**Computer Vision Project – Semantics segmentation**

Liurcă Daniel

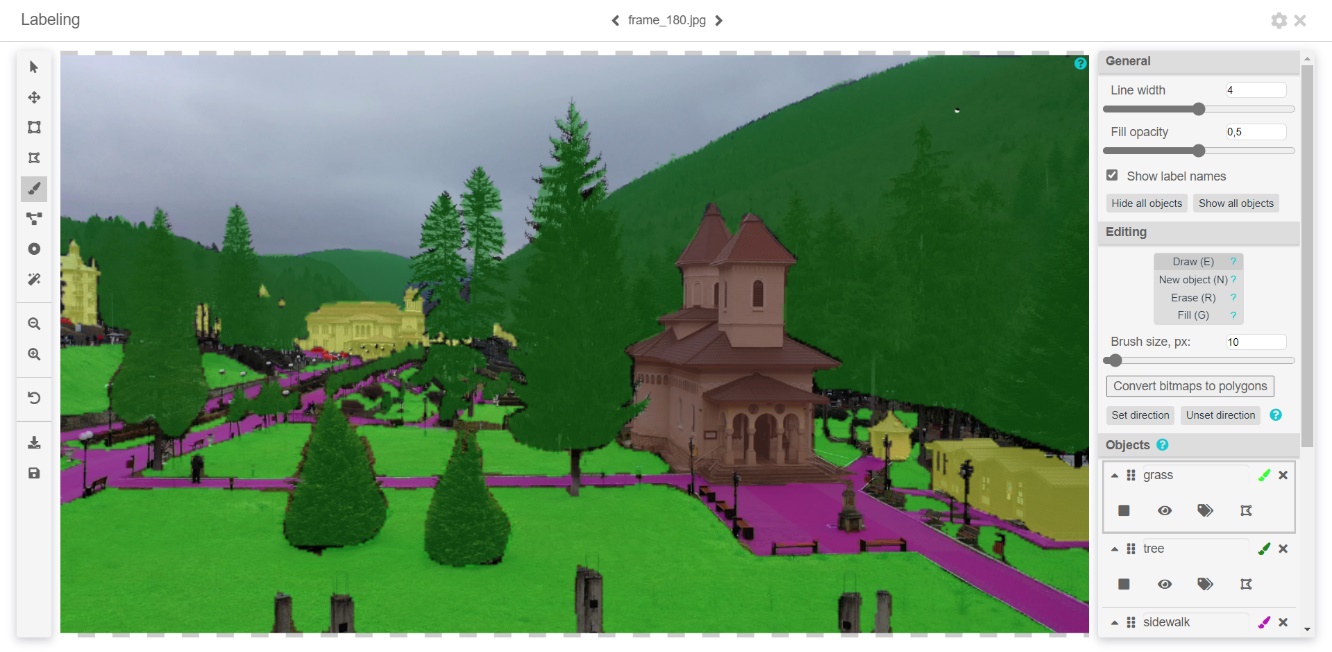
1. **Abstract**

The scope of this project is to try segment the given scene into several classes. The classes I am going to be using are: church, grass, road, sidewalk, building, tree, car. The method I used is manually labelling some of the frames and then training a UNet neural network. Finally, I apply the obtained model on every frame in a video and output the resulting video.

1. **Obtaining the data**

The first step is a subset of all the frames in the video. I decided to extract a frame every 3 seconds. That means every 90 frames, since the video is at 30 frames per second.

After that I selected 15 of what I considered to be the frames that encapsulate the best all the scene. After that, I toke those 15 frames and labelled them is the <http://www.sentisight.ai>. This is how this tool looks like:



After downloading the labelled images, it looks like this (which is really visually satisfying):

