Autonomiczne pojazdy

To przecież już...

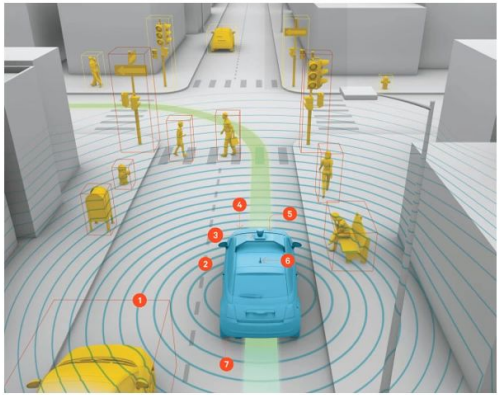
Agenda

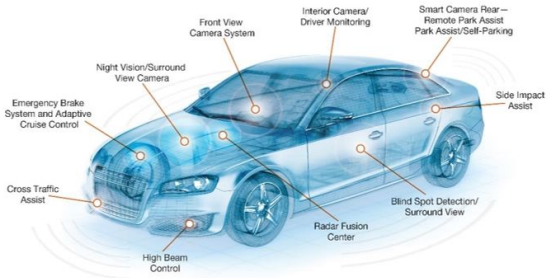
1. Co mamy do dyspozycji
2. Garść statystyk
3. Uczenie nienadzorowane z BB8
4. Różnice względem popularnych rozwiązań
5. Zrób to sam, czyli ,,uczenie” i granie

