TUGAS AKHIR – EE 184801

SISTEM PENGENALAN RUANGAN SECARA VISUAL BERBASIS *CONVOLUTIONAL NEURAL NETWORK* DI DEPARTEMEN TEKNIK ELEKTRO ITS UNTUK *SERVICE ROBOT*

Aurell Quinta Miranda
NRP 07111640000072

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng
Muhammad Attamimi B.Eng., M.Eng., Ph.D.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EE 184801

SISTEM PENGENALAN RUANGAN SECARA VISUAL BERBASIS CONVOLUTIONAL NEURAL NETWORK DI DEPARTEMEN TEKNIK ELEKTRO ITS UNTUK *SERVICE ROBOT*

Aurell Quinta Miranda
NRP 07111640000072

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

FINAL PROJECT – EE 184801

VISUAL ROOM RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS FOR SERVICE ROBOTS AT ELECTRICAL ENGINEERING DEPARTMENT ITS

Aurell Quinta Miranda
NRP 07111640000072

Supervisor
Dr. Ir. Djoko Purwanto, M.Eng
Muhammad Attamimi B.Eng., M.Eng., Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Sistem Pengenalan Ruangan Secara Visual Berbasis *Convolutional Neural Network* di Departemen Teknik Elektro ITS untuk *Service Robot*" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya,

Aurell Quinta Miranda
NRP. 0711 16 4000 0072

Halaman ini sengaja dikosongkan

SISTEM PENGENALAN RUANGAN SECARA VISUAL BERBASIS *CONVOLUTIONAL NEURAL NETWORK* DI DEPARTEMEN TEKNIK ELEKTRO ITS UNTUK *SERVICE ROBOT*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I,

Dr. Ir. Djoko Purwanto, M.Eng
NIP. 196512111990021002

SURABAYA
JUNI, 2020

Halaman ini sengaja dikosongkan

SISTEM PENGENALAN RUANGAN SECARA VISUAL BERBASIS *CONVOLUTIONAL NEURAL NETWORK* DI DEPARTEMEN TEKNIK ELEKTRO ITS UNTUK *SERVICE ROBOT*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing II,

M. Attamimi, B.Eng., M.Eng., Ph.D.
NIP. 198503272019031006

SURABAYA
JUNI, 2020

Halaman ini sengaja dikosongkan

SISTEM PENGENALAN RUANGAN SECARA VISUAL BERBASIS *CONVOLUTIONAL NEURAL NETWORK* DI DEPARTEMEN TEKNIK ELEKTRO ITS UNTUK *SERVICE ROBOT*

Nama : Aurell Quinta Miranda
Pembimbing I : Dr. Ir. Djoko Purwanto, M.Eng
Pembimbing II : Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRAK

Service robot merupakan robot yang didesain untuk melayani manusia. Dalam menjalankan berbagai tugasnya, akan lebih efektif jika *service robot* memiliki kemampuan untuk bergerak secara bebas dalam suatu lingkungan tanpa bantuan manusia dan bahkan mampu diinstruksikan untuk pergi ke ruangan yang dinginkan. Untuk dapat melakukan hal tersebut,hal krusial yang diperlukan robot adalah lokalisasi, atau kemampuan untuk mengetahui letak dirinya berada. Pada tugas akhir ini, dikembangkan suatu sistem lokalisasi kualitatif robot, yaitu sistem yang mampu mengenali lima buah ruangan di Departemen Teknik Elektro ITS dengan menggunakan indera visual. Metode yang digunakan untuk mengenali ruangan tersebut adalah dengan menggunakan jaringan syaraf konvolusional. Pada tugas akhir ini penulis melakukan pengujian terhadap beberapa model jaringan syaraf konvolusional dengan menggunakan jenis data visual yang berbeda-beda, dan kemudian mengimplementasikan model dengan tingkat akurasi tertinggi kepada sistem pengenalan ruangan. Dari hasil pengujian, model yang dirancang penulis (Model-A), yang menggunakan dua jenis data visual (RGB dan Infrared) sebagai masukannya, memiliki tingkat akurasi tertinggi yaitu sebesar 48.23%. Dari lima buah ruangan yang dikenali, model ini paling mampu mengenali ruangan meeting sekretariat DTE, yaitu dengan tingkat akurasi sebesar 62.15%, dan paling mengalami kesulitan dalam mengenali ruangan C215, yaitu dengan tingkat akurasi sebesar 31.80%

Kata kunci: rekognisi visual, jaringan syaraf konvolusional, *service robot*

Halaman ini sengaja dikosongkan

Visual Room Recognition using Convolutional Neural Networks for Service Robots at Electrical Engineering Department ITS

Name : Aurell Quinta Miranda
Supervisor : Dr. Ir. Djoko Purwanto, M.Eng
Co-Supervisor : Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRACT

Service robots are robots that are designed to serve human needs. For the sake of being able to perform their task more effectively, it's preferable for service robots to be able to move freely on an environment and even able to move to a certain place when instructed. To be able to do capabilities said, one of the crucial thing the robot should have is localization, or the ability to know where it is. Therefore, in this final project a visual room recognition system is constructed to be able to recognize five different rooms at Electrical Engineering Department (DTE) ITS using visual senses. Method used to recognize rooms said is by using convolutional neural networks (CNN). In this final project, three different models, trained on different visual data types, were tested for the sake of finding the best combination of model to data type to be implemented as visual room recognition. From the test results, the model that was able to perform the best was the model that the writer constructed (Model-A) and trained using two visual data (RGB and Infrared), which was able to recognize the rooms with an accuration of 48.23%. From five of the rooms recognized, this model performs the best on recognizing DTE's secretarial meeting room, with a precision of 62.15%. On the other hand, this model performs the worst on recognizing room C215, with a precision of 31.80%

Key words: visual recognition, convolutional neural network, service robots

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kepada Tuhan Yesus Kristus atas berkat dan kasih karunia yang Ia diberikan, penulis dapat menyelesaikan laporan tugas akhir dengan judul "**SISTEM PENGENALAN RUANGAN SECARA VISUAL BERBASIS *CONVOLUTIONAL NEURAL NETWORK* DI DEPARTEMEN TEKNIK ELEKTRO ITS UNTUK *SERVICE ROBOT***", sebagai salah satu persyaratan dalam menyelesaikan pendidikan program S1 di Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.