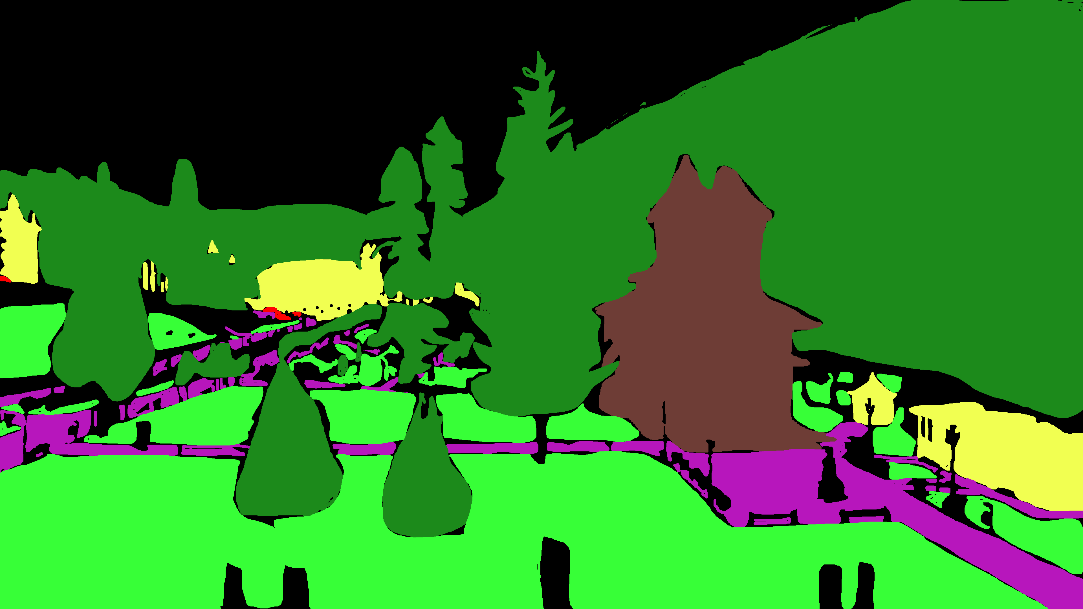


Figure 1. Frame 180

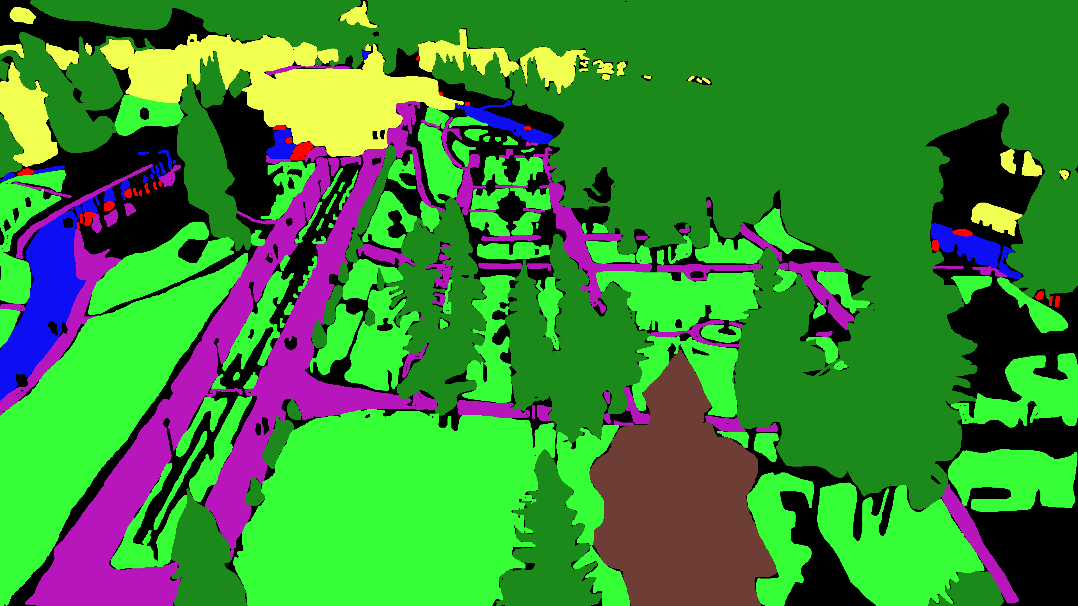


Figure 2. Frame 720

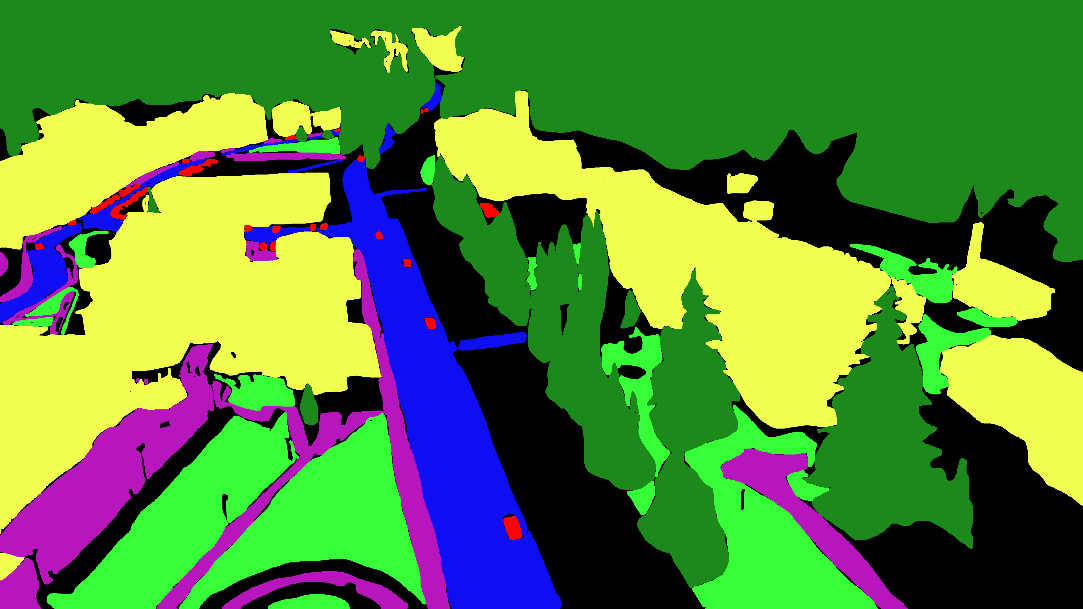


