Sistem Segmentasi Jalan dan Objek untuk Kendaraan Otonom Menggunakan Kamera RGB-D

Aldy Helnawan, Muhammad Attamimi, dan Astria Nur Irfansyah
Departemen Teknik Elektro, Institut Teknologi Sepuluh Nopember (ITS)
E-mail: attamimi@ee.its.ac.id

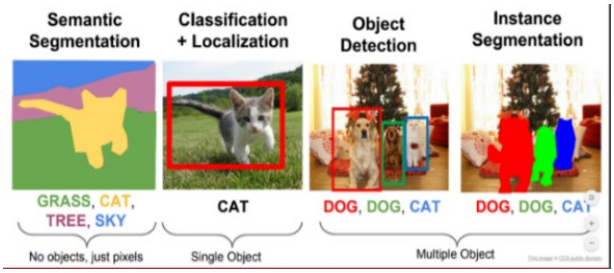
Abstrak*—Kendaraan roda empat yang memiliki kemampuan untuk melakukan perjalanan antar titik tanpa adanya operator manusia yang dimana menggunakan kombinasi antar sensor, kamera, radar, dan kecerdasan buatan (AI). Penggunaan kamera RGBDNIR (Red Green Blue *Depth* Near Infrared) dengan alat yaitu kamera Intel RealSense D435i yang dapat digunakan baik didalam ataupun diluar ruangan dimana sensor modul *depth* dan NIR dapat digunakan ketika keadaan kurang pencahayaan atau lingkungan redup. Dilakukannya penelitian ini dikarenakan untuk mencari solusi dari tingginya tingkat kecelakaan di jalan serta mencari solusi atas kelemahannya kamera RGB dalam penangkapan citra untuk yang dipadukan dengan penggunaan *machine learning* untuk pengambilan keputusan dalam menentukan kelas objek yang terdeteksi dan diproses untuk menghasilkan solusi dalam melakukan segmentasi di lingkungan terbuka (luar ruangan). Untuk perangkat lunak pemprograman yang akan digunakan yaitu Python serta pustaka yang akan digunakan antara lain PyTorch, OpenCV, dan TensorFlow dengan alat komputasi berupa laptop yang memiliki GPU Nvidia RTX 3060 atau sejenisnya. Hasil dari penelitian ini berupa gambar segmentasi dan pengenalan kelas objek yang terdeteksi dengan tingkat keakuratan dengan beberapa model mulai dari 39.60% hingga 63.71% yang dapat digunakan untuk penentuan kelas yang terbaca. Dengan tingkat literasi beragam sampai nilai terkecil 2E-10 dan memiliki waktu pemprosesan untuk setiap citra dari 0.22 detik sampai 0.01 detik.

***Kata Kunci*—Kendaraan Otonom, Kamera Stereo, *Segmentation*, *Computer Vision*.**

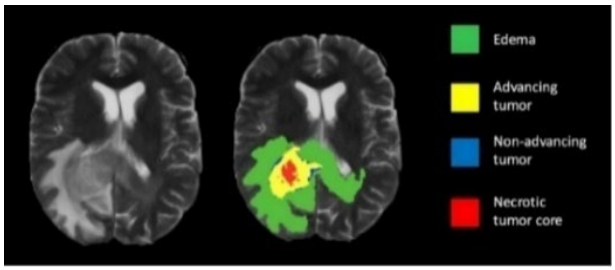
I. PENDAHULUAN

PADA dasarnya, perkembangan teknologi didasari atas kebutuhan masyarakat yang menginginkan solusi dan kemajua teknologi. Pada era industri 4.0 ini, perkembangan teknologi semakin maju ini mendorong masyarakat untuk menciptakan inovasi dengan memperbanyak peran mesin dalam sebuah pekerjaan. Dalam mobilisasi membutuhkan moda transportasi yang memadai denga naman dan nyaman digunakan, sehingga dapat menghasilkan solusi seperti sistem lalu lintas yang dapat memperkecil kemungkinan adanya kecelakaan dalam berkendara. Tetapi kenyataannya tetap terjadinya kecelakaan dikarenakan berbagai faktor.

Pada masalah tersebut, dapat diberikan sebuah solusi yaitu dengan menerapkan penggunaan *machine learning* pada kendaraan otonom dengan menggunakan *machine vision* dengan *object detection* yang dapat menangkap informasi dari citra visual, dimana kendaraan tersebut dapat mengambil keputusan sesuai dengan parameter sebagai batasan dalam berkendara dimana terdapat peraturan lalu lintas yang perlu dipatuhi agar dapat aman serta nyaman dalam berkendara. Pengguanaan *machine learning* tersebut bertujuan untuk meningkatkan kapabilitas kendaraan agar dapat berjalan otomatis lebih akurat walaupun minimnya marka jalan yang



Gambar 1. Perbedaan Antara *Semantic* dan *Instance Segmentation*.



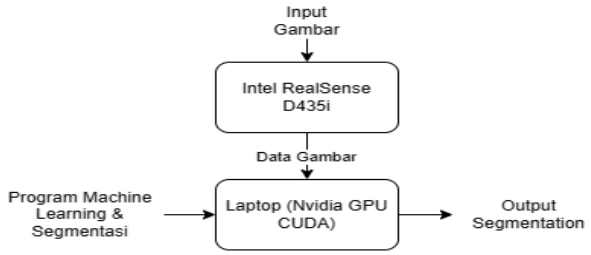
Gambar 2. Perbedaan Segmentasi Gambar Dalam Bidang Medis.

terdapat di beberapa ruas jalan mengingat tidak semua jalan umum memiliki marka jalan yang lengkap. Untuk metode *machine learning* yang digunakan merupakan *supervised machine learning* dimana sistem tersebut diberikan parameter atau batasan agar tetap berkendara sesuai dengan regulasi atau peraturan yang berlaku. Dengan menggunaan *machine vision* serta *object detection* untuk mendeteksi seperti marka jalan, rambu lalu lintas, dan objek lainnya di jalan. Lalu dengan terdapat masalah mengenai penggunaan sensor untuk mendapatkan citra digital yang akan dijadikan data, yaitu sensor RGB, sensor *depth*, dan juga sensor NIR. Dari kegita tersebut, terdapat kelebihan dan juga kekurangan yang perlu diperhatikan untuk memiliki hasil yang mumpuni dengan tingkat keakuratan yang cukup tinggi untuk meminimalisir adanya *loss*.