Dalam penyusunan dan penyelesaian laporan Tugas Akhir ini penulis mendapatkan banyak sekali doa, bantuan dan dukungan dari berbagai pihak. Maka dari itu penulis ingin menyampaikan ucapan terimakasih kepada:

1. Orang tua dan keluarga, yang senantiasa memberikan kasih sayang dan dukungan kepada penulis.
2. Bapak Dr. Ir. Djoko Purwanto, M.Eng serta Bapak Muhammad Attamimi, B.Eng., M.Eng., Ph.D., selaku kedua dosen pembimbing atas gagasan topik tugas akhir serta bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir.
3. Herline, Riri, Divyah, Mail, Salik, Mar'ie, dan teman-teman angkatan e56 lainnya, atas doa dan bantuan yang diberikan selama pengerjaan Tugas Akhir ini.
4. Teman-teman Sok Seleb, yang senantiasa menjadi tempat penghiburan dan peristirahatan selama pengerjaan Tugas Akhir ini.

Penulis menyadari bahwa masih banyaknya kekurangan dan hal-hal yang dapat dikembangkan pada tugas akhir ini. Oleh karena itu penulis berharap dapat menerima masukan serta kritik yang membangun. Semoga tugas akhir ini dapat membawa manfaat bagi berbagai pihak.

Surabaya, 9 Juni 2019

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

PERNYATAAN KEASLIAN i
Lembar Pengesahan I iii
Lembar Pengesahan II v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xv
DAFTAR TABEL xvii
BAB 1 PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Perumusan Masalah 3
1.3. Tujuan Penelitian 3
1.4. Batasan Masalah 3
1.5. Metodologi Penelitian 3
1.6. Sistematika Penulisan 4
1.7. Relevansi 5
BAB 2 TINJAUAN PUSTAKA DAN TEORI PENUNJANG 7
2.1. Teori Penunjang 7
2.1.1. Rekognisi Visual 7
2.1.2. Data Visual 8
2.1.3. Jaringan Syaraf Konvolusional 10
2.1.4. Akurasi Top-N 13
2.1.5. *Confusion Matrix* 14
2.2. Tinjauan Pustaka 16
2.2.1 AlexNet 16
2.2.2 DeepSpace 17
2.2.3 Multi-Layer Perceptron Convolutional Layer 18
2.2.4 Intel RealSense D435i 19
2.2.5 Keras 21
2.2.6 TensorFlow 21
2.2.7 OpenCV-Python 22
BAB 3 PERANCANGAN SISTEM 23
3.1. Data 25

3.2. Pelatihan Jaringan 28
3.2.1. Pelatihan Jaringan dengan 1 Jenis Data 29
3.2.2. Pelatihan Jaringan dengan 2 Jenis Data 33
3.3. Pembuatan Program Utama 35
3.4. Realisasi Sistem 39
BAB 4 PENGUJIAN DAN ANALISA 41
4.1. Metode Pengujian 41
4.2. Pengujian Sistem 41
4.2.1. Sistem dengan Masukan 1 Jenis Data 41
4.2.2. Sistem dengan Masukan 2 Jenis Data 42
4.3. Analisa 44
BAB 5 PENUTUP 49
5.1. Kesimpulan 49
5.2. Saran 49
DAFTAR PUSTAKA 51
LAMPIRAN A 53
LAMPIRAN B 59
LAMPIRAN C 64
LAMPIRAN D 67
LAMPIRAN E 72
LAMPIRAN F 74
BIODATA PENULIS 76

DAFTAR GAMBAR

Gambar 2. 1 Perbedaan Citra RGB dan Depth 9
Gambar 2. 2. Perbedaan Citra RGB dan NIR 9
Gambar 2. 3. Arsitektur Jaringan Syaraf Konvolusional pada Umumnya 10
Gambar 2. 4. Ilustrasi Lapisan Penyatu 12
Gambar 2. 5. Arsitektur AlexNet [3] 17
Gambar 2. 6. Arsitektur DeepSpace 18
Gambar 2. 7. ConvLayer (kiri) dan MLPConvLayer (kanan) 19
Gambar 2. 8. Kamera Intel RealSense D435i 20
Gambar 2. 9. Logo Keras 21
Gambar 3. 1. Diagram Blok Jaringan Syaraf Konvolusional 24
Gambar 3. 2. Diagram Blok Sistem 24
Gambar 3. 3. Tampilan Intel RealSense Viewer 26
Gambar 3. 4. Perintah rs-convert 26
Gambar 3. 5. Format Database Keras 27
Gambar 3. 6. Contoh Citra RGB, infrared, dan depth yang digunakan. 28
Gambar 3. 7. Arsitektur Model-A 33
Gambar 3. 8. Contoh Arsistektur Model-A dengan Input RGB dan Infrared 33
Gambar 3. 9. Flowchart Program Utama 36
Gambar 3. 10. Realisasi Fisik Sistem 39
Gambar 3. 11. Tampilan *Interface* Sistem 40
Gambar 4. 1. Confusion Matrix Model-A 45
Gambar 4. 3. Contoh gambar ruangan C215 yang diambil pada sudut yang berbeda-beda 47
Gambar 4. 2. Contoh gambar ruangan meeting sekretariat DTE yang diambil pada sudut yang berbeda-beda. 47

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2. 1. Contoh Hasil Klasifikasi Multi-Class 14
Tabel 2. 2. *Confusion Matrix* Klasifikasi Biner..... 14
Tabel 2. 3. *Confusion Matrix* Klasifikasi *Multi-Class*..... 15
Tabel 2. 4. Spesifikasi Kamera Intel RealSense D435i 21
Tabel 3. 1 Grafik Jumlah Data per Kategori 27
Tabel 4. 1. Akurasi Validasi Model dengan 1 Jenis Data 43
Tabel 4. 2. Hasil Pengujian Model dengan 1 Jenis Data 43
Tabel 4. 3. Akurasi Validasi Model dengan 2 Jenis Data 44
Tabel 4. 4. Hasil Pengujian 2 Jenis Data 44
Tabel 4. 5 Presisi Model-A Terhadap 5 Kategori Ruangan 46

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Tugas akhir merupakan suatu penelitian yang dilakukan sebagai persyaratan akademik untuk mendapatkan gelar sarjana teknik di Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Pada bab ini membahas mengenai hal-hal yang mendahului pelaksanaan tugas akhir. Hal tersebut meliputi latar belakang, perumusan masalah, tujuan penulisan, batasan masalah, metodologi penelitian, sistematika penulisan, dan relevansi.