Figure 3. Frame 1350

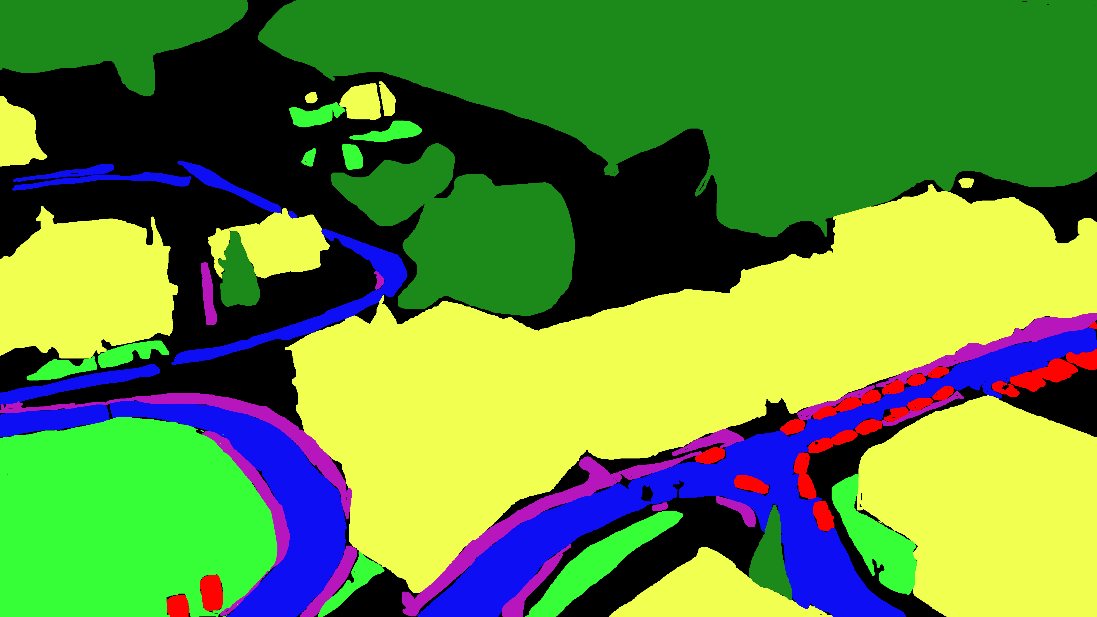


Figure 4. Frame 1710



Figure 5. Frame 1890

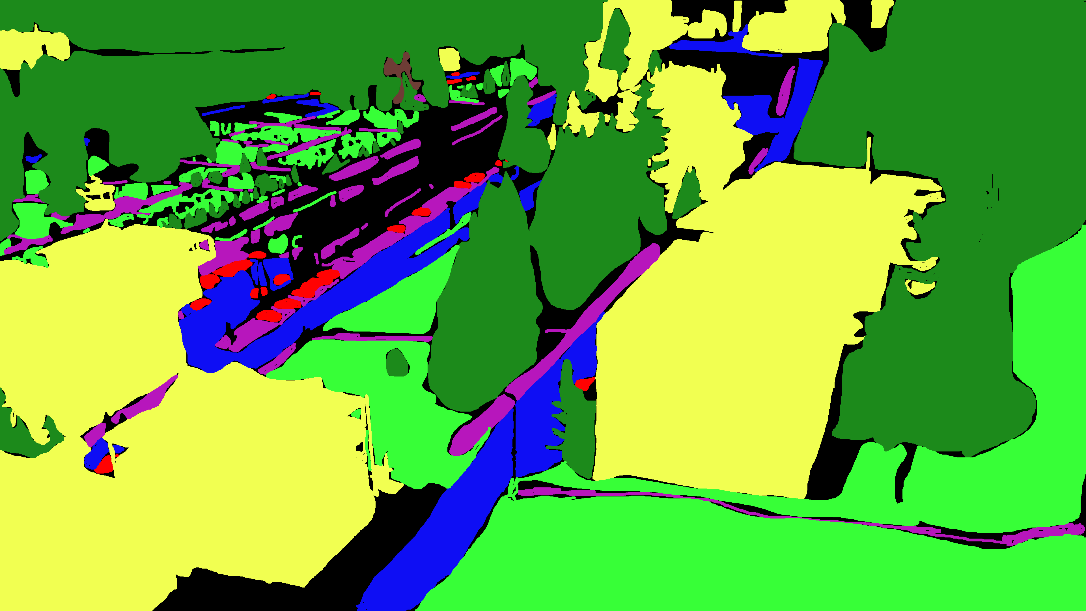