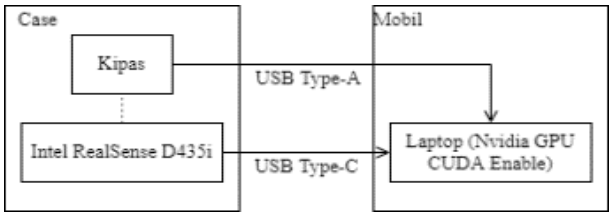
Tujuan dari tugas akhir ini yaitu membuat sebuah program *machine learning* dengan penggunaan *object detection* untuk mendeteksi jalan atau objek lainnya dengan menggunakan *image segmentation* dengan alat berupa komputer skala kecil atau laptop untuk pengolahan gambar dengan mengharapkan meningkatkan sistem akurasi dari sistem pendeteksi dan segmentasi dengan *deep neural network*.

Pengembangan Sistem Segmentasi Jalan dan Objek Untuk Kendaraan Otonom Menggunakan Kamera RGB-D ini diharapkan dapat membantu perkembangan dari mobil otonom dan dapat diaplikasikan dalam berbagai jenis riset/penelitian dan kebutuhan lainnya. Alat ini dapat membantu menyelesaikan masalah dalam bidang segmentasi visual komputer untuk kendaraan menggunakan kamera RGB-D. Alat ini memiliki keunggulan yaitu dapat digunakannya kamera *infrared* dan *depth* sehingga dapat

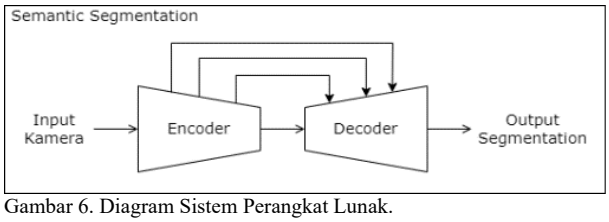
Gambar 3. Kamera Intel RealSense D435i.



Gambar 4. Diagram Blok Sistem.



Gambar 5. Diagram Sistem Pengait.



Gambar 6. Diagram Sistem Perangkat Lunak.

memungkinkan komputer untuk melihat pandangan yang tidak dapat dilihat oleh kamera RGB, dimana alat ini menggunakan kamera Intel RealSense D435i yang memiliki modul RGB, *depth*, dan NIR.

II. TINJAUAN PUSTAKA

A. *Pengolahan Citra Digital*

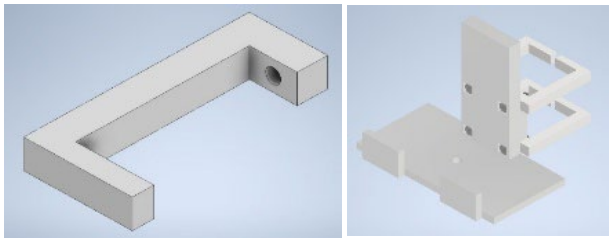
Pengolahan citra digital merupakan salah satu proses dalam pengambilan data untuk mesin atau komputer dalam melakukan perhitungan logika dan matematis yang baanyak dimanfaatkan baik dalam lingkup masyarakat umum, perusahaan, maupun penelitian. Pengolahan citra digital digunakan agar dapat menyajikan citra tersebut lebih informatif, dimana data tersebut dapat meningkatkan kualitas dari hasil suatu produk, melakukan *editing*, dan lain sebagainya [1].

B. *Image Segmentation*

Image Segmentation atau segmentasi gambar adalah proses partisi gambar menjadi beberapa segmen, yang biasanya digunakan untuk menemukan objek dan batas dalam gambar. Atau dapat dikatakan *Image Segmentation* merupakan proses dimana gambar digital dipartisi menjadi beberapa subkelompok (*pixel*) yang disebut objek gambar, yang dapat mengurangi kompleksitas gambar, dan dengan demikian menganalisis gambar menjadi lebih sederhana. Hasil dari segmentasi gambar adalah satu set segmen yang secara kolektif mencakup seluruh gambar, atau set kontur yang diekstrak dari gambar, dimana masing-masing *pixel* di suatu wilayah mirip sehubungan dengan beberapa karakteristik atau properti yang dihitung, seperti warna, intensitas, atau tekstur. *Image segmentation* dibagi menajdi dua, yaitu *semantic segmentation* dan *instance segmentation* seperti yang ditunjukkan pada Gambar 1.

Semantic segmentation adalah proses segmentasi *pixel* gambar ke dalam kelas masing-masing yang akan dibedakan dari kelas lainnya. Misal, pada gambar kucing di Gambar 1, merupakan contoh dari segmentasi semantic. *Instance segmentation* atau segmentasi instans merupakan segmentasi dimana menjadi lebih menyeluruh dan biasanya muncul dalam gambar ketika berhadapan dengan beberapa objek. Perbedaannya adalah objek yang terdeteksi ditutup dengan warna sehingga semua pixel yang terkait dengan gambar diberi warna yang sama seperti yang ditunjukkan pada Gambar 2. Dan beberapa objek dari kelas yang sama diperlakukan sebagai entitas yang berbeda akan diwakili dengan warna yang berbeda.

Gambar 7. Desain *Case*/Cangkang Kamera.



Gambar 8. (Kiri) Desain Capit, (Kanan) Desain *Mounting Case* Kamera ke Mobil.

Tabel 1.
Spesifikasi Asus FX506HM.

Operating System	Windows 10 Home 64-Bit
Processor	Intel Core i7-1180H Processor 2.3GHz (24M Cache, up to 4.6 GHz, 8 Cores)
Graphics	NVIDIA GeForce RTX 3060 Laptop GPU, up to 1630 MHz at 90W (95W with Dynamic Boost), 6GB GDDR6
Display	15.6-inch, FHD (1920 x 1080) 16:9, anti-glare display, sRGB:62.5%, Adobe:47.34%, Refresh Rate:144Hz
Memory	24GB DDR4 SO-DIMM (3200MHz)
Storage	1.5TB M.2 NVMe PCIe 3.0 SSD
Battery	90Wh
Power Supply	ø6.0, 200W AC Adapter, Output: 20V DC, 10A, 200W, Input: 100-240V AC, 50/60Hz universal

Gambar 9. Perancangan Elektronik.