*nie nauka

Co mamy do dyspozycji

Chess Pieces: Self-Driving Car Sensors



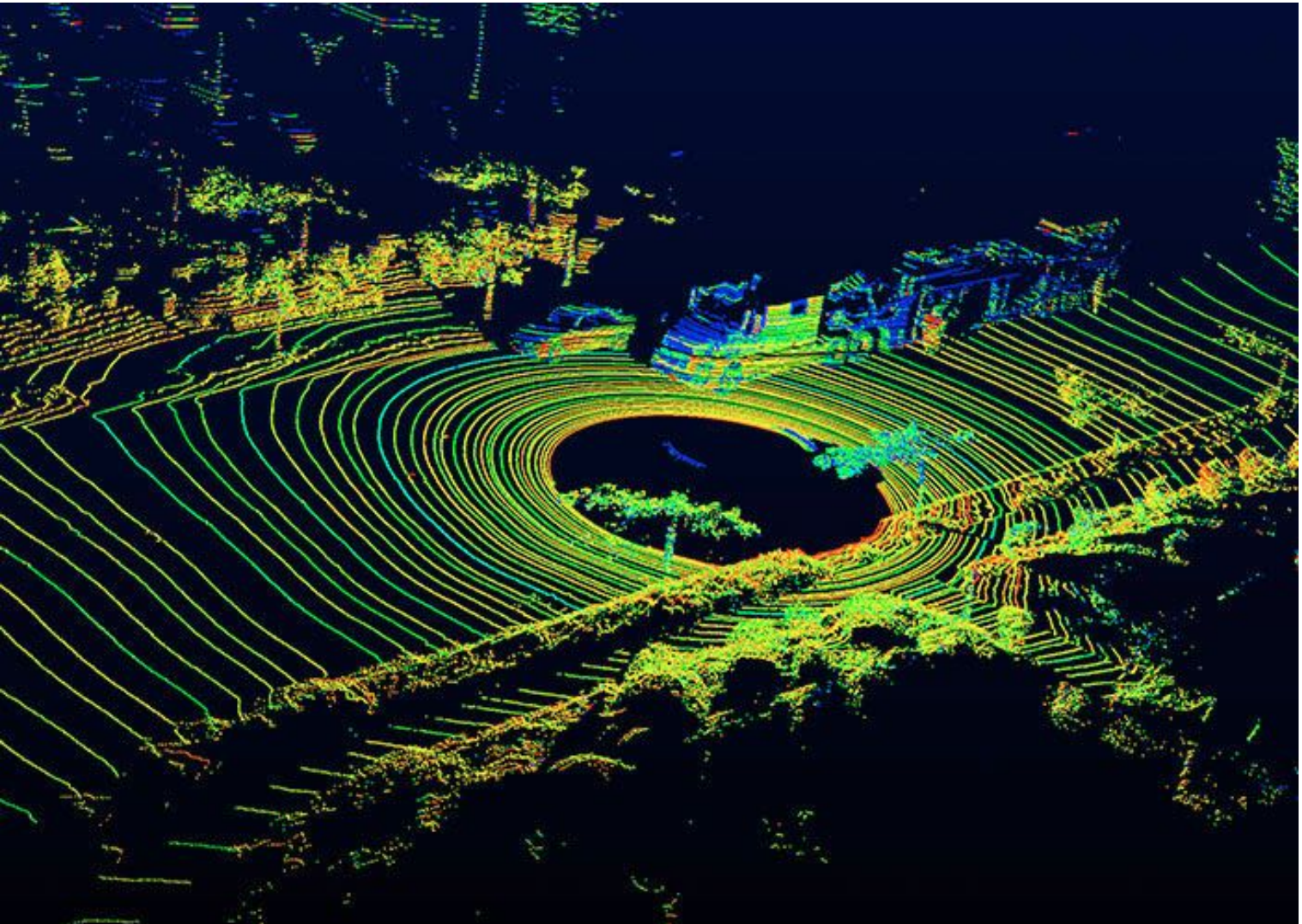


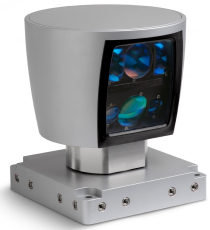
External

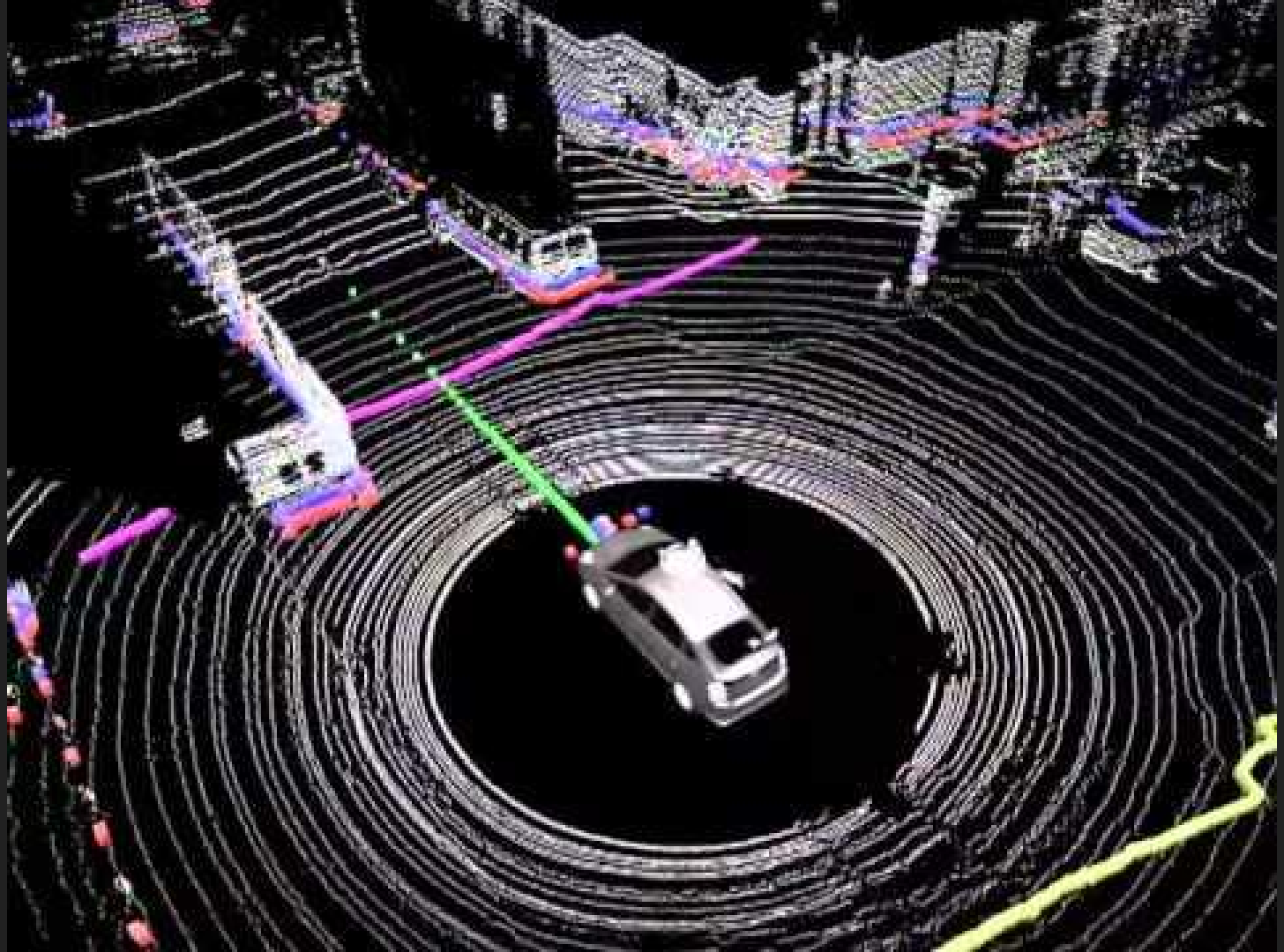
