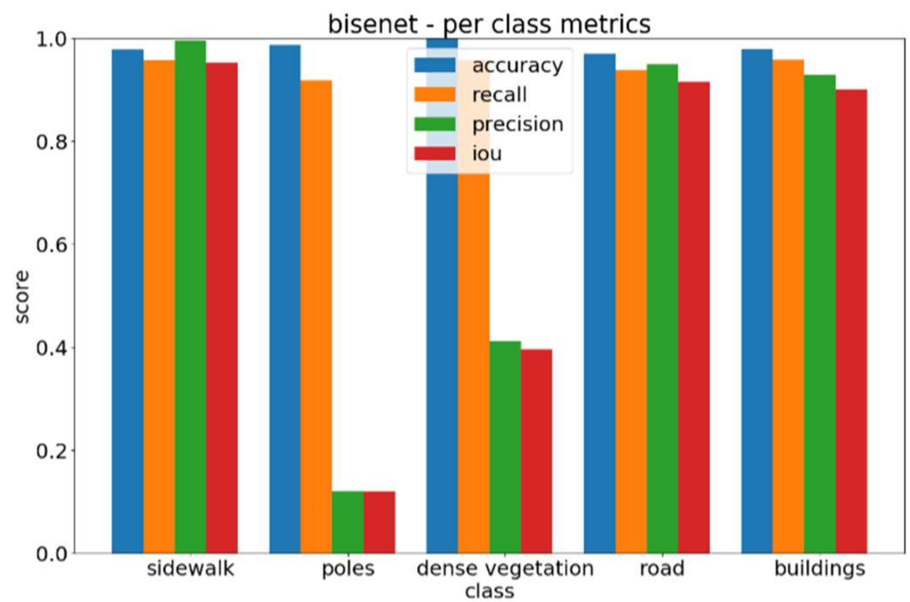
Experiment Setting:

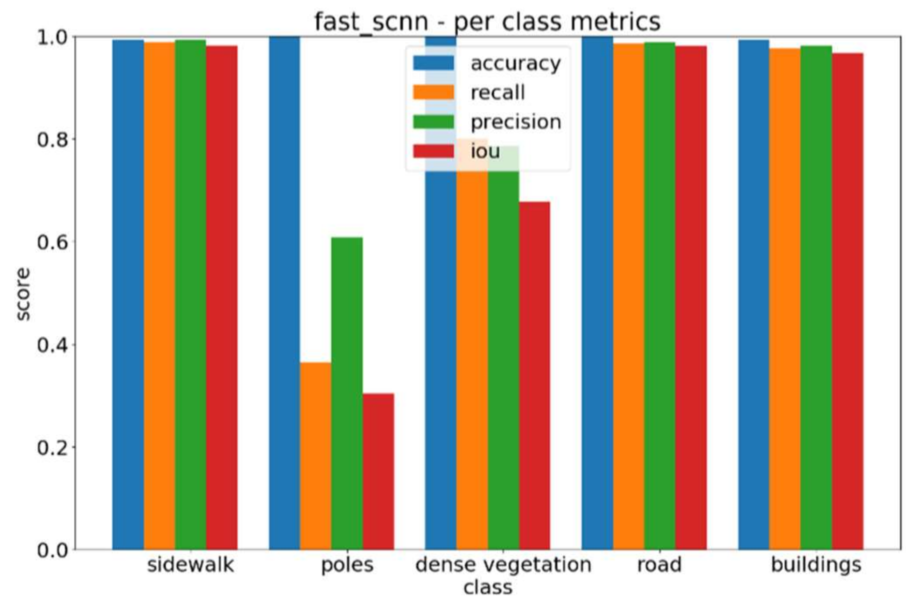
- Dataset
  - Created using AirSim on top of Unreal Engine
  - Contains image and full pixel-wise labeling for each image
  - Size of the training set is 995 images
  - Size of the test set is 438 images
- Hardware
  - Single CPU - Xeon(R) CPU E5-2683 v4 @ 2.10GHz
  - Single GPU - GeForce RTX 2080
- Software
  - Linux(Ubuntu)
  - Python 3.8
  - Pytorch 1.5

Semantic Segmentation Correctness:

metric name	segmentation model	mean value
mean iou	bisenet	0.803
mean iou	fast scnn	0.886
pixelwise accuracy	bisenet	0.957
pixelwise accuracy	fast scnn	0.993

Figure 10: All Classes Metrics


bisenet - per class metrics
accuracy
recall
precision
iou
score
1.0
0.8
0.6
0.4
0.2
0.0
sidewalk
poles
dense vegetation
road
buildings
class


fast_scnn - per class metrics
accuracy
recall
precision
iou
score
1.0
0.8
0.6
0.4
0.2
0.0
sidewalk
poles
dense vegetation
road
buildings
class

Runtime Speeds (in m.s.):