Tabel 2.
Jenis Kelas dan Label Warna.

No	Kelas Objek	Kode Warna (R,G,B)	Kode Warna #HEX	Visual Warna
1	*void*	0,0,0	#000000	
2	Jalan	128,64,128	#804080	
3	Trotoar	244,35,232	#F423E8	
4	Gedung	70,70,70	#464646	
5	Tembok	102,102,156	#66669C	
6	Pagar	190,153,153	#BE9999	
7	Tiang	153,153,153	#999999	
8	Lampu Lalu Lintas	250,170,30	#FAAA1E	
9	Rambu Lalu Lintas	220,220,0	#DCDC00	
10	Vegetasi	107,142,35	#6B8E23	
11	Medan Tanah	152,251,152	#98FB98	
12	Langit	70,130,180	#4682B4	
13	Manusia	220,20,60	#DC143C	
14	Pengendara	255,0,0	#FF0000	
15	Mobil	0,0,142	#00008E	
16	Truk	0,0,70	#000046	
17	Bus	0,60,100	#003C64	
18	Kereta	0,80,100	#005064	
19	Sepeda Motor	0,0,230	#0000E6	
20	Sepeda	119,11,32	#770B20	

Gambar 10. Gambar Cuplikan *Ground Truth* Anotasi *Semantic Segmentaion* dari Jalan di ITS.

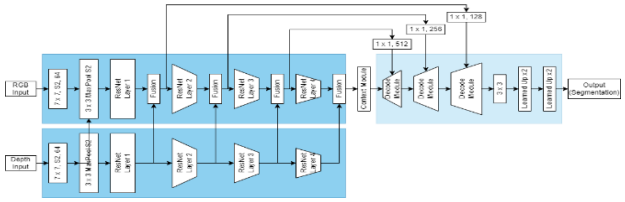
Penggunaan dari segmentasi gambar dapat digunakan pada mobil otonom untuk memberikan perbedaan yang mudah antara berbagai objek, seperti sinyal lalu lintas, papan nama, manusia, hewan, dan kendaraan lainnya. Dapat juga digunakan untuk deteksi cacat pada papan sirkuit dimana untuk mendeteksi penempatan komponen sesuai dengan standar atau terdapat cacat yang dihasilkan dalam produksi. Pendeteksian wajah dan pencitraan medis, dimana penggunaan untuk membedakan setiap wajah dan juga untuk mengekstrak informasi yang relevan secara klinis seperti mengelompokan tumor.

C. *Machine Learning*

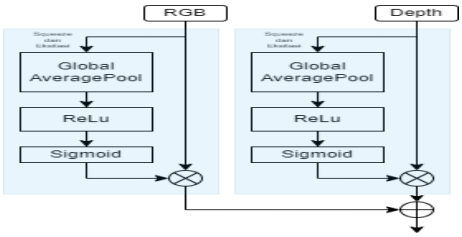
Menurut sumber dari Azure – Microsoft, *machine learning* adalah proses menggunakan model matematika data untuk membantu komputer belajar tanpa instruksi langsung. Hal tersebut dianggap sebagai bagian dari kecerdasan buatan (AI atau *Artificial Intellegence*). *Machine Learning* menggunakan algoritma untuk mengindentifikasi pola dalam data, dan pola-pola tersebut kemudian digunakan untuk membuat model data yang dapat membuat prediksi. Dengan peningkatan data dan juga pengalaman, hasil pembelajaran mesin lebih akurat, seperti bagaimana meningkat dengan lebih banyak latihan. Kemampuan beradaptasi pembelajaran mesin ini menjadikan pilihan yang bagus dalam skenario dimana data selalu berubah, sifat permintaan atau tugas selalu

```
#!/usr/bin/python
# python imports
from __future__ import print_function, absolute_import, division
import os, glob, sys

# cityscapes imports
from cityscapesscripts.helpers.csHelpers import printError
from cityscapesscripts.preparation.json2labelImg import json2labelImg

# The main method
def main():
    # Where to look for Cityscapes
    if 'CITYSCAPES_DATASET' in os.environ:
        cityscapesPath = os.environ['CITYSCAPES_DATASET']
    else:
        cityscapesPath = os.path.dirname("raw/datasets/cityscapes")

    # how to search for all ground truth
    searchFine   = os.path.join( cityscapesPath , "gtFine"   , "*" , "*" ,
"*_gt*_polygons.json" )
    searchCoarse = os.path.join( cityscapesPath , "gtCoarse" , "*" , "*" ,
"*_gt*_polygons.json" )
    # search files
    filesFine = glob.glob( searchFine )
    filesFine.sort()
    filesCoarse = glob.glob( searchCoarse )
    filesCoarse.sort()

    # concatenate fine and coarse
    files = filesFine + filesCoarse

    # quit if we did not find anything
    if not files:
        printError( "Did not find any files. Please consult the README." )

    # a bit verbose
    print("Processing {} annotation files".format(len(files)))

    # iterate through files
    progress = 0
    print("Progress: {:>3} %".format( progress * 100 / len(files) ), end='
')
    for f in files:
        # create the output filename
        dst = f.replace( "_polygons.json" , "_labelTrainIds.png" )

        # do the conversion
        try:
            json2labelImg( f, dst, "trainIds" )
        # do the conversion
        try:
            json2labelImg( f, dst, "trainIds" )
        # do the conversion
        try:
            json2labelImg( f, dst, "trainIds" )
end=' ')
        sys.stdout.flush()

if __name__ == "__main__":
    main()
```

Gambar 11. Rancangan Program Dataset.



Gambar 12. Rancangan Program RGBD.



Gambar 13. Rancangan Program Fusion RGBD (Kiri).