1.1. Latar Belakang

Service Robots seperti namanya, merupakan robot yang didesain untuk melayani manusia. Beberapa *Service Robots* didesain untuk membantu kita dalam melakukan pekerjaan-pekerjaan rumah, seperti menyapu, membersihkan, menghidangkan makanan, dan lainnya. *Service Robots* juga sering kali didesain untuk memberikan edukasi dan hiburan, dengan memberikan robot kemampuan untuk berkomunikasi dan bahkan mengenali emosi manusia. Ada juga *Service Robots* yang berfungsi untuk menjadi asisten pribadi, yang membantu dalam mengerjakan hal-hal seperti memasang alarm, membelikan barang secara online, memberitahukan ramalan cuaca, dan banyak lagi. Begitu banyak variasi kemampuan yang dapat dimiliki *Service Robots*, namun kebanyakan hanya berorientasi pada satu tugas dan dalam satu area saja. Contohnya pada Robot Ugo[1], robot yang dibuat oleh suatu perusahaan Jepang bernama Mira Robotics, memiliki kemampuan untuk melipat baju, menjemur, dan memasukkannya dalam mesin cuci. Robot ini hanya akan menghabiskan waktunya bekerja pada area *laundry*, sehingga tidak memiliki kemampuan untuk bergerak secara mandiri menuju tempat-tempat dalam rumah. Ada juga robot Zenbo[2], robot yang dikembangkan oleh Asus ini berguna sebagai asisten pribadi, bekerja seperti Amazon-Alexa versi robot. Memang robot ini dapat bergerak bebas didalam rumah, namun dengan cara mengikuti “master”-nya (*Follow-Me Robot*), bukan secara mandiri. Dengan perkembangan terus menerus dalam bidang *Service Robots*, bukan suatu hal yang tidak masuk akal jika kita berekspektasi atas munculnya suatu robot yang tidak membutuhkan bantuan manusia dalam pergerakannya dalam suatu lingkungan dan mampu kita instruksikan untuk pergi ke ruangan yang kita inginkan. Untuk dapat melakukan hal tersebut, suatu hal krusial yang diperlukan

robot adalah lokalisasi secara akurat.

Menurut Tsotsos,dkk pada 2016 13th Conference on Computer and Robot Vision[8], lokalisasi merupakan suatu kemampuan robot untuk mengetahui letak dirinya, baik secara kuantitatif atau secara kualitatif. Lokalisasi secara kuantitatif dapat berupa koordinat tempat robot berada, sedangkan secara kualitatif berupa pengetahuan terhadap ruang robot itu berada, seperti nama ruang, nama gedung, dsb. Umumnya untuk mendapatkan kemampuan lokalisasi secara kualitatif ini adalah dengan menggunakan "mata" untuk melihat ruangannya, yaitu menggunakan pengenalan tempat secara visual (*Visual Place Recognition*), dimana suatu sensor visual digunakan untuk mendapatkan citra dari ruangan yang dilewati dan kemudian diproses menjadi suatu informasi berupa jawaban dari pertanyaan (1)Apakah robot pernah melewati ruangan tersebut? (2)Jika Ya, apakah robot mengetahui informasi terkait ruangan itu (contoh: nama)?. Kedua jawaban dari pertanyaan ini kemudian dapat digunakan *Service Robots* untuk memvalidasi letak dia berada, ketika mendapatkan suatu perintah untuk harus melewati atau mendatangi suatu ruangan, sehingga dapat meningkatkan akurasi dari lokalisasi robot tersebut.

Untuk dapat menjawab kedua pertanyaan tersebut dengan akurat, sistem pengenalan ruangan harus mampu mengatasi perbedaan-perbedaan yang mungkin terjadi pada ruang yang sama, seperti perbedaan pencahayaan, skala, dan perspektif. Dalam mencapai hal tersebut, suatu metode yang mulai dikenal adalah menggunakan *Convolutional Neural Network* (CNN). CNN merupakan jenis Neural Network yang menggunakan teknik konvolusi, dimana suatu filter kernel konvolusi berukuran tertentu akan digerakan ke suatu input citra. Informasi representatif baru akan didapatkan dari hasil perkalian bagian gambar tersebut dengan filter yang digunakan. CNN sendiri sudah terkenal untuk digunakan dalam rekognisi berbasis visual, dengan tingkat akurasi yang tinggi juga. Pada tugas akhir ini, akan dilihat bagaimana performansi CNN dalam mengenali ruangan dengan menggunakan jenis data atau kombinasi jenis data yang berbeda-beda. Hasil analisa tersebut pun akan digunakan untuk membangun suatu sistem pengenalan ruangan menggunakan data visual dengan metode berbasis Convolutional Neural Network pada lingkungan kampus ITS, dengan harapan dapat menjadi dasar pengembangan lokalisasi pada *Service Robots* yang mampu bergerak secara mandiri dalam suatu lingkup dalam ruang.

1.2. Perumusan Masalah

1. Model jaringan syaraf konvolusional seperti apa yang paling mampu mengenali ruangan-ruangan di DTE?
2. Jenis data visual apa yang dibutuhkan untuk mendapatkan tingkat akurasi pengenalan ruangan yang paling optimal?
3. Bagaimana proses pengambilan data yang realistis terhadap kemungkinan pengembangan robot?

1.3. Tujuan Penelitian

Tujuan yang ingin dicapai dari tugas akhir ini adalah sebagai berikut;

1. Membangun sistem rekognisi ruangan secara visual yang mampu mengenali ruangan-ruangan di Departemen Teknik Elektro ITS.
2. Mengamati dan menganalisa performansi sistem dengan jenis data dan kombinasi jenis data yang berbeda-beda.

1.4. Batasan Masalah

1. Tugas akhir ini hanya menitikberatkan pada sistem rekognisi ruangan yang dapat diimplementasikan pada *Service Robots*, bukan pada pengembangan *Service Robots* itu sendiri.
2. Ruangan yang dikenali merupakan ruangan yang objek didalamnya dalam keadaan statis.
3. Ruangan yang dikenali berada pada kondisi pencahayaan terang (siang hari).

1.5. Metodologi Penelitian

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. Studi literatur

Studi literatur bertujuan untuk mempelajari dan mengkaji teori dasar dan ataupun data-data penelitian sebelumnya yang relevan dengan tugas akhir yang akan dikerjakan. Literatur ini dapat diambil dari jurnal-jurnal, buku, artikel ataupun *paper* yang memiliki keabsahan sehingga dapat dijadikan referensi.

2. Pembuatan Database

Tahap ini meliputi pengambilan serta pra-proses data. Pengambilan data dilakukan dengan cara merekam video dari 5 buah

ruangan yang akan dikenali. Video yang direkam merupakan video RGB, depth, dan juga IR Video tersebut kemudian dipotong Frame-to-frame sehingga didapatkan data berupa citra RGB, depth, dan NIR. Citra tersebut diubah dalam bentuk jpg dan ukurannya diubah menjadi sebesar 227x227px. Citra yang sudah selesai diproses kemudian disimpan dalam bentuk format database keras.