1. Radar
2. Visible-light camera
3. LIDAR
4. Infrared camera
5. Stereo vision
6. GPS/IMU
7. CAN
8. Audio

Internal

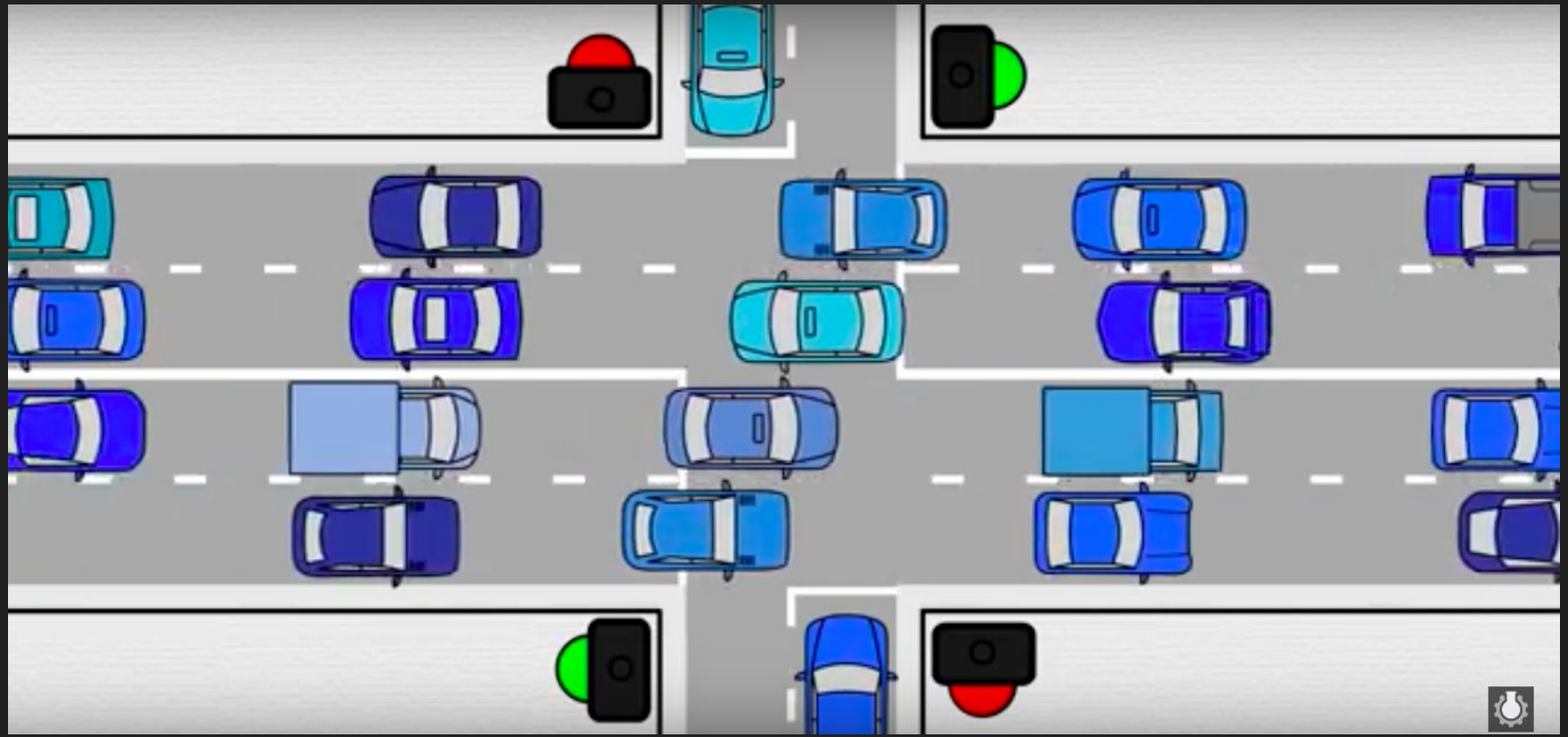
1. Visible-light camera
2. Infrared camera
3. Audio