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{}{}$$

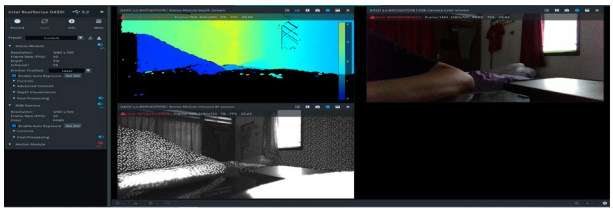
Gambar 14. Ilustrasi IoU.

bergeser, atau pengkodean solusi akan secara efektif tidak mungkin.

D. *Intel RealSense D435i*

Intel RelaSense D435i merupakan kamera *depth* modul yang mencakup bidang sensor *depth* tampilan yang lebih luas dan sensor RGB yang dibuat oleh perusahaan Intel. Kamera ini dapat disambungkan dengan komputer atau divais komputasi lainnya dengan menggunakan kabel USB tipe C. Kamera Intel RealSense ini juga memiliki spesifikasi seperti resolusi 1280 x 720, modul RGB dan Depth dapat mencapai

90 fps, dan modul NIR mencapai 30 fps, serta jarak ideal 0.3 m hingga 3 m seperti yang ditunjukkan pada Gambar 3.



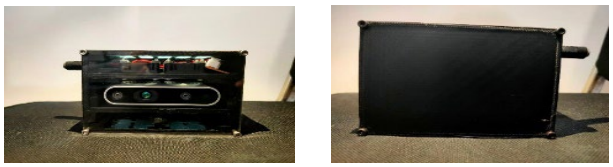
Gambar 15. Tampilan Aplikasi Intel RealSense Viewer.

```
pip install pyrealsense2
```

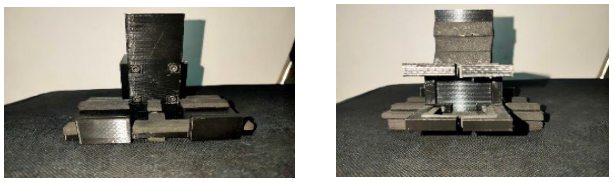
Gambar 16. Perintah Pip yang digunakan untuk pemasangan dari pusrtaka.

.

Gambar 17. Tampilan kamera RGB (kiri), *Depth* (tengah), NIR (kanan).



Gambar 18. *Case* tampak depan (kiri) dan belakang (kanan).



Gambar 19. Pengait (mount) tampak depan (kiri) dan belakang (kanan).

E. PyTorch

Pytorch merupakan kerangka kerja *machine learning* yang bersifat *open source* berdasarkan perpustakaan Torch yang digunakan untuk pengaplikasian pada bidang *computer vision*, *natural language processing*. Pytorch dikembangkan oleh Meta AI, sebuah divisi perusahaan Meta (yang dulunya bernama Facebook) yang merupakan sebuah perusahaan teknologi asal Amerika Serikat yang dapat digunakan dalam berbagai bahasa pemprograman, seperti Python, dan C++.

III. METODOLOGI

A. Diagram Blok Sistem

Untuk blok diagram pada Gambar 4 perangkat keras pada bagian penangkapan citra kamera terdiri dari kamera Intel RealSense D435i, *casing*/cangkang, serta pengait untuk menempatkan kamera di depan kendaraan/mobil. Untuk kendaraan yang digunakan pada saat pengumpulan gambar menggunakan Toyota Agya 1.0 G M/T yang dapat dilihat pada diagram blok yang ditunjukkan pada Gambar 5.

Sementara untuk perangkat lunak terdiri dari input kamera, *encoder*, *decoder*, dan *output* berupa hasil segmentasi yang dapat dilihat pada Gambar 6.

B. Rancangan Perangkat Mekanis

Pada perangkat keras mekanis ini atau *hardware* yang pertama akan dibahas yaitu sistem pengait serta *case* dari kamera ke mobil atau kendaraan. Untuk cangkang/*case* ini dibuat menggunakan print 3D berbahan PLA+ atau *polyactic*

```
Processing 5000 annotation files
Progress: 31.48 %
```

Gambar 20. Tampilan Persiapan Dataset.

Name	Date modified	Type	Size
test	12/6/2022 9:12 PM	File folder	
train	12/6/2022 9:26 PM	File folder	
valid	12/6/2022 9:34 PM	File folder	
class_colors_1+19.txt	12/1/2022 11:36 PM	Text Document	1 KB
class_colors_1+33.txt	12/1/2022 11:36 PM	Text Document	1 KB
class_names_1+19.txt	12/1/2022 11:36 PM	Text Document	1 KB
class_names_1+33.txt	12/1/2022 11:36 PM	Text Document	1 KB
test_depth_raw.txt	12/1/2022 11:36 PM	Text Document	58 KB
test_disparity_raw.txt	12/1/2022 11:36 PM	Text Document	64 KB
test_labels_19.txt	12/1/2022 11:36 PM	Text Document	73 KB
test_labels_33.txt	12/1/2022 11:36 PM	Text Document	73 KB
test_rgb.txt	12/1/2022 11:36 PM	Text Document	67 KB
train_depth_raw.txt	12/1/2022 11:36 PM	Text Document	120 KB
train_disparity_raw.txt	12/1/2022 11:36 PM	Text Document	131 KB
train_labels_19.txt	12/1/2022 11:36 PM	Text Document	149 KB
train_labels_33.txt	12/1/2022 11:36 PM	Text Document	149 KB
train_rgb.txt	12/1/2022 11:36 PM	Text Document	137 KB
valid_depth_raw.txt	12/1/2022 11:36 PM	Text Document	21 KB
valid_disparity_raw.txt	12/1/2022 11:36 PM	Text Document	23 KB
valid_labels_19.txt	12/1/2022 11:36 PM	Text Document	26 KB
valid_labels_33.txt	12/1/2022 11:36 PM	Text Document	26 KB
valid_rgb.txt	12/1/2022 11:36 PM	Text Document	24 KB