3. Pelatihan dan Pengujian Model Jaringan

Database yang sudah siap digunakan kemudian dijadikan sebagai input bagi pelatihan serta pengujian Neural Network. Pada tahap ini Neural Network yang digunakan adalah AlexNet, DeepSpace, serta 1 buah model neural network yang dirancang oleh penulis. Pelatihan dan pengujian dilakukan beberapa kali sebanyak jumlah neural network x jumlah jenis data yang digunakan sebagai input.

4. Analisa Model Jaringan

Hasil pelatihan serta pengujian tiap neural network untuk tiap jenis data yang ada pada tahap tiga kemudian dianalisa untuk kemudian didapatkan kombinasi Neural Network serta jenis data yang terbaik dalam melakukan rekognisi.

5. Pembuatan Sistem Rekognisi

Pada tahap ini dibangun sistem rekognisi dengan mengimplementasikan pilihan terbaik dari tahap 3. Sistem rekognisi ini dilakukan pada notebook dengan menjalankan program dengan bahasa pemrograman Python, yang terkoneksi dengan kamera Intel RealSense D435i.

6. Analisa dan Evaluasi

Analisa dilakukan untuk mengetahui kelancaran serta keberhasilan sistem dalam melakukan tugasnya. Jika hasil analisa menyatakan bahwa sistem belum sesuai maka akan dilakukan evauasi.

7. Penulisan Laporan Tugas Akhir

Tahap akhir adalah penulisan laporan. Proses pengerjaan dan data-data hasil pengujian sistem dilaporkan dalam bentuk tulisan yang merupakan kesimpulan dari tugas akhir ini.

1.6. Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai

berikut:

- **BAB I: Pendahuluan**
  Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.
- **BAB II: Tinjauan Pustaka**
  Bab ini berisi tentang teori-teori penunjang serta tinjauan pustaka dari penelitian yang pernah dilakukan. Selain itu juga dijelaskan perangkat-perangkat yang digunakan pada tugas akhir ini.
- **BAB III: Perancangan Sistem**
  Bab perancangan sistem akan membahas tahap-tahap yang dilakukan dalam merancang sistem, hal ini meliputi tahap-tahap dalam menentukan model jaringan yang digunakan, serta tahap dalam membuat sistem.
- **BAB IV: Pengujian dan Analisis**
  Pada bab ini menguraikan tentang pengujian sistem pengenalan ruangan menggunakan model jaringan syaraf pada bab III dan dilakukan analisa terhadap hasil pengujian.
- **BAB V: Penutup**
  Bab ini berisi tentang kesimpulan yang diperoleh dari pembuatan alat serta saran untuk pengembangan lebih lanjut.

1.7. Relevansi

Hasil dari tugas akhir ini diharapkan mampu menjadi dasar pengembangan *service robot* yang mampu bergerak secara mandiri pada suatu area, ataupun kebutuhan lokalisasi kualitatif dalam ruangan lainnya.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA DAN TEORI PENUNJANG

Pada bab ini akan memaparkan teori-teori pendukung. Teori-teori ini menjadi dasar dalam penyusunan laporan tugas akhir ini. Tinjauan pustaka memuat teori dan penelitian yang telah dilakukan sebelumnya. Teori tersebut digunakan untuk menemukan solusi dalam permasalahan pada tugas akhir ini.

2.1. Teori Penunjang

Service robot merupakan suatu robot yang memiliki tugas untuk melayani dan membantu manusia untuk mengerjakan berbagai macam pekerjaan. Salah satu kemampuan penting yang harus dimiliki suatu service robot yang dapat bergerak bebas dalam suatu area adalah mengetahui letak dia berada, hal ini disebut dengan lokalisasi. Pada tugas akhir ini lokalisasi dari robot dibuat menggunakan sistem pengenalan ruangan berbasis jaringan syaraf konvolusional dengan menggunakan indera visual berupa kamera.

2.1.1. Rekognisi Visual

Rekognisi visual merupakan suatu kegiatan mengenali sesuatu menggunakan sensor visual. Rekognisi visual merupakan metode untuk menyediakan suatu inspeksi atau analisis berbasis citra pada berbagai aplikasi seperti inspeksi otomatis, kontrol proses, robot, dan industri. Rekognisi visual pada robot menggunakan kamera untuk melihat dan kemudian citra yang didapatkan dari kamera diproses sehingga menghasilkan fitur-fitur, fitur tersebut kemudian digunakan untuk mendeterminasikan output sistem, biasanya berupa nama atau identifier dari benda ataupun tempat atau apapun yang sedang dikenali. Secara mendasar, suatu sistem rekognisi visual memiliki alur sebagai berikut:

Input Data ⟹ Ekstraksi Fitur ⟹ Rekognisi ⟹ Output

Input data bagi suatu rekognisi visual dapat berupa citra RGB, HSI, Depth, ataupun NIR. Jenis data apa yang perlu digunakan perlu disesuaikan dengan jenis rekognisi yang hendak dilakukan. Jika rekognisi benda, seringkali digunakan citra RGB dan atau HSI. Namun jika rekognisi yang hendak dilakukan memerlukan pengetahuan berupa jarak dari benda ataupun bentuk dari suatu ruangan, dapat digunakan data depth dan NIR.

2.1.2. Data Visual

Data visual merupakan jenis data yang diperoleh dari suatu sensor visual. Dari sensor visual berupa kamera Intel RealSense D435i terdapat beberapa jenis data yang dapat diperoleh, yaitu RGB, Infrared, dan depth.

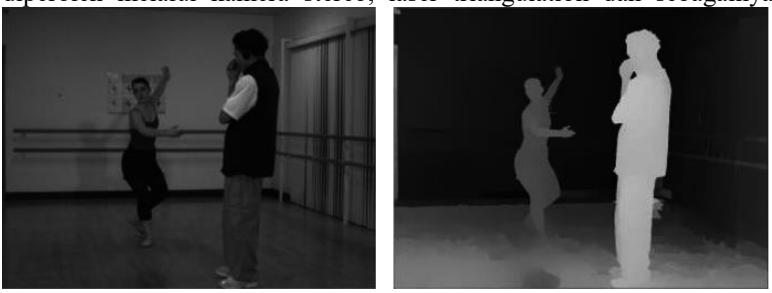
RGB

RGB merupakan singkatan dari *Red Green Blue* yang mengacu pada tiga warna cahaya yang dapat digabungkan dan akan menghasilkan warna-warna lainnya. Suatu data RGB merupakan jenis data yang merepresentasikan suatu warna pada suatu pixel berdasarkan kombinasi intensitas warna merah, hijau, dan birunya. Ketika 100% dari warna bergabung, maka akan terbentuk cahaya berwarna putih. Namun sebaliknya jika 0% dari warna bergabung, tidak ada cahaya yang terbentuk sehingga menghasilkan warna hitam. Jumlah warna yang dapat dihasilkan dari model warna RGB ini bergantung terhadap nilai yang mungkin yang dapat digunakan untuk merah, hijau, dan biru. Hal ini disebut dengan kedalaman warna dan diukur dalam bit. Kedalaman warna yang paling umum adalah 24-bit, sering juga disebut dengan “true color” atau warna asli. 24-bit kedalaman warna RGB ini berarti terdapat 8-bit bagi setiap warna. Hal ini berarti terdapat 2^8 atau 256 nilai pada tiap warna merah, hijau, dan biru, atau jumlah keseluruhan 256 x 256 x 256 = 16,777,216 warna yang mungkin dihasilkan.