Figure 6. Frame 2070

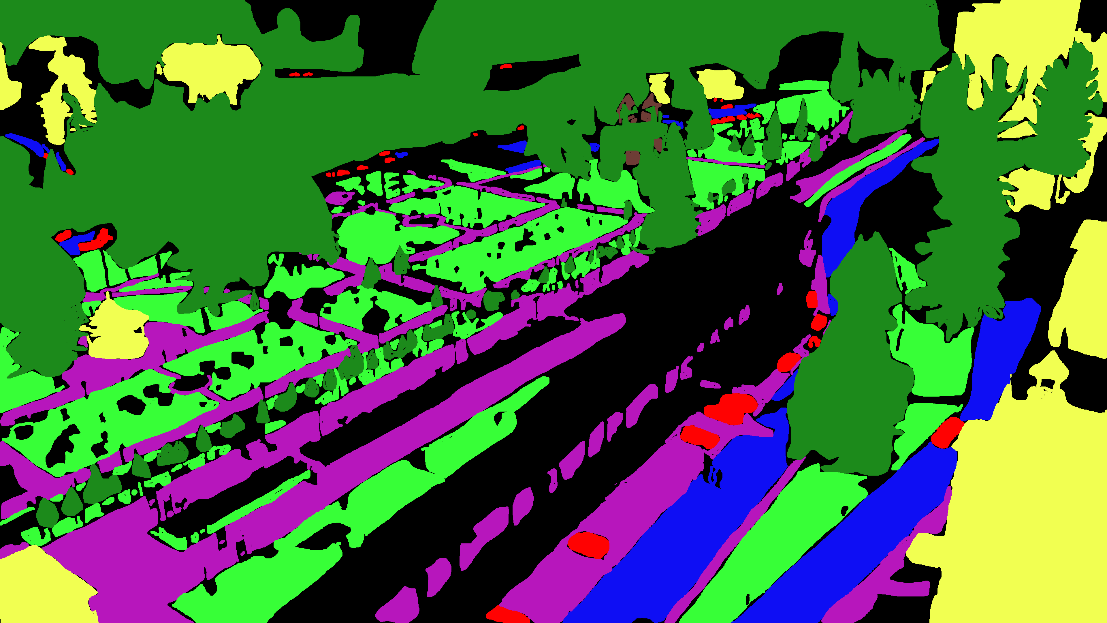


Figure 7. Frame 2340

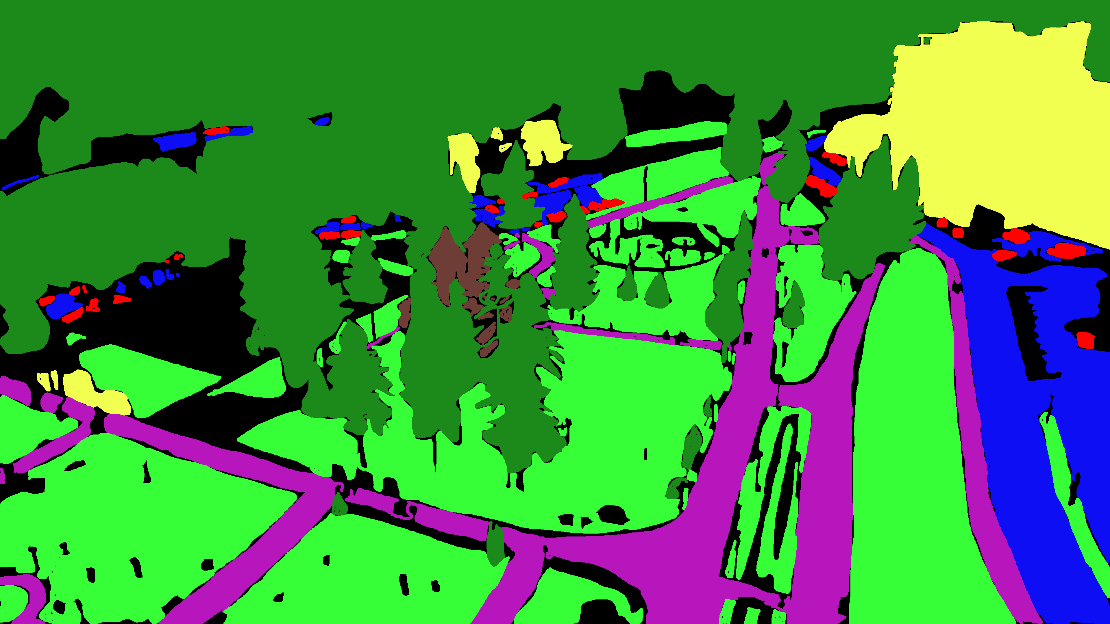


Figure 8. Frame 2610