Gambar 21. Tampilan Hasil Persiapan *Dataset.*

Nama Model	Parameter	Seed	Final Epoch	Loss (Tr)	mIoU	lr	Time (Total)
CGNet	496306	1234	99	1.0389	0.5888	0.0000079	9 Jam 30 Menit
DABNet	756643	1234	99	1.0393	0.6371	0.0000079	8 Jam 52 Menit
EDANet	689485	1234	99	1.054	0.6241	0.0000079	11 Jam 32 Menit
ENet	360422	1234	99	1.2094	0.4341	0.0000079	-
ESPNet	201542	1234	99	1.1782	0.464	0.0000079	7 Jam 55 Menit
FastSCNN	1138051	1234	99	1.0546	0.5553	0.0000079	7 Jam 9 Menit
FSSNet	175881	1234	99	1.191	0.396	0.0000079	8 Jam 11 Menit
LinkNet	11535699	1234	99	1.099	0.5155	0.0000079	7 Jam 45 Menit
SegNet	29453971	1234	99	1.117	0.4588	0.0000079	-
SQNet	16262771	1234	99	1.0559	0.5408	0.0000079	17 Jam 51 Menit
UNet	14790099	1234	299	0.9604	0.5518	0.0000079	-
RGBD	-	-	99	0.553	0.5874	2.00E-10	20 Jam 33 Menit

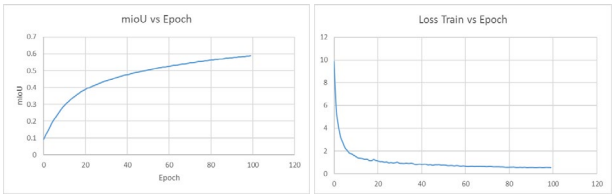
Gambar 22. Hasil *Train* dan Validasi Setiap Model.

acid plus merupakan variasi dari bahan PLA atau *polyactic acid* yang telah ditingkatkan kualitasnya dengan menambahkan beberapa zat adiktif. Desain dari *case* ini memiliki ketebalan bahan sebesar 5 mm dengan penutup dengan menggunakan papan akrilik berukuran 115 × 85 mm (*panjang* × *lebar*) yang memiliki ketebalan sebesar 3 mm. Untuk penghubung antara *case* dengan sistem pengait/*mounting* digunakannya baut M6. Pada *case* ini juga terdapat lubang *exhaust* untuk sirkulasi udara dengan ditambahkannya kipas didalamnya agar kamera dapat terhindar dari *overheat*. Lalu juga dipasang filter pada lubang *exhaust* bertujuan agar debu/kotoran dari lingkungan di jalan tidak mengotori kamera yang berada di dalam *case*. Untuk desain dari *case* tersebut dapat dilihat pada Gambar 7.

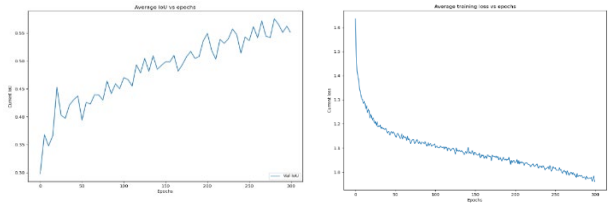
Untuk bagian pengait atau *mounting* dari *case* kamera tersebut memiliki ketebalan bervariasi. Untuk bagian dudukan dan penghalang depan memiliki ketebalan 5 mm dengan bagian dudukan ditambahkan busa dengan ketebalan 5 mm. Tujuan ditambahkannya busa tersebut sebagai peredam getaran yang berasal dari getaran kendaraan baik saat keadaan diam ataupun melaju di jalan. Untuk bagian penghalang belakang memiliki ukuran 40 × 85 mm (*lebar* × *tinggi*) dengan ketebalan 10 mm sebagai penghubung antara pengait dengan dudukan *case*. Untuk bagian pengait memiliki ukuran area sebesar 50 × 50 mm dengan bentuk pengait seperti capit yang dapat dilepas-pasang dengan mur dan baut M3 pada bagian penghalang belakang. Untuk bagian capit juga digunakan busa sebagai peredam getaran. Untuk desain dari mount dapat dilihat pada Gambar 8.

C. Rancangan Perangkat Elektronik

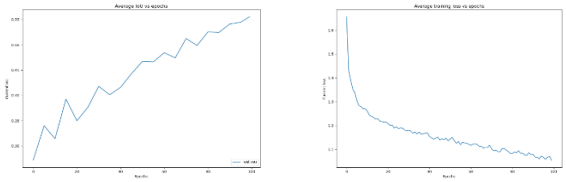
Untuk perangkat elektronik ini yang menjadi pusat (otak)



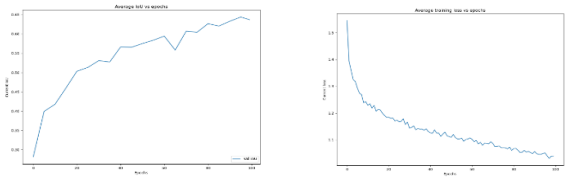
Gambar 23. Grafik mIou dan *Loss* RGBD.



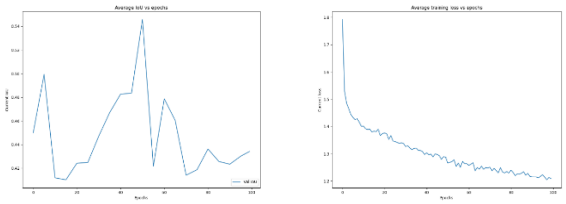
Gambar 24. Grafik mIoU dan *Loss* Unet.



Gambar 25. Grafik mIoU dan *Loss* FastSCNN.



Gambar 26. Grafik mIoU dan *Loss* DABNet.



Gambar 27. Grafik mIoU dan *Loss* Enet.

dari sistem ini adalah laptop Asus F15 FX506HM. Untuk spesifikasi dari laptop tersebut yang bersumber dari laman asus.com dapat dilihat pada Tabel 1.