Depth

Depth map merupakan suatu citra yang mengandung informasi terkait jarak antara bidang objek dari suatu sudut pandang tertentu.

Melalui depth map kita dapat mengetahui objek mana yang lebih dekat atau lebih jauh dari kita atau kamera.Nilai intensitas dari tiap pixel merepresentasikan jarak objek terhadap titik pandang. Citra depth dapat diperoleh melalui kamera stereo, laser triangulation dan sebagainya.

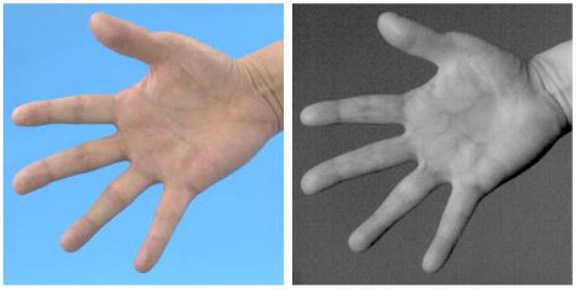


(a) Real-image (b) Depth-map

Gambar 2. 1 Perbedaan Citra RGB dan Depth

NIR

Suatu citra NIR merupakan citra yang didapatkan dari kamera NIR, yaitu kamera yang mampu menangkap bagian pada spektrum elektromagnetik yang berdekatan namun tidak pada jangkauan yang dapat dilihat manusia. NIR banyak digunakan dalam aplikasi yang memiliki pengondisian cahaya yang buruk, seperti monitor jalanan.

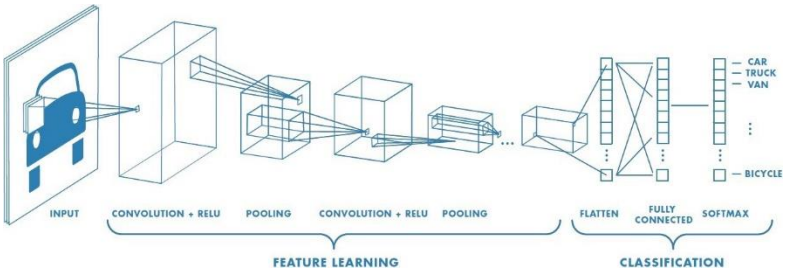


(a) RGB (b) NIR

Gambar 2. 2. Perbedaan Citra RGB dan NIR

2.1.3. Jaringan Syaraf Konvolusional

Jaringan syaraf konvolusional (*Convolutional Neural Network*) merupakan suatu kategori jaringan syaraf yang cukup dikenali dengan kemampuannya dalam melakukan deteksi citra dan klasifikasi dengan mengekstraksi fitur seperti sudut, garis, bentuk, dan warna, tanpa perlunya desain yang terlalu kompleks. Pada umumnya suatu jaringan syaraf konvolusional memiliki tiga jenis lapisan yang selalu ada, yaitu lapisan konvolusional, lapisan penyatu, dan lapisan terhubung-penuh.



Gambar 2. 3. Arsitektur Jaringan Syaraf Konvolusional pada Umumnya

Lapisan Konvolusional

Lapisan konvolusional bekerja dengan dasar teknik konvolusi gambar dimana filter akan menyelusuri setiap pixel dari data untuk menghasilkan suatu nilai yang merepresentasikan pixel seukuran filter tersebut. Tujuan utamanya adalah untuk mengekstraksi fitur dari suatu input $x = \{L_1, L_2, \dots, L_i\}$, dimana x merupakan suatu input citra pada tingkat pertama dan fitur dari input citra tersebut berada pada tingkat tengah. Setiap input dapat dibagi menjadi suatu set yang disebut dengan l_i. Terdapat juga filter *kernel*, dimana ukuran dari tiap filter digunakan dengan bias untuk mengkalkulasi ouput dari suatu sel FM_j sebagai berikut:

$$FM_j^i = b_j + \sum k_j \times L_i$$

Lapisan konvolusional merupakan lapisan *Multi-Layer Perceptron*

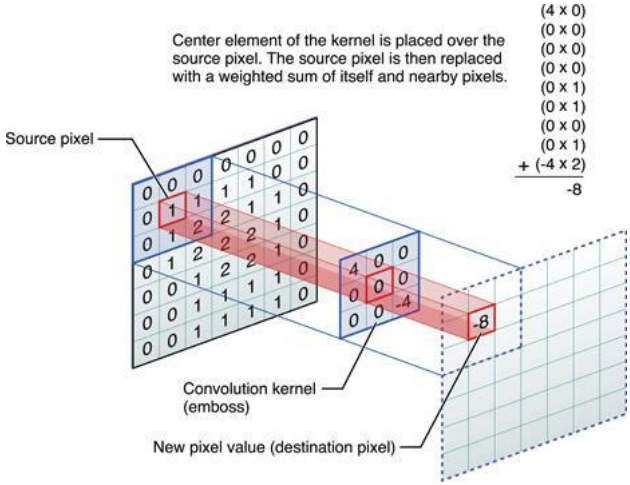
(MLP) dalam versi lebih terbatas yang bekerja dengan menerima masukan berupa obyek yang bukan satu dimensi melainkan dua dimensi. Dasar utama dari lapisan konvolusional adalah untuk mengambil fitur-fitur dasar seperti tepi, sudut, dan titik-titik akhir, dan kemudian menggabungkannya menggunakan lapisan-lapisan untuk mendapatkan fitur yang lebih kompleks dan nantinya dapat digunakan untuk mendeskripsikan suatu obyek secara keseluruhan. Cara kerja dari arsitektur ini secara garis besar adalah sebagai berikut: Lapisan pertama f1( ) mengandung fitur-fitur dasar, lalu lapisan kedua f2(f1( ))merupakan fungsi bagi fitur-fitur dari lapisan pertama, lalu lapisan ketiga f3(f2(f1( ))) merupakan fungsi bagi fitur-fitur dari lapisan kedua, dan seterusnya. Arsitektur seperti ini sangat berguna karena fitur-fitur yang lebih mendalam (*high-level features*) dapat diperoleh untuk mendeskripsikan suatu obyek dengan fitur yang akan lebih sedikit jika diproses secara linear.