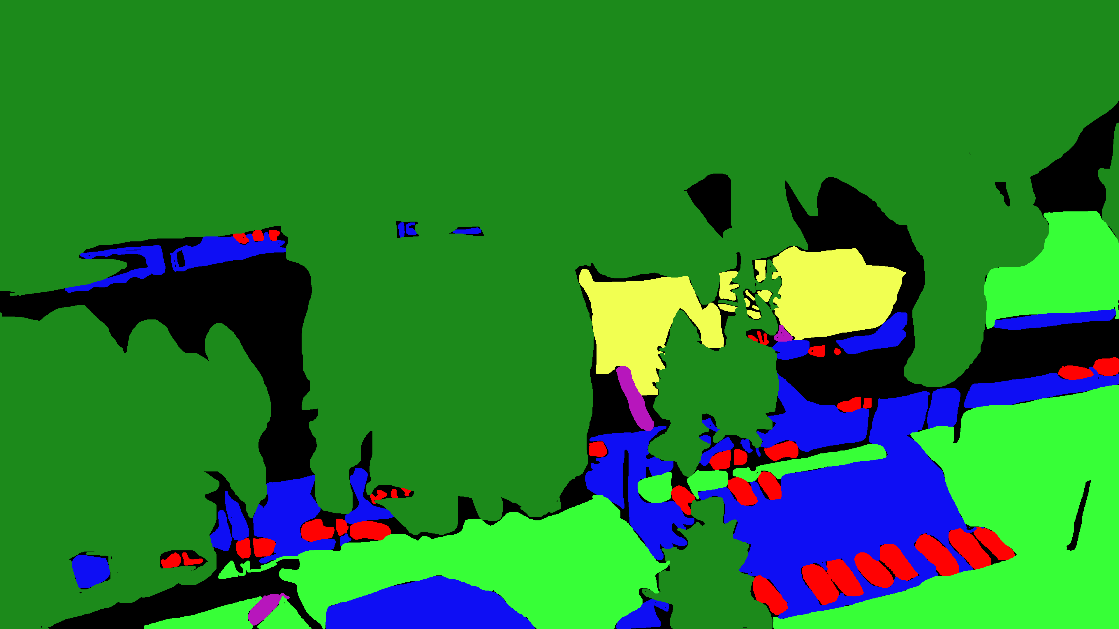


Figure 9. Frame 2880

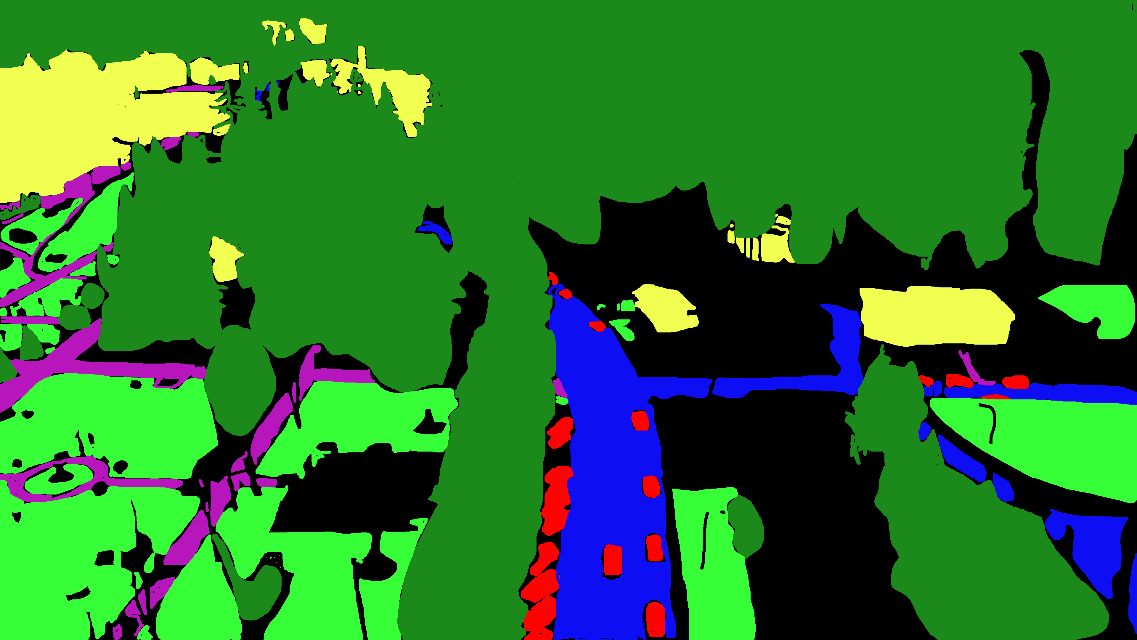


Figure 10. Frame 3150



Figure 11. Frame 3330