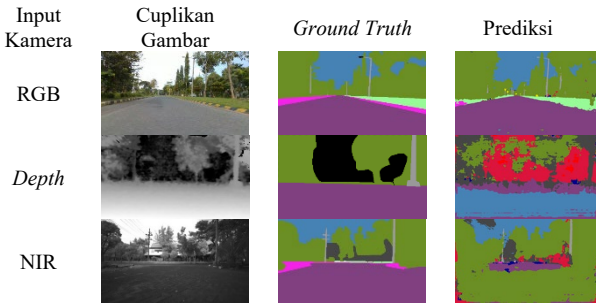
Selain laptop, terdapat komponen perangkat keras lainnya yang dibutuhkan dalam penelitian ini yang dapat dilihat sebagai berikut:

1) Intel RealSense D435i *Stereo Camera*
2) *Case* dan *Mounting System* dari *Print* 3D dengan bahan PLA+
3) Kabel USB 3.1 Gen 1 Type C (minimal)
4) Kabel USB 3.0 Type-A to DC jack 2.5 mm
5) Papan Akrilik sebagai penutup depan *case*
6) Kipas DC 30 mm 5V (2 buah)

Untuk diagram perancangan elektronik secara keseluruhan dapat dilihat pada Gambar 9.

D. Perancangan Dataset

Dengan menggunakan pustaka cityscapesScripts yang telah disediakan, kita dapat menggunakannya untuk kebutuhan apa saja yang diperlukan, seperti menambahkan inisiasi untuk menambahkan *dataset disparity* yang akan dikonversi menjadi *depth image*. Atau dapat menginisiasi



Gambar 28. Cuplikan gambar jalan di ITS (atas) dan hasil segmentasi (bawah) dengan Unet.

Tabel 3.
Tabel waktu dan *Frame Rate.*

Model	t (detik)	*Frame rate* (fps)
CGNet	0.02	50
DABNet	0.02	50
EDANet	0.02	50
ENet	0.02	50
ESPNet	0.01	100
FastSCNN	0.01	100
FSSNet	0.02	50
LinkNet	0.02	50
SegNet	0.22	4.545454545
SQNet	0.08	12.5
UNet	0.12	8.333333333
RGBD	0.128	7.8125

label-label yang akan digunakan, lalu menginisiasikan kelas yang akan dikonversikan dengan sisa kelas yang tidak digunakan dianggap menjadi null (misal 19 kelas + 1 *null*).

Untuk kelasnya terdapat total 20 kelas dengan rincian 19 kelas dengan tambahan 1 kelas *void* seperti pada Tabel 2. Kelas *void* ini dimaksud apabila objek yang tertangkap citra tidak dapat didefinisikan oleh kelas yang lain, maka objek tersebut dapat dikatakan tidak dikenali atau *void*. Warna-warna tersebut digunakan berdasarkan referensi dari pustaka Python yang digunakan *cityscapesscripts* untuk membantu dalam proses persiapan *dataset* yang akan digunakan sebagai bahan pembelajaran dan pengujian seperti yang ditunjukkan pada Tabel 2.

Untuk dataset terdapat 2 jenis yang disatukan dengan rincian *dataset* local dan *dataset* umum yang telah beredar dengan total 5000 gambar. Hal tersebut dilakukan agar data yang disediakan lebih bervariasi untuk area perkotaan. Untuk *dataset* local berasal dari rekaman jalan di ITS, data yang digunakan merupakan gambar hasil rekaman kamera Intel RealSense D435i dengan total 3000 gambar dengan rincian 1000 gambar RGB, 1000 gambar *depth*, dan 1000 gambar NIR yang dianotasi untuk *masking* dengan CVAT.

Hasil dari gambar yang telah dianotasi terdapat data berupa gambar contoh, gambar hasil anotasi, label, dan data anotasi dengan ekstensi .json. Untuk ilustrasi warna dari kelas tersebut dapat dilihat pada Gambar 10.

Setelah dilakukan proses *dataset preparation* tersebut, akan diberikan output sesuai dengan kebutuhan dari penulis, yaitu antara lain gambar dan list dalam bentuk data .txt yang berisi *train*, validasi, dan *test* yang berisi dari gambar serta *file polygon* dari gambar yang akan digunakan. Serta jumlah dari kelas dan banyaknya gambar yang akan digunakan untuk proses *training*, validasi, serta *testing* dari model tersebut. Untuk *library* yang utamanya digunakan yaitu cityscapesScripts, OpenCV, dan Pillow.

Untuk *dataset* bagian NIR, penulis menggunakan metode pendekatan untuk menkonversi gambar RGB menjadi

(seperti) NIR yang akan digunakan pada dataset RGBDNIR. Ada beberapa cara metode yang dapat mengisi kekosongan dari dataset NIR, diantaranya grayscale, Pix2Pix [2]-[3], serta CycleGAN [3]. Tetapi dari semua itu, penulis memberikan catatan dimana bahwa gambar RGB pada dasarnya tidak dapat diubah (sempurna) menjadi *infrared* (*stereo*) atau *infrared thermal* dikarenakan memiliki spektrum gelombang warna yang berbeda seperti yang ditunjukkan pada Gambar 11.

E. Perancangan Model

Pada perancangan model ini, model dirancang sesuai dengan kebutuhan serta alat yang digunakan agar dapat berjalan lancer seperti yang ditunjukka pada Gambar 12. Mengingat penulis menggunakan GPU versi mobile yang memiliki keterbatasan performa seperti keterbatasan VRAM, *clock speed*, daya, dsb. Untuk itu, penulis membuat model seefisien mungkin dimana dapat berjalan dengan hasil yang terbaik. Untuk itu, penulis menggunakan *library* PyTorch sebagai pustaka mesin pembelajaran.

Penulis membuat model untuk RGBD yang dimana mengikuti konsep seperti pada Gambar 12, tetapi dengan beberapa tambahan seperti pada setiap *layer* akan terdapat fusion dengan konsep *modality* atau modalitas, yaitu menggabungkan *embedding* tingkat tinggi dari beberapa *input* yang berbeda lalu menerapkan *softmax* [4]. Atau singkatnya menggabungkan beberapa input menjadi satu yang menghasilkan satu *output*. Tujuan dari adanya modalitas ini untuk membangun model yang dapat memproses dan menghubungkan informasi dari berbagai modalitas.

Untuk *fusion* atau penggabungan antara RGB, D, penulis menggunakan metode penggabungan dimana setelah dilakukannya diperkecil (squeeze) dan eksitasi (mengekstrak fitur) dengan melakukan penjumlahan dari kedua model tersebut. Untuk diagram dari fusion tersebut dapat dilihat pada Gambar 13.