Lapisan Penyatu (Pooling Layer)

Lapisan penyatu atau *pooling layer* merupakan lapisan yang berfungsi untuk memperkecil ukuran spasial secara progresif dari suatu representasi obyek untuk mengurangi jumlah parameter dan komputasi pada jaringan. Lapisan ini beroperasi secara mandiri terhadap tiap-tiap peta fitur. Penambahan lapisan penyatu setelah suatu lapisan konvolusional merupakan pola yang umum digunakan jika suatu lapisan akan digunakan kembali sekali maupun beberapa kali pada model jaringan tersebut. Lapisan penyatu ini beroperasi terhadap setiap peta fitur untuk membuat suatu set peta fitur yang baru.

Dalam melakukan *pooling*, berarti harus menentukan juga operasi *pooling* yang hendak digunakan, seperti menentukan filter yang akan diaplikasikan pada peta fitur. Ukuran dari operator pooling ini harus lebih kecil dari ukuran peta fiturnya, secara umum besarnya adalah 2x2 pixel dengan langkah sebesar 2 pixel. Hal ini berarti lapisan penyatu akan memperkecil ukuran dari setiap peta fitur sebesar factor dari 2. Sebagai contohnya, jika lapisan penyatu berukuran 2x2 pixel diterapkan pada suatu peta fitur yang berukuran 6x6 pixel, maka keluaran dari lapisan ini

adalah suatu peta fitur sebesar 3x3.



Gambar 2. 4. Ilustrasi Lapisan Penyatu

Operasi dari penyatunya juga perlu ditentukan. Terdapat dua fungsi yang paling umum digunakan sebagai operator *pooling*, yaitu *Average Pooling*, yang mengkalkulasikan nilai rata-rata dari setiap potongan (seukuran filternya) pada peta fitur, dan juga *Max Pooling*, yang mengkalkulasikan nilai maksimum dari setiap potongan pada peta fitur.

Lapisan Terhubung-Penuh (Fully-Connected Layers)

Lapisan terhubung-penuh merupakan lapisan terakhir dari suatu model jaringan syaraf konvolusional. Lapisan ini beroperasi sebagai Multi-Layer Perceptron (MLP) yang biasanya memiliki 2 atau 3 lapisan tersembunyi dan 1 lapisan klasifikasi. Sifat dari MLP ini membuat lapisan ini berfungsi dengan sangat baik sebagai pemberi aproksimasi, hanya dengan 2 lapisan tersembunyi dapat memberikan aproksimasi bagi fungsi apapun. Masukan dari lapisan ini berasal dari semua neuron pada lapisan-lapisan sebelumnya (lapisan pooling atau lapisan konvolusional). Dengan

kata lain, setiap neuron dari lapisan sebelumnya terhubung dengan setiap neuron pada lapisan tersembunyi pertama pada lapisan terhubung-penuh ini. Setiap keluaran dari lapisan tersembunyi pertama tersebut kemudian terhubung dengan setiap neuron dari lapisan tersembunyi selanjutnya, hal inilah yang menyebabkan lapisan ini diberi nama lapisan terhubung-penuh. Keluaran dari lapisan tersembunyi yang terakhir kemudian akan diberikan kepada lapisan klasifikasi yang terakhir. Ukuran dari lapisan ini bergantung kepada banyaknya jumlah kelas pada jaringan syaraf. Sebagai contohnya, jika jaringan berfungsi melakukan prediksi terhadap 5 jenis barang maka ukuran dari lapisan ini harus sebesar 5.

Mengacu kepada satu buah neuron yang disebut dengan J, pada MLP dapat dideskripsikan secara matematis dengan persamaan berikut ini

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

$$y_j(n) = \phi_j(v_j(n)))$$

Dimana yj(n) merupakan keluaran dari neuron j pada iterasi ke-n. Weight antara neuron i (dari layer sebelumnya) dan neuron j (pada layer saat ini) adalah wji. Sementara itu, yj(n) merupakan jumlah dari semua neuron pada layer sebelumnya. ϕj(·) adalah fungsi aktivasi dari neuron j dan m adalah total banyaknya masukan atau neuron dari lapisan sebelumnya. Bias dari lapisan ini dituliskan sebagai i=0 dan y0(n) = 1.

2.1.4. Akurasi Top-N

Akurasi pada jaringan syaraf adalah jumlah prediksi yang benar yang dikeluarkan suatu model. Pada tugas akhir ini, akan digunakan dua jenis akurasi, yang pertama adalah akurasi top-1 dan yang kedua adalah akurasi top-2. Akurasi top-1 adalah akurasi pada umumnya, yaitu banyaknya prediksi yang benar yang dikeluarkan oleh suatu model. Tentu saja, prediksi yang dimaksud disini adalah label dengan probabilitas tertinggi diantara label lainnya. Sementara itu, akurasi top-2 adalah seberapa seringnya label asli berada pada tingkat probabilitas kedua teratas diantara label lainnya. Misalnya, jika pada suatu model klasifikasi

citra yang terdiri dari 5 kelas; anjing, kucing, kelinci, tikus, dan beruang. Ketika diuji pada 5 buah citra, model tersebut mengeluarkan hasil sebagai berikut:

No. Citra	Label Asli	3 Label dengan probabilitas tertinggi
1	Anjing	Kucing, Anjing, Beruang
2	Kucing	Kucing, Beruang, Tikus
3	Beruang	Tikus, Beruang, Kucing
4	Tikus	Tikus, Anjing, Kelinci
5	Kucing	Kelinci, Tikus, Beruang

Tabel 2. 1. Contoh Hasil Klasifikasi Multi-Class

Maka berdasarkan hasil tersebut, dapat dilihat bahwa label asli muncul pada label dengan probabilitas tertinggi sebanyak 4/5. Hal ini berarti akurasi top-2 dari model tersebut adalah sebesar 80%.