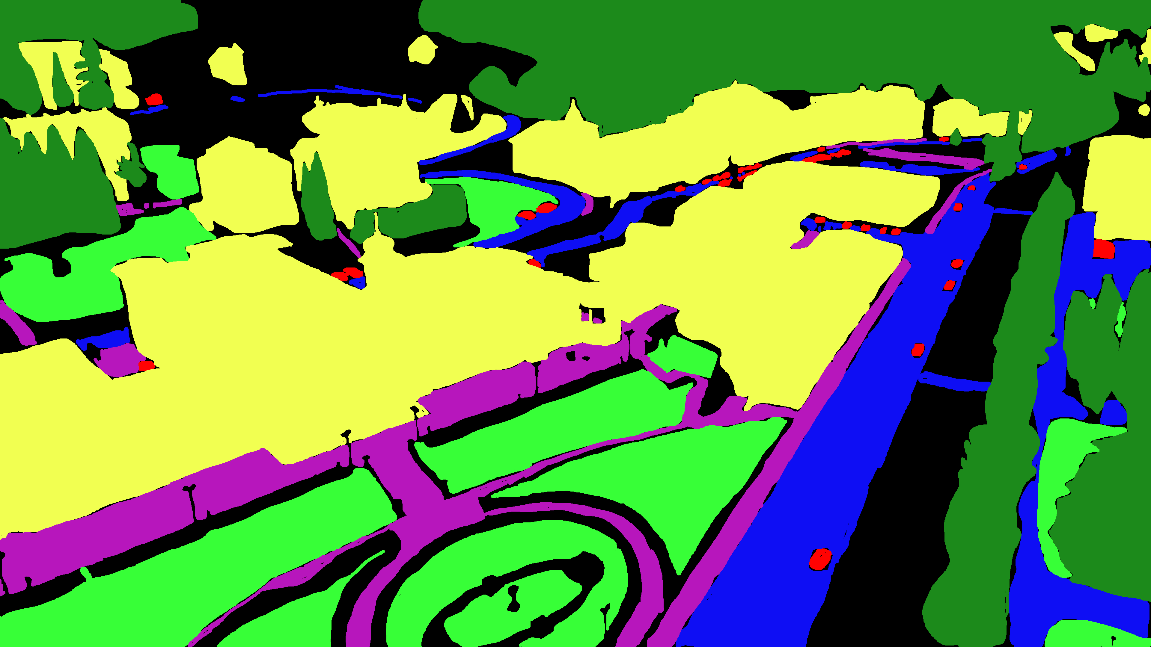


Figure 12. Frame 3960

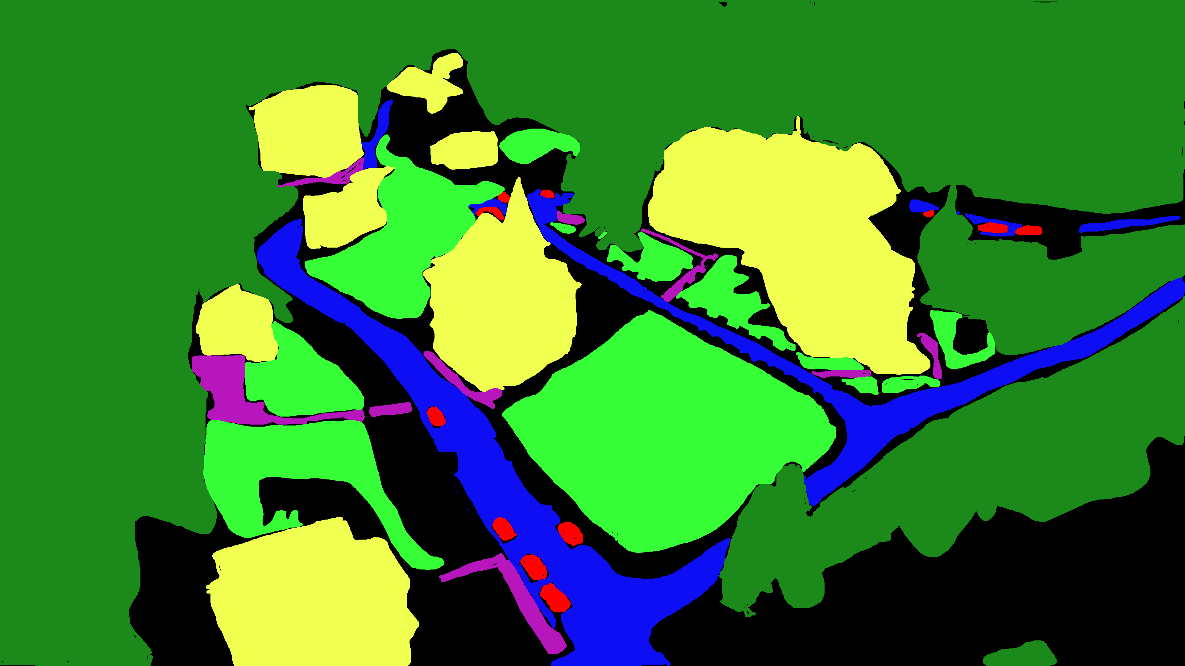


Figure 13. Frame 4680

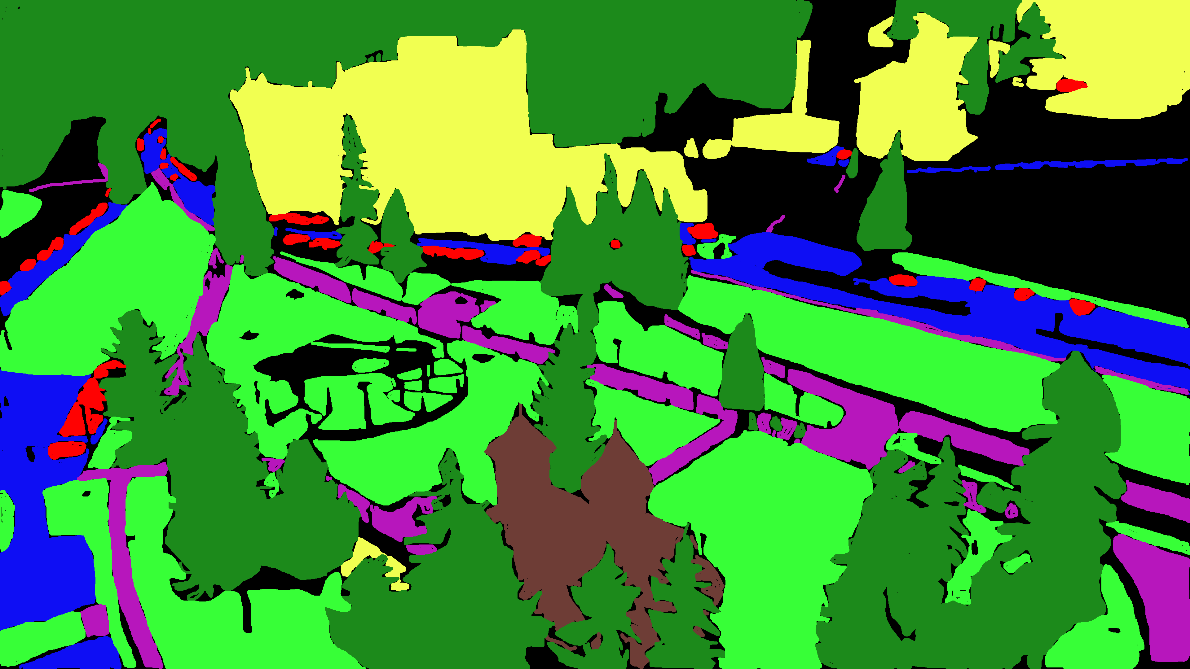