F. Loss Function

Pada saat dilakukannya *training*, tentu perlu dilakukannya menentukan seberapa besarnya error yang dimiliki dari sebuah model. Untuk perhitungan *loss* pada penelitian ini menggunakan metode Focal *Loss* for Dense Object Detection yang dipublikasikan oleh Facebook AI Research (FAIR) pada tahun 2017. Metode tersebut didesain untuk mengatasi masalah pada pendeteksian objek satu tahap dimana dapat terjadinya ketidakseimbangan antara kelas *foreground* dengan *background* selama pelatihan dengan menfokuskan *training* pada hard negative dengan menambahkan persamaan faktor modulasi $-(1-p_t)^{\gamma}$ ke cross entropy *loss*, dengan parameter fokus yang dapat disetel $\gamma \geq 0$. Persamaan (1) merupakan rumus Cross Entropy *Loss*:

$$CE(p,y) = \begin{cases} -\log(p), & jika\ y = 1 \\ -\log(1-p), & lainnya. \end{cases} \quad (1)$$

Pada Persamaan 1, $y \in \{\pm 1\}$ dispesifik pada kelas *ground truth* dan $p \in [0,1]$ merupakan probabilitas estimasi kelas pada model dengan label y=1. Dengan begitu, dapat didefinisikan p_t:

$$p_t = \begin{cases} p, & jika\ y = 1 \\ 1-p, & lainnya. \end{cases} \quad (2)$$

Dan dapat ditulis kembali $CE(p,y) = CE(p_t) = -\log(p_t)$. Dengan begitu, untuk persamaan Focal *Loss* sebagai *Loss Function* yang digunakan pada pengerjaan tugas akhir ini dapat didefinisikan pada Persamaan 3.

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \quad (3)$$

Ketika $\gamma = 0$, maka nilai FL sama seperti cross entropy. Pada penelitian ini, penulis menggunakan gamma sebesar $\gamma = 2$ dan alpha sebesar $\alpha = 0.5$ dengan p_t merupakan weight yang terdistribusi dari model tersebut [5].

G. Perhitungan IoU

Ketika akan dilakukannya proses validasi, tentunya perlu mendapatkan nilai keakuratan dari sebuah model yang telah dilatih dengan nilai metrik. Untuk mendapatkan nilai tersebut dapat menggunakan konsep IoU (*Intersection over Union*) atau Jaccard index merupakan perhitungan metrik yang mengvalidasi tumpang tindih antara masking ground-thruth (gt_m) dengan masking yang diprediksi (pd_m). IoU dapat dihitung sebagai luas tumpeng tindih atau persimpangan antara gt dan pd dibagi dengan luas persatuan antar keduanya pada Persamaan 4 [6].

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (4)$$

Atau untuk kebutuhan IoU ini dapat dibuat seperti pada Persamaan 5.

$$IoU = \frac{area(gt_m \cap pd_m)}{area(gt_m \cup pd_m)} \quad (5)$$

Dari persamaan tersebut, dapat dilihat gambar ilustrasi dari IoU seperti yang ditunjukkan pada Gambar 14 [3].

H. Validasi dan Testing serta Pengambilan Data

Pada perancangan program validasi ini dilakukan untuk mengvalidasi hasil *train* dari model pembelajaran mesin yang digunakan. Program validasi ini meliputi pemuatan data validasi dari dataset, memuat model untuk divalidasi dari setiap *epoch*, dan penyimpanan dari model yang divalidasi sesuai dengan keinginan penulis yaitu dari setiap 5 *epoch* bersamaan dengan pemeriksaan IoU dari model yang telah dilatih. Dari validasi ini juga akan disimpannya besaran tingkat iterasi dari model serta grafik *loss* terhadap *epoch* dan IoU terhadap *epoch*.

Pada pengujian alat ini dilakukan diluar ruangan. Untuk diluar ruangan, tepatnya di jalan yang berada di dalam Kampus ITS seperti pada gambar dibawah. Alat ini yang berupa kamera, serta laptop di luar ruangan dibawa dengan menggunakan kendaraan mobil milik teman saya Hafid Suparman mengitari kampus dengan menggunakan kamera Intel RealSense D435i melalui channel RGB, D, dan NIR menggunakan aplikasi Intel RealSense Viewer yang disediakan oleh tim Intel RealSense pada OS Windows 10 dan Ubuntu 20.04. Untuk pengujian ini menggunakan konfigurasi semua *default* dari pengembang serta untuk resolusi menggunakan konfigurasi 1280 × 720-*pixel* dengan 30 fps. Untuk tampilan dari RealSense Viewer tersebut (untuk Windows 10) dapat dilihat pada Gambar 15.

Selain melakukan pengujian dengan aplikasi tersebut, dilakukan juga pengujian dengan menggunakan program python dengan pustaka pyrealsense2. Untuk pemasangan pustaka pyrealsense2 dapat digunakan pada OS Windwos

ataupun Ubuntu. Untuk pengujian ini, penulis menggunakan OS Windows 10 dan Ubuntu 20.04. Untuk pemasangan dari pustaka tersebut, penulis menggunakan perintah pip seperti pada Gambar 16.

Setelah pemasangan pustaka tersebut, penulsi membuat sebuah kode yang dapat mengoperasikan kamera dengan mode yang dibutuhkan seperti penggunaan kamera RGB, *Depth*, dan NIR yang memiliki konfigurasi seperti pada program *viewer* seperti hasil yang ditunjukkan oleh Gambar 17.

IV. ANALISIS HASIL

A. Hasil Percobaan Alat

Dari percobaan yang telah dilakukan, alat yang digunakan sebagai *case* kamera berbentuk seperti kubus dan pengait tersebut memiliki bentuk seperti capit dengan foto seperti yang ditunjukkan pada Gambar 18 dan Gambar 19.

B. Hasil Dataset Preparation

Hasil dari persiapan dataset yang telah dilakukan dengan jumlah gambar sebanyak 5000 gambar sebagai *ground truth* dapat dilihat pada Gambar 20.

Dapat dilihat dari Gambar 19 bahwa dataset tersebut telah diproses untuk persiapan *train*, validasi, dan *testing* sebanyak 5000 file anotasi. Data tersebut akan dipergunakan selanjutnya untuk dilatih ke model yang akan digunakan seperti yang ditunjukkan pada Gambar 21.