2.1.5. *Confusion Matrix*

Confusion matrix adalah suatu tabel yang sering kali digunakan untuk mendeskripsikan performansi dari suatu model klasifikasi terhadap suatu set data test, yang nilai atau label asli nya sudah diketahui. Dengan tabel ini kita dapat memvisualisasikan performansi dari suatu algoritma, dan memudahkan dalam mengidentifikasi kebingungan antar kelas/kategori, contohnya seperti kelas manakah yang paling sering salah diprediksi, dan sebagainya. Secara umum, confusion matrix memiliki bentuk sebagai berikut:

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Tabel 2. 2. *Confusion Matrix* Klasifikasi Biner

Dengan penjelasan berikut ini;

- *True Positive* (TP) : observasi positif, dan terprediksi positif.
- False Negative (FN) : observasi positif namun terprediksi negatif.
- True Negative (TN) : observasi negatif dan terprediksi negatif.
- False Positive (FP) : observasi negatif namun terprediksi positif.

Dari tabel tersebut kemudian kita dapat menghitung berbagai macam representasi performansi yang lain seperti *precision* dan *recall* dengan menggunakan rumus-rumus dibawah ini.

$$\textbf{Precision} = \frac{TP}{TP + FP} \qquad \textbf{Recall} = \frac{TP}{TP + FN}$$

Namun bentuk tabel seperti itu hanya berlaku bagi klasifikasi dengan 2 kelas saja, negatif dan positif, atau sering kali disebut dengan klasifikasi biner. Pada klasifikasi dengan kelas lebih dari 2, *multiple class classification*, bentuk dari *confusion matrix* adalah sebagai berikut:

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

Tabel 2. 3. *Confusion Matrix* Klasifikasi *Multi-Class*

Dimana pada tabel tersebut, sumbu Y merupakan kelas terprediksinya dan sumbu X merupakan kelas aslinya. Dalam klasifikasi pada lebih dari 2 kelas ini, untuk menghitung TP, FN, TN, dan FP nya adalah dengan menghitung nilai-nilai tersebut untuk setiap kelasnya. Contohnya dengan menggunakan tabel 3.1 kita menghitung nilai TP, FN, TN, dan FPnya maka akan didapatkan hasil berikut.

- TP = 7
- TN = (2+3+2+1) = 8
- FP = (8+9) = 17
- FN = (1+3) = 4

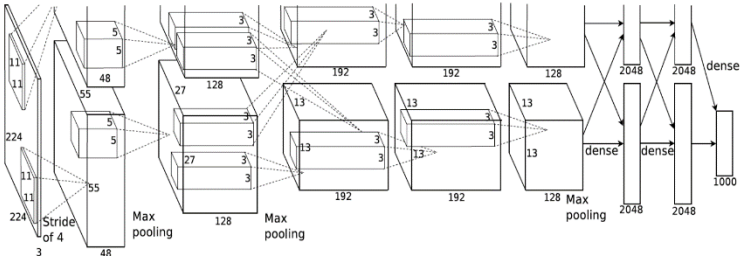
Dengan metode seperti itu baru kemudian kita dapat menghitung nilai *precision* maupun *recall*nya.

2.2. Tinjauan Pustaka

Pada subbab ini akan dipaparkan beberapa hasil penelitian sebelumnya yang memuat penjelasan dari model jaringan syaraf konvolusional yang hendak digunakan, serta perangkat-perangkat maupun kebutuhan tambahan bagi tugas akhir ini.

2.2.1 AlexNet

Pada tahun 2012, AlexNet[3] berhasil mengungguli kompetitornya dan memenangkan kompetisi ImageNet karena mampu memperkecil persentase kesalahan top-5 yang tadinya sebesar 26% menjadi 15.3%. Arsitektur asli dari AlexNet terdiri dari 8 lapisan: 5 lapisan konvolusi dan 3 lapisan terhubung (*fully-connected*). Jaringan ini memiliki arsitektur yang sangat mirip dengan jaringan LeNet[4] oleh Yann LeCun, dkk namun AlexNet bersifat lebih dalam, dengan jumlah filter per lapisan yang lebih banyak, dan lapisan konvolusional yang bertumpuk. AlexNet terdiri dari 11x11, 5x5, 3x4, konvolusi, *maxpool, dropout*, augmentasi data, aktivasi ReLu, dan SGD dengan momentum. Pada setiap akhir lapisan konvolusi dan lapisan terhubung pada jaringan ini selalu dilanjutkan dengan aktivasi jenis ReLu.



Gambar 2. 5. Arsitektur AlexNet [3]

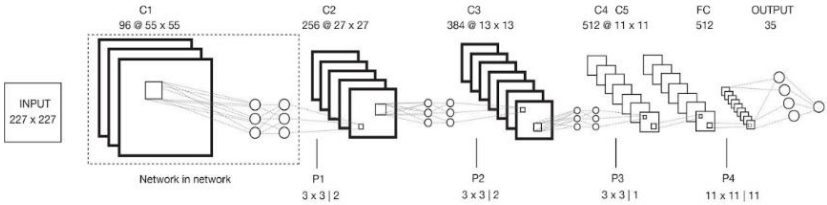
AlexNet dilatih selama 6 hari berturut-turut menggunakan 2 buah GPU Nvidia Geforce GTX 580. Walaupun menurut aslinya AlexNet bersifat sangat kompleks dan untuk melatihnya membutuhkan waktu yang sangat lama dan kebutuhan GPU yang besar, namun saat ini sudah semakin banyak kita temukan berbagai macam model jaringan yang menyederhanakan AlexNet ini. Salah satunya adalah *mydatahack*[5], yang membuat jaringan AlexNet yang telah disederhanakan menggunakan Keras. *Mydatahack* membuat beberapa perubahan yaitu menghilangkan paralelisme dari AlexNet dan menggunakan model sekuensial sebagai gantinya. Contohnya adalah pada lapisan konvolusinal AlexNet sebenarnya, terdiri dari 2 lapisan dengan 48 neuron tiap lapisannya. Namun pada versi ini, kedua lapisan tersebut digabungkan menjadi 1 lapisan dengan 98 neuron. Selain itu, jika pada AlexNet semulanya memiliki normalisasi pada beberapa lapisannya, pada versi ini ditambahkan normalisasi pada setiap akhir lapisan dan juga *dropout* di antara lapisan terhubung untuk mengurangi *overfitting.*

2.2.2 DeepSpace

DeepSpace[6] merupakan model jaringan yang dibuat oleh Zhang, dkk dalam studi kasus mengenai rekognisi dalam ruangan menggunakan jaringan syaraf konvolusional. Zhang, dkk melakukan studi kasus tersebut pada gedung kampus *Massachusetts Institute of Technology* (MIT) untuk melihat kemampuan model untuk mengenali ruangan-ruangan pada gedung tersebut menggunakan indera visual. Model DeepSpace ini

dengan referensi utama AlexNet dan NIN[7,]. Jaringan ini memiliki 7 lapisan dengan *weight*; 3 lapisan pertama merupakan lapisan MlpConv (Multi Layer Perceptron Convolution) yang berfungsi untuk mempelajari fitur-fitur yang bersifat non-linear. Dua lapisan berikutnya merupakan lapisan konvolusional biasa, dan lapisan setelahnya adalah lapisan terhubung. Output dari layer ini kemudian diberikan kepada lapisan *softmax* yang kemudian akan menghasilkan 35 label kelas.