Synchronizacja.. Ten temat należy do innej działki



Garść statystyk

10,000 mil to 16,093 km

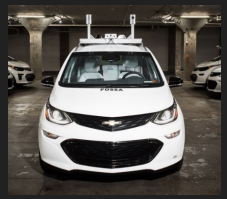
Tesla => 0.5 - 1/200mln

Człowiek => 0.1

Waymo => 2 - prace drogowe, śmieci na drodze).

BMW => 16, przez niewyraźne oznaczenia linii.

Nissan => 68

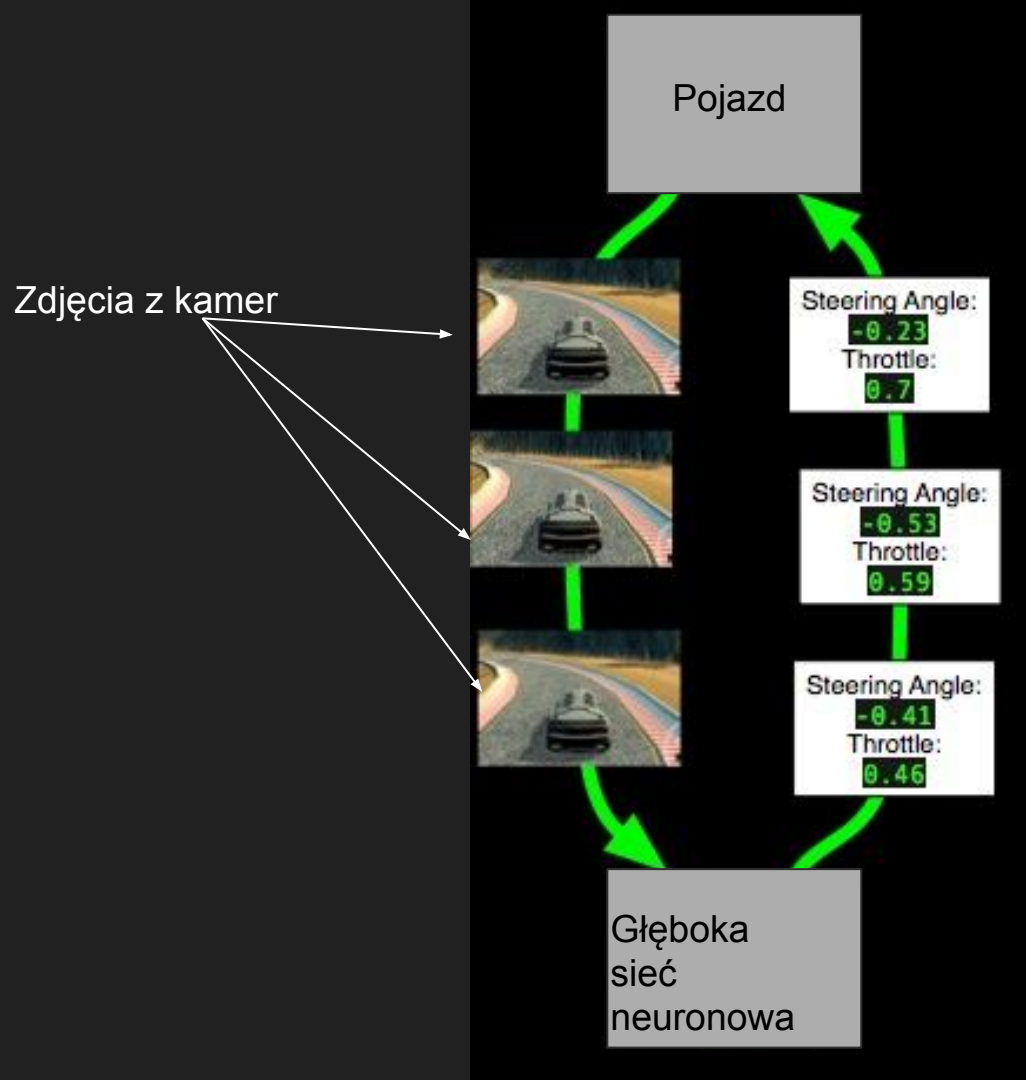


Cruise Automation => 185

Mercedes Benz => 4999, w centrach miast, nie na autostradach

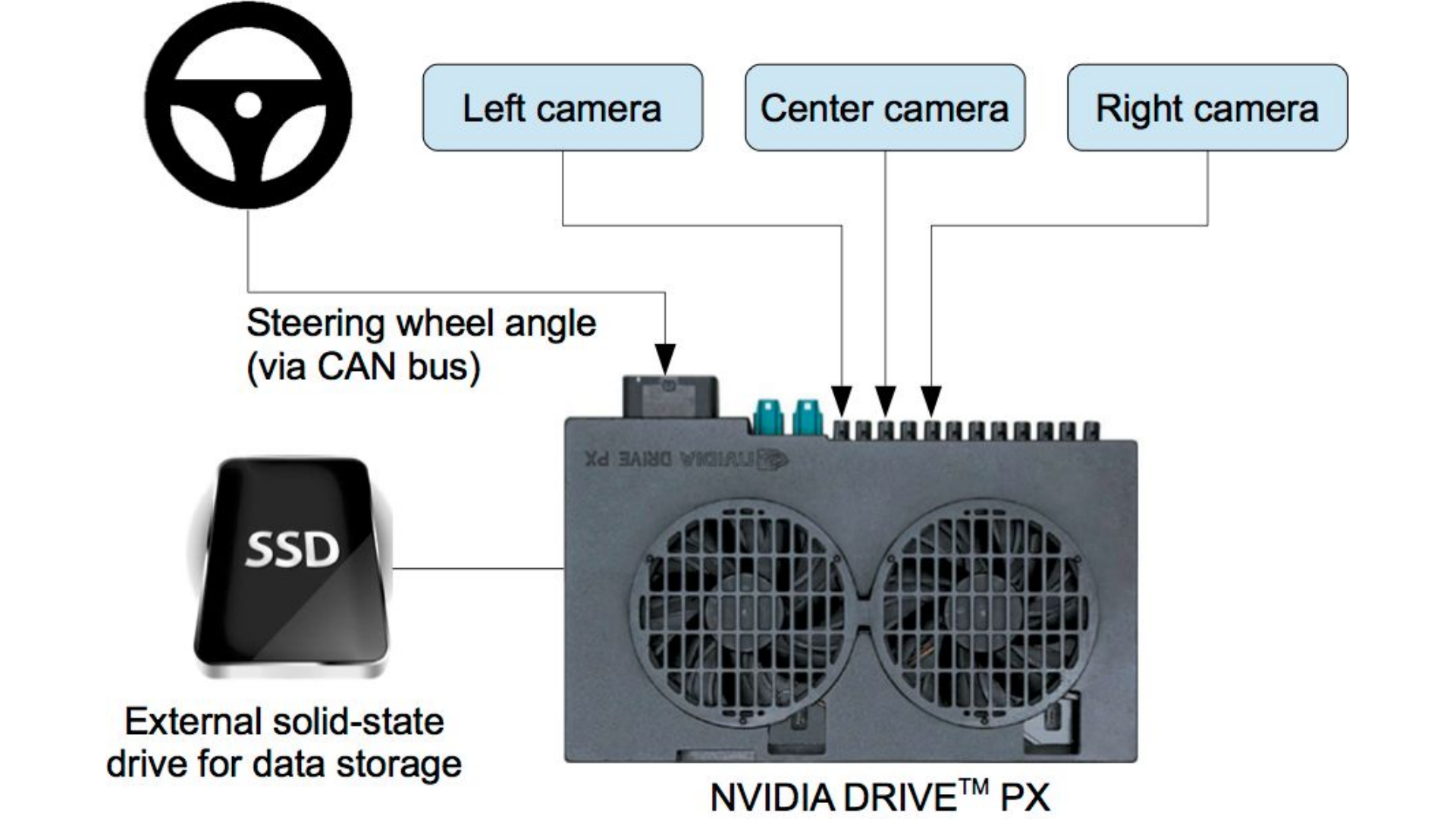
Źródło: https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/disengagement_report_2016

Uproszczony schemat



WHAT DO WE WANT?

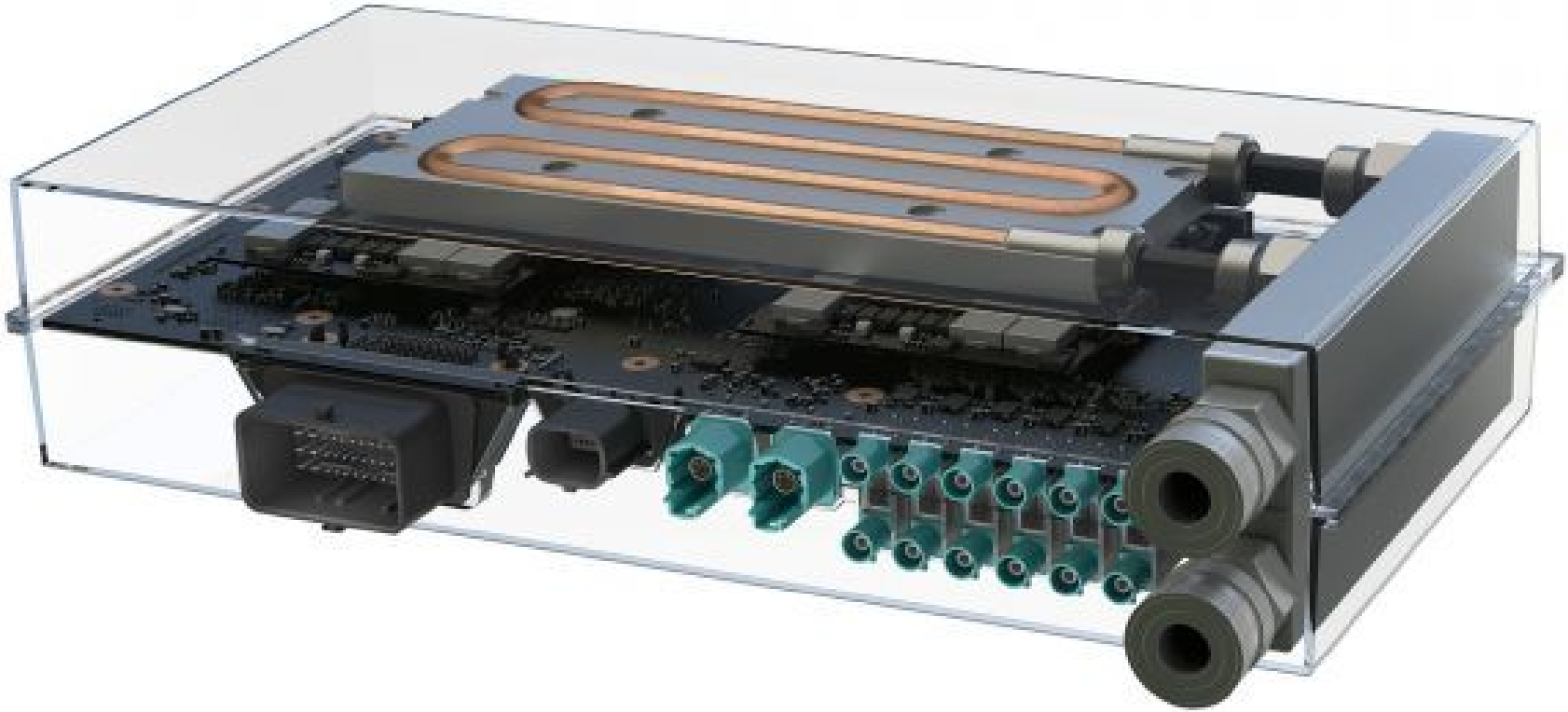
MORE DETAILS!!!