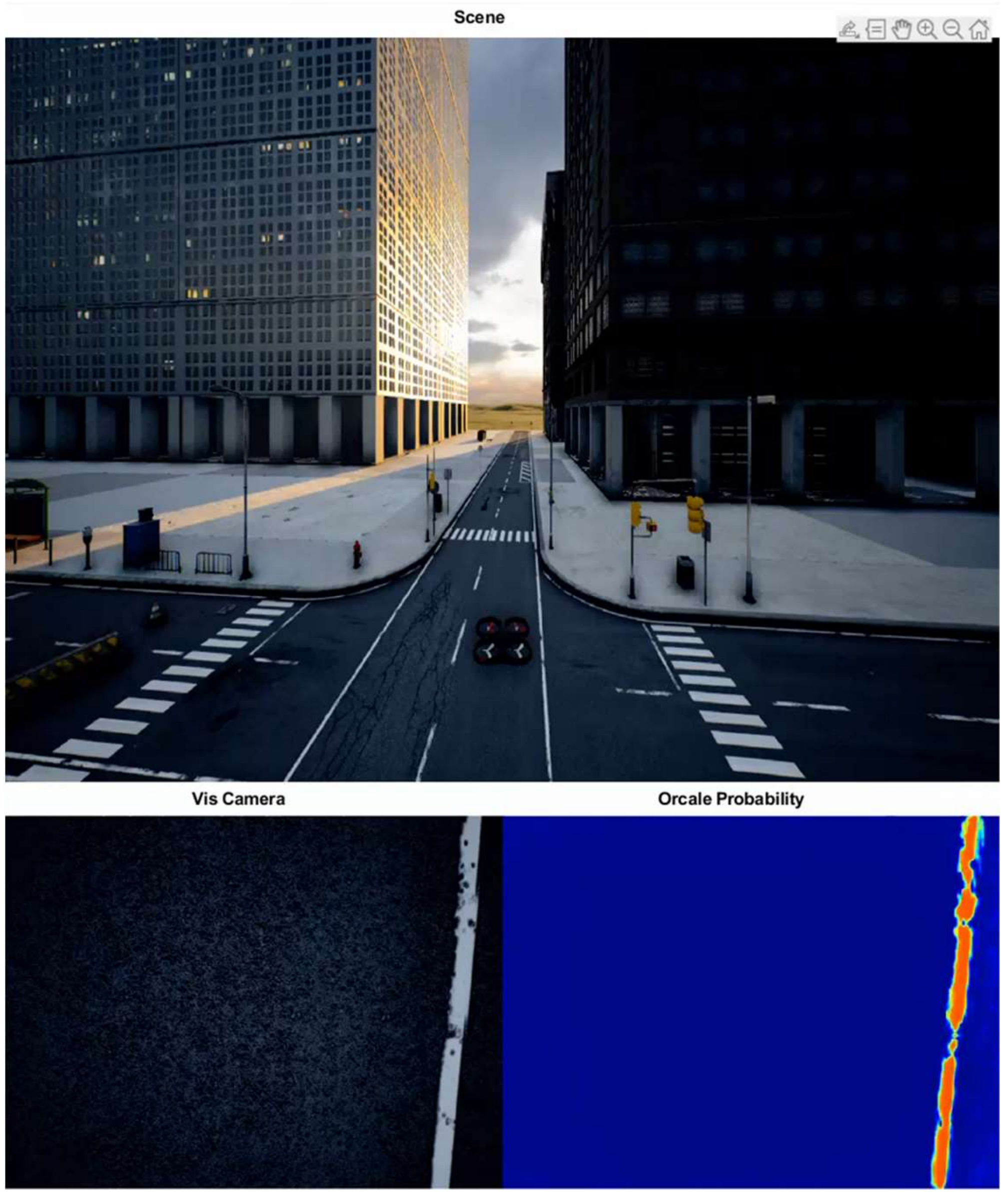
device	module	segmentation model	mean value
cpu	data conversion input transform	bisenet	50.399
cpu	data conversion input transform	fast scnn	45.864
cpu	data conversion output transform	bisenet	9.939
cpu	data conversion output transform	fast scnn	11.422
cpu	oracle	bisenet	459.432
cpu	oracle	fast scnn	2928.876
cpu	segmentation	bisenet	393.751
cpu	segmentation	fast scnn	510.711
cuda	oracle	bisenet	46.010
cuda	oracle	fast scnn	26.350
cuda	segmentation	bisenet	20.344
cuda	segmentation	fast scnn	11.182

Total time = (cpu input + output conversion) + (cuda oracle time)

- Fast-SCNN: (46 + 12) + (26) = 84ms = 11 FPS
- BiSeNet: (50 + 10) + 46 = 106ms = 9FPS

Integration

- Oracle is integrated and ready for phase 1 on the mission computer
- Receives continuously images and outputs the correct output to the decision-making module


Scene
Vis Camera
Orcale Probability

Conclusions

- We have chosen to base the oracle on semantic segmentation, over geometric reconstruction, plane analysis.
- Implemented 2 segmentation models, and both achieved good results on the simulated dataset.
- Got to runtime speeds of ~ 10 FPS
- Kept the module simple and flexible – the mode of operation can be configured, and the definition of the landing site can be changed by modifying the set of classes and their scores.
- Oracle is integrated into the mission computer

Future Work

- Transferring from simulation to the real-world
  - **Weak Supervision** - train the segmentation models based on partial labeling, for example by using image level classification labels or object bounding box annotations in the images
  - Transfer Learning
    - **Synthetic Data Set Augmentation**: manipulating the synthetic data to look more realistic or render it with different settings to force the model learn more meaningful feature.
    - **Fine Tunning:** train the segmentation model on a large dataset and then retrain last layers on real world small dataset.

- Optimizations:
  - **Compiling model with TensorRT** - TensorRT is offered by NVIDIA for optimizing deep learning models resulting lighter and faster inference time.
  - **Software Optimization** - profiling and optimizing the different parts in the module