Figure 14. Frame 8550

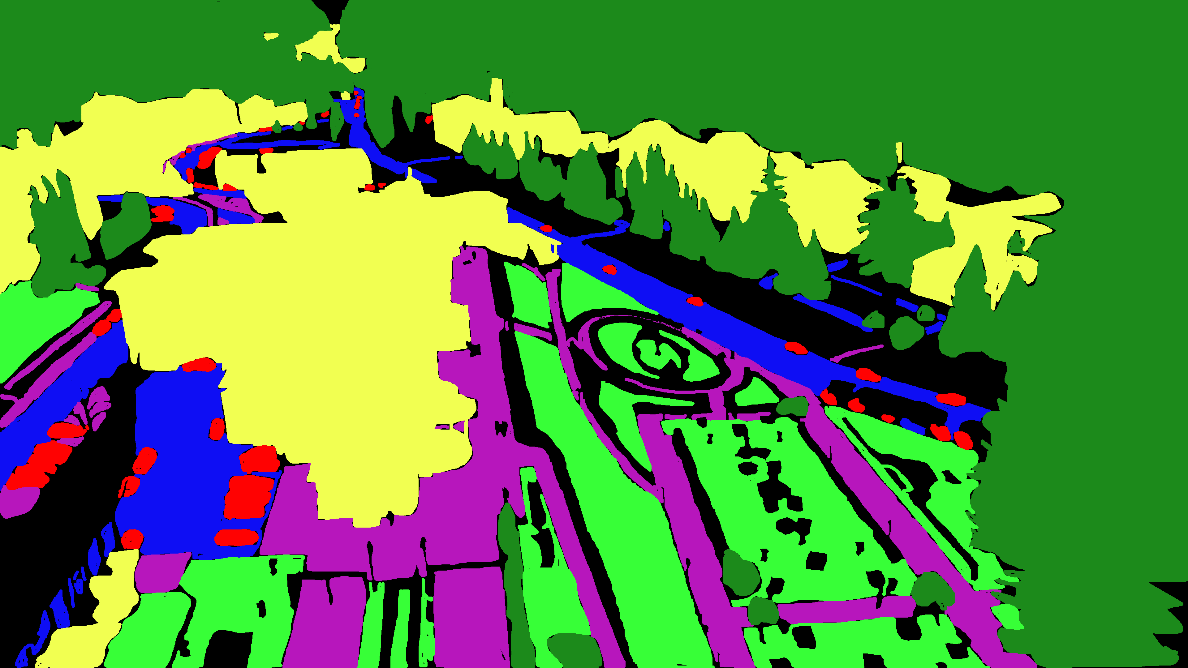


Figure 15. Frame 9000

The colors are representative for one of the seven classes. The correlation is:

* Car – red
* Sidewalk – purple
* Road – blue
* Building – yellow
* Church – brown
* Grass – light green
* Tree – dark green

1. **Implementation**

The second step is defining a neural network. I use the UNet architecture since it is built for this type of problem. The architecture looks like this, with the mention that input size is according to our problem and every layer has 4 times less filters:

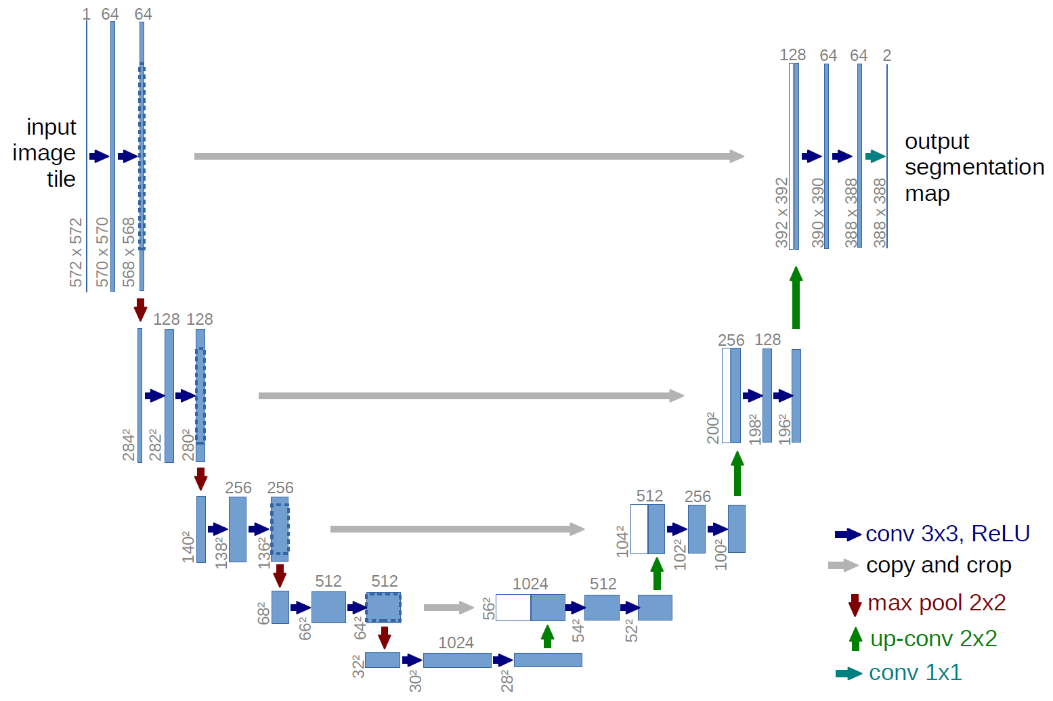


Figure 16. U-Net Architecture

Since the video was really high, I had to cut every dimension in half, in order to have enough GPU memory to train the network.

The last step is training and running the models. The main idea is to transform the class of into categorical data. That is having a vector for every pixel that contains a 1 at the position of the class it contains and 0 in the rest.

The accuracy was not great(around 55-60). Because some of the classes appear less in training set, I chose a threshold of 0.3, so that less frequent neuron classes can activate. Obviously, for every pixel I choose the class with the highest activation value.

1. **Results**



Figure 17. Example result 1



Figure 18. Example result 2



Figure 19. Example result 3



Figure 20. Example result 4



Figure 21. Example result 5



Figure 22. Example result 6



Figure 23. Example result 7

1. **Bonus**

So, as a bonus I ran the model I trained for vegetation detection on the delta images. It is interesting to observe that if that model trained in the delta can detect vegetation from the mountains. Here are a few of the obtained frames:



Figure 24. Delta model result 1



Figure 25. Delta model result 2



Figure 26. Delta model result 3



Figure 27. Delta model result 4

1. **Conclusion**

We can see that the model is pretty good at recognizing the classes that appear a lot in the training data. The class car doesn’t really appear. The class church is confused with the class building.

In conclusion, using the UNet architecture, based on convolutional layer, yielded satisfying results. The labelling was quite laborious work, but in the end, it paid off. Better results can be obtained by having more training examples and an even more meticulous labelling.