Left camera
Center camera
Right camera
Steering wheel angle
(via CAN bus)
SSD
External solid-state
drive for data storage
NVIDIA DRIVE™ PX

NVIDIA DRIVE PX 2

Rozmiary…, chłodzenie,
PS4 ~4.5 TFLOPS,

	TITAN X	DRIVE PX 2
Process	28nm	16nm FinFET
CPU	–	12 CPU cores 8-core A57 + 4-core Denver
GPU	Maxwell	Pascal
TFLOPS	7	8
DL TOPS	7	24
AlexNet	450 images / sec	2,800 images / sec



NVIDIA.


NVIDIA
BB8 learns how to drive by watching us

W oparciu o analizę obrazu

To robi Tesla i wszyscy na ich ogonie + NVIDIA Driveworks

Składa się z wielu osobnych systemów spinanych w jeden:

- Zasady ruchu drogowego
- Pozyskanie istotnych informacji z obrazu
- Dashboard ze aktualnym stanem pojazdu
- Reakcja na ruch innych
- Próba przewidywania przyszłości
- Siatka wolnego terenu

3 sieci neuronowe

W skrócie:
No..no...no...found, człowiek tak nie zrobi

Behavioral cloning

BB8

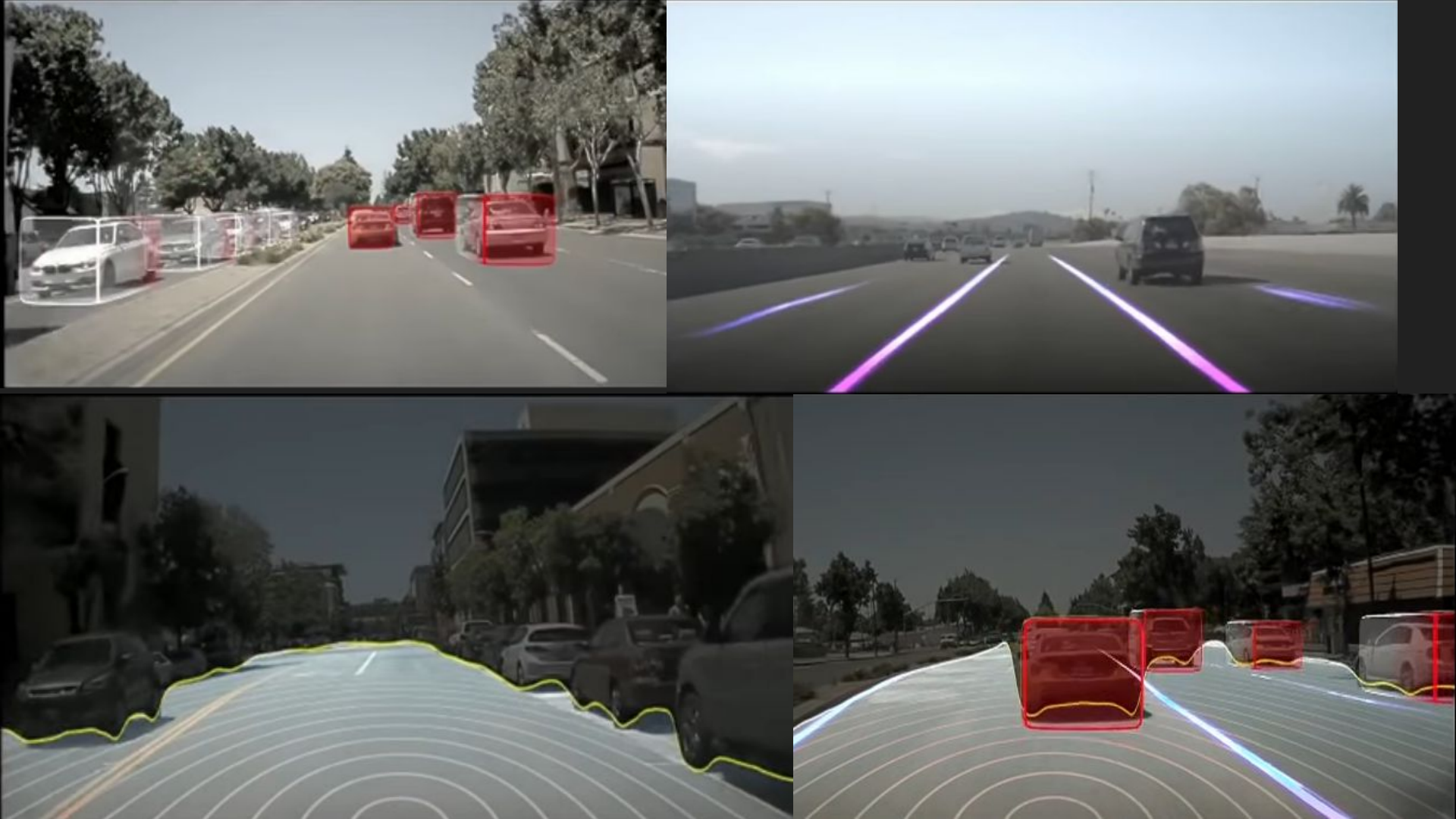
- Nie projektujemy logiki
- Nie pokazujemy na co zwracać uwagę podczas jazdy
- Uczymy przez kopiowanie naszych zachowań i samemu dochodzenie do schematu