C. Hasil Pelatihan dan Validasi

Dari hasil *train* dan validasi yang dilakukan, didapatkan hasil seperti yang ditunjukkan Gambar 22.

Dari hasil yang ditunjukkan Gambar 22 merupakan ringkasan dari keseluruhan data yang telah didapat dengan batas maksimal mIoU yaitu 1 (satu). Dan dapat dilihat, bahwa model dengan waktu *train*&valid tercepat adalah ESPNet dan model dengan waktu *train*&valid terlama adalah RGBD seperti yang ditunjukkan Gambar 23 sampai Gambar 27. Untuk nilai IoU tertinggi adalah model DABNet sebesar 0.6371 atau 63.71% dan model dengan nilai IoU terkecil adalah FSSNet. Lalu untuk model dengan nilai *loss* tertinggi adalah ENet dengan nilai 1.2094 dan model dengan nilai *loss* terkecil adalah RGBD dengan nilai *loss* sebesar 0.553.

D. Hasil Segmentasi

Hasil segmentasi atau *testing* setelah dilakukannya *train* and valid dapat dilihat pada Gambar 28.

E. Hasil Waktu Segmentasi

Dari dilakukannya pengujian tersebut juga mengukur rentang waktu antar frame saat dilakukannya *testing* segmentasi dengan kamera di jalan ITS dimana dapat dikatakan sebagai *frame rate* dengan satuan fps, dimana banyaknya *frame* yang dapat ditampilkan dalam satuan waktu (detik). Untuk besar rentang waktu tersebut dan mengkonfersikannya ke besaran FPS dapat digunakan rumus seperti pada persamaan 6.

$$Frame\ Rate = \frac{1}{t} \quad (6)$$

Dimana t merupakan waktu yang dibutuhkan untuk memproses segmentasi satu *frame*/gambar. Dengan begitu didapatkan besaran *frame rate* yang dapat dilihat pada Tabel 3.

Dapat dilihat bahwa model dengan waktu tersingkat (atau FPS terbesar) adalah ESPNet dan FastSCNN, dan yang model dengan *frame rate* terkecil adalah SegNet.

V. KESIMPULAN

Dari penelitian serta percobaan yang telah dilakukan dapat disimpulkan bahwa: (1) Program *machine learning* untuk segmentasi pada komputer dapat mendeteksi jalan beserta objek lainnya dengan kamera Intel RealSense D435i sebagaimana hasil tersebut berupa segmentasi dengan kelas yang memiliki warna masing-masing dengan pengujian model CGNet, DABNet, EDANet, ENet, ESPNet, FastSCNN, FSSNet, LinkNet, SegNet, SQNet, UNet, dan RGBD; (2) Untuk sistem pendeteksian menggunakan *semantic segmentation* dapat dihasilkannya pemetaan kelas objek yang terbaca oleh komputer walaupun masih terdapatnya beberapa *error* atau kesalahan pada saat *testing*; (3) Untuk performa yang didapat, penulis mendapatkan performa berupa nilai mIoU, *loss*, serta besarnya *frame rate* dari berbagai model. Hasil yang diapat dapat disimpulkan untuk model RGBD memiliki nilai mIoU 58.74% dimana lebih rendah dari beberapa model lainnya seperti DABNet 63.71%, EDANet 62.41%, dan CGNet 58.88%. Walaupun nilai IoU yang didapat bukan yang tertinggi, tetapi untuk nilai *loss* yang diapat merupakan nilai *loss* terkecil dengan nilai 0.553 dengan model seperti DABNet 1.0393, EDANet 1.054, CGNet 1.0389, serta UNet 0.9604. Untuk besarnya *frame rate*, model dengan nilai *frame rate* tertinggi adalah ESPNet dan FastSCNN sebesar 100 fps, untuk DABNet, EDANet, dan CGNet sebesar 50 fps. Untuk model RGBD didapatkan *frame rate* sebesar 7.8125 fps atau waktu yang dibutuhkan untuk melakukan segmentasi antar gambar/*frame* yaitu sebesar 0.128 detik; (4) Untuk peningkatan performa berupa *frame rate* dapat menggunakan resolusi yang lebih kecil atau model yang lebih sederhana seperti FastSCNN dan ESPNet yang memiliki *frame rate* mencapat 100 fps dengan perbandingan model UNet, SegNet, dan SQNet sebsar 4.54, 8.33, dan 12.5 fps sehingga masih dibawah *frame rate* kamera Intel RealSense D435i sebesar 30 fps. Untuk meningkatkan nilai IoU dapat menggunakan model seperti DABNet, EDANet, CGNet, serta RGBD yang memiliki nilai IoU tertinggi.

Berdasarkan penelitian tersebut, terdapat kekurangan yang dapat diperbaiki dan ditingkatkan, seperti: (1) Disarankan menggunakan kamera dengan FOV antar modul yang sama dari pabrikan agar tidak perlu dilakukannya proses sinkronisasi (*alignment*); (2) Penggunaan laptop dengan kartu grafis Nvidia RTX 3060 Mobile GPU dengan memory 6GB sebaiknya diganti dengan divais yang lebih kuat ataupun memiliki kapasitas VRAM yang lebih besar agar dapat terhindar dari adanya keterbatasan performa. Walaupun tujuan dari pengerjaan tugas akhir ini diutamakan untuk *platform mobile*; (3) Pembuatan model *machine learning* lebih baik dibuat lebih sederhana karena akan mempersulit *troubleshooting* jika terjadi kesalahan, terutama kesalahan pada *library* Python yang tidak kompatibel.

DAFTAR PUSTAKA

[1] A. Pratama and A. S. Sembiring, "Implementasi metode histogram equalization dan median filter dalam perbaikan citra satelit," *Pelita Inform. Inf. dan Inform.*, vol. 7, no. 2, pp. 114–119, 2018.

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[4] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H.-M. Gross, "Efficient Rgb-D Semantic Segmentation For Indoor Scene Analysis," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13525–13531. doi: 10.1109/ICRA48506.2021.9561675.

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[6] S. Kosub, "A note on the triangle inequality for the Jaccard distance," *Pattern Recognit. Lett.*, vol. 120, pp. 36–38, 2019.