NVIDIA
and cruises right through

Korzyści:

- Umiejętność odnalazenia się w nietypowych warunkach
  (brak pasów, przeszkody ruchome, trawa zamiast pobocza, nierówna droga)

- Mniej do programowania, taniej



Aktywność neuronów podczas jazdy NVIDIA BB8

Aktywność neuronów podczas jazdy NVIDIA BB8


I SHOULD
GET A SELF-DRIVING CAR
memegenerator.net

Nie ma pieniędzy na auto?


NO
ROLLER SKATI
OR
SKATEBOARDIN
IN THIS AREA


ROUTE 66 STORE
TESLA
CGimodels

Krótka historia długiego projektu

Kanał YT i Twitch: sentdex

Karta graficzna: GTX TITAN X 12GB

Biblioteki: opencv, Imagegrab

Python, AlexNet jako model

python plays

14. In the city

Pierwsze kroki

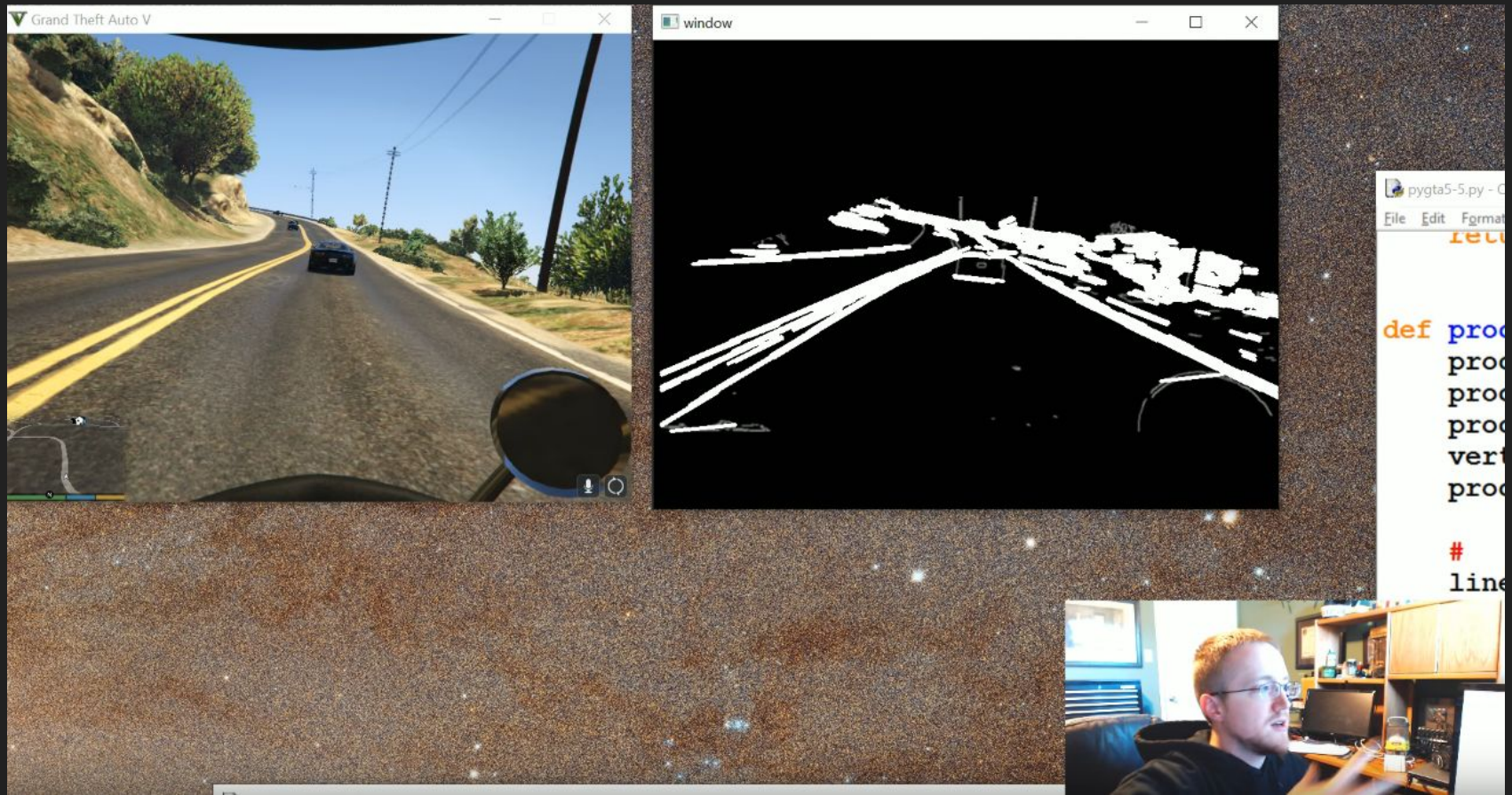

YEAH. I'M GONNA
NEED MORE DETAILS.

Krótka historia długiego projektu

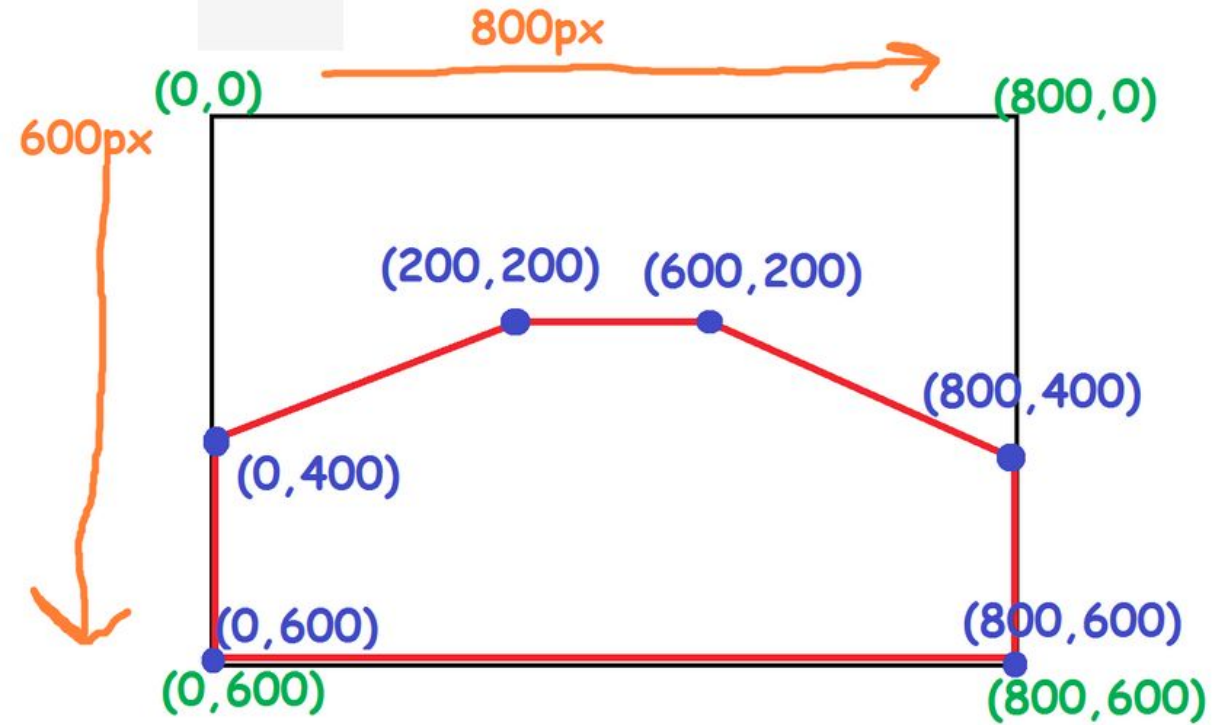
```
30  import numpy as np              # na dłuższą metę must-have
31                                  # przydaje się do macierzy -> operacja, przechowywanie
32  import pyscreenshot as ImageGrab        # from PIL import ImageGrab na Windows i macOS
33                                          # na Linuxa pyscreenshot
34  import cv2                              # opencv2 do wychwytywania wzorców na zdjęciach
35  import time                             # mierzenie czasu
36
37
38  def process_img(image):
39      original_image = image
40      processed_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # konwersja w odcienie szarości
41      processed_img =  cv2.Canny(processed_img, threshold1 = 200, threshold2=300) # proste wykrywanie krawędzi
42      return processed_img
43
44  def main():
45      last_time = time.time()                         # zapamiętujemy czas
46      while True:
47          # start (x = 0,y = 40), koniec (x = 800, y = 640)
48          screen =  np.array(ImageGrab.grab(bbox=(0,40,800,640)))
49          last_time = time.time()
50          screen_in_greyscale = process_img(screen)
51          cv2.imshow('window', screen_in_greyscale)   # wyświetl w oknie
52          if cv2.waitKey(25) & 0xFF == ord('q'):      # wyjście na klawisz
53              cv2.destroyAllWindows()
54              break
55
```

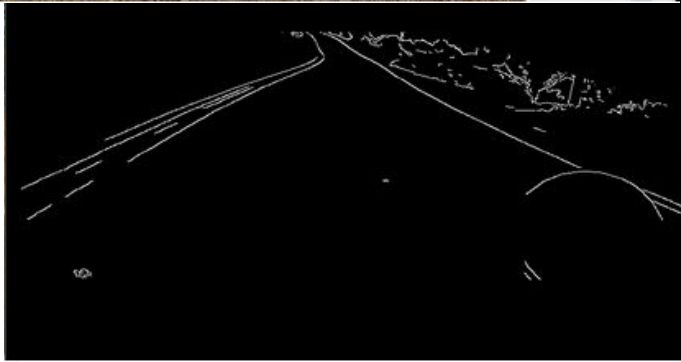
```
import numpy as np              # na dłuższą metę must-have
                                # przydaje się do macierzy -> operacja, przechowywanie
import pyscreenshot as ImageGrab        # from PIL import ImageGrab na Windows i macOS
                                        # na Linuxa pyscreenshot
import cv2                              # opencv2 do wychwytywania wzorców na zdjęciach
import time                             # mierzenie czasu
from directkeys import PressKey, W, A, S, D
# ^^
def process_img(image):
    original_image = image
    processed_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # konwersja w odcienie szarości
    processed_img =  cv2.Canny(processed_img, threshold1 = 200, threshold2=300) # proste wykrywanie krawędzi
    return processed_img

def main():

    last_time = time.time()                         # zapamiętujemy czas
    while True:
        PressKey(W) # <----------------------
        # start (x = 0,y = 40), koniec (x = 800, y = 640)
        screen =  np.array(ImageGrab.grab(bbox=(0,40,800,640)))
        last_time = time.time()
        screen_in_greyscale = process_img(screen)
        cv2.imshow('window', screen_in_greyscale)   # wyświetl w oknie
        if cv2.waitKey(25) & 0xFF == ord('q'):      # wyjście na klawisz
            cv2.destroyAllWindows()
            break

```

Poprawione wartości threshold

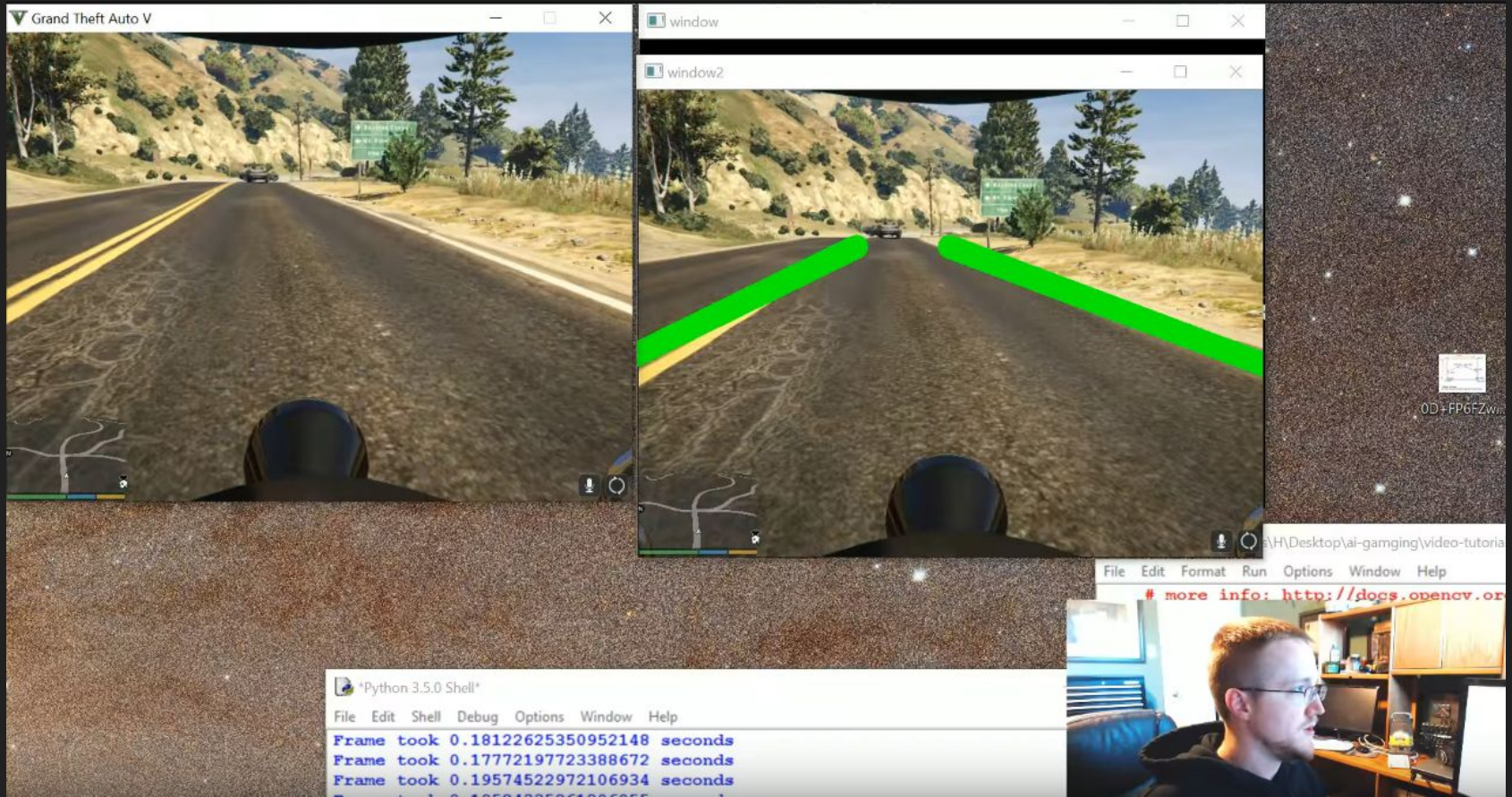


```python
def roi(img, vertices):
    mask = np.zeros_like(img)     # pusta maska
    cv2.fillPoly(mask, vertices, 255) # obszar wypelniony 1
    masked = cv2.bitwise_and(img, mask) # obraz AND maska = ograniczone pole widzenia
    return masked

```





Efekt końcowy z opencv



python plays

is.

stream

Samemu do spróbowania

ai-tor / **DeepGTAV**

Watch 32 | Star 250 | Fork 67

Code | Issues 10 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

A plugin for GTAV that transforms it into a vision-based self-driving car research environment.

dataset-generation | reinforcement-learning | gtav | deep-learning | self-driving-car

116 commits | **1** branch | **0** releases | **3** contributors | GPL-3.0

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

ai-tor committed on **GitHub** Merge pull request #40 from dtmoodie/bbox ··· Latest commit 22db0c3 on 20 Apr

Rewarders	Improved code readability	5 months ago
bin	updated binary.	a month ago
lib	DeepGTAV v2	3 months ago
.gitignore	Updated gitignore and deleted old .asi file	6 months ago
DeepGTAV.filters	Reorganized directories, uploaded full VS project	6 months ago
DeepGTAV.sdf	Reorganized directories, uploaded full VS project	6 months ago
DeepGTAV.v12.suo	Reorganized directories, uploaded full VS project	6 months ago
DeepGTAV.vcxproj	DeepGTAV v2	3 months ago


```
{"start": {
  "scenario": {
    "location": [1015.6, 736.8],
    "time": [22, null],
    "weather": "RAIN",
    "vehicle": null,
    "drivingMode": [1074528293, 15.0]
  },
  "dataset": {
    "rate": 20,
    "frame": [227, 227],
    "vehicles": true,
    "peds": false,
    "trafficSigns": null,
    "direction": [1234.8, 354.3, 0],
    "reward": [15.0, 0.5],
    "throttle": true,
    "brake": true,
    "steering": true,
    "speed": null,
    "yawRate": false,
    "drivingMode": null,
    "location": null,
    "time": false
  }
}}
```

michal.martyniak@linux.pl
GradientPG

github.com/micmarty

Linki do ciekawych artykułów i filmów:

goo.gl/MQ0osj