Lapisan MLPConv pertama merupakan filter bagi citra berukuran 227x227x3 dengan 96 *kernel,* yang setiapnya berukuran 11x11 dan menggunakan lompatan sebesar 4. Hasil fitur yang didapatkan kemudian diberikan kepada lapisan max pooling berukuran 3x3 yang akan menghasilkan peta fitur berukuran 55x55. Proses ini berlanjut hingga pada layer kelima, yang pada akhirnya akan menghasilkan 512 peta fitur dengan ukuran 11x11, sebelum kemudian diberikan pada lapisan terhubung yang akan memberikan keluaran berupa prediksi dari kelas yang ada.



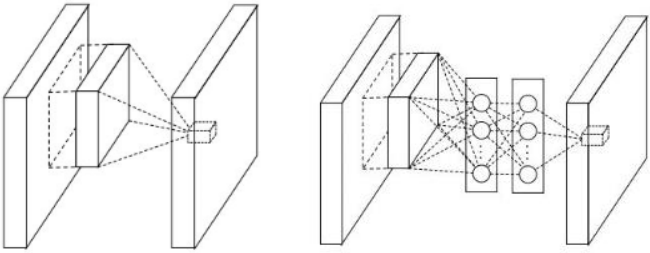
Gambar 2. 6. Arsitektur DeepSpace

Performansi dari model DeepSpace pada studi kasus yang dilakukan memiliki hasil yang sangat bagus. Ketika dibandingkan dengan model AlexNet dan NIN, model ini mampu memberikan akurasi yang lebih tinggi yaitu pada 81.72%.

2.2.3 Multi-Layer Perceptron Convolutional Layer

Multi-Layer Perceptron Convolutional Layer merupakan salah satu jenis lapisan yang digunakan pada penelitian Network in Network[7]. Lapisan ini merupakan suatu lapisan konvolusional yang kemudian

dilanjutkan dengan lapisan terhung pernuh dan fungsi aktivasi non-linear. Lin dkk menyatakan bahwa dengan menggunakan lapisan seperti ini, dapat meningkatkan pembelajaran model terhadap representasi-representasi yang bersifat non-linear.



Gambar 2. 7. ConvLayer (kiri) dan MLPConvLayer (kanan)

2.2.4 Intel RealSense D435i

RealSense D435i merupakan kamera stereo yang dikembangkan oleh Intel dan mampu menangkap citra RGB, depth, dan infrared. Kamera ini berukuran 90mx25mx25m dan dilengkapi dengan 2 buah modul IR stereo untuk menangkap citra infrared dan juga depth. Resolusi dari citra RGB yang dihasilkan kamera ini adalah sebesar 1920 × 1080 dan *frame rate* sebesar 30fps. Sementara itu, resolusi dari citra depth yang dihasilkan dapat mencapai 1280 × 720 dengan *frame rate* yang dapat mencapai 90fps.

Intel RealSense D435i ini dilengkapi oleh suatu library yang disebut dengan LibRealSense SDK 2.0. LibRealSense SDK 2.0 merupakan suatu perpustakaan lintas platform yang didesain bagi pengguna atau developer untuk mengimplementasikan berbagai macam aplikasi dengan menggunakan kamera RealSense seri SR300 dan D4000. Dengan menggunakan SDK ini, dapat dilakukan *streaming* citra RGB dan juga depth. Perpustakaan ini juga dilengkapi dengan perintah-perintah untuk menjalankan streaming sintetik (*pointcloud*, pelurusan citra depth dan RGB dan sebaliknya), dan juga suatu aplikasi yang mendukung

dilakukannya perekaman atau pemutaran ulang hasil rekaman kamera.

Gambar 2. 8. Kamera Intel RealSense D435i

Fitur	**Lingkungan Penggunaan:** Luar dan dalam ruangan **Teknologi Sensor:** Global Shutter, 3µm × 3µm pixel	**Rentang Maksimum:** Sekitar 10 meter. Bergantung pada kalibrasi, kondisi pencahayaan, serta tempat.
Depth	**Teknologi:** Active IR Stereo **Jarak Minimum Depth (Min-Z):** 0.105 m	**Depth Field of View (FOV):** 86° × 57° (±3°) **Resolusi Keluaran Depth:** Up to 1280 × 720 **Kecepatan Frame:** Up to 90 fps
RGB	**Resolusi:** 1920 × 1080 **Kecepatan Frame:** 30 fps	**RGB Sensor FOV (H × V × D):** 69.4° × 42.5° × 77° (±3°)
	Modul Kamera: Intel RealSense	**Prosesor Visi:** Intel RealSense Vision Processor D4

Komponen Utama	Module D430 + RGB Camera	
Fisik	**Panjang × Lebar × Tinggi:** 90 mm × 25 mm × 25 mm	**Konektor:** USB-C* 3.1 Gen 1*

Tabel 2. 4. Spesifikasi Kamera Intel RealSense D435i

2.2.5 Keras

Keras merupakan suatu API (*Application Programming Interface*) tingkat tinggi yang ditulis dalam bahasa pemrograman Python dan berjalan diatas platform yang disebut dengan TensorFlow. Keras didesain untuk membantu pengguna atau developer dalam membangun suatu program deep learning.

Gambar 2. 9. Logo Keras

2.2.6 TensorFlow

Tensorflow, dibuat dan dikembangkan oleh tim Google, merupakan suatu perpustakan untuk komputasi numerik dan machine learning berskala besar. Tensorflow menggabungkan model machine learning dan deep learning dan algoritma-algoritmanya dan membuatnya sangat berguna. Tensorflow menggunakan Python untuk menyediakan front-end API untuk membangun aplikasi-aplikasi pada framework tersebut, dan dapat juga mengeksekusikan aplikasi-aplikasi tersebut menggunakan C++. Tensorflow dapat melatih serta menjalankan jaringan syaraf untuk klasifikasi, rekognisi, jaringan syaraf recurrent, natural language processing, dan berbagai aplikasi jaringan syaraf lainnya.

2.2.7 OpenCV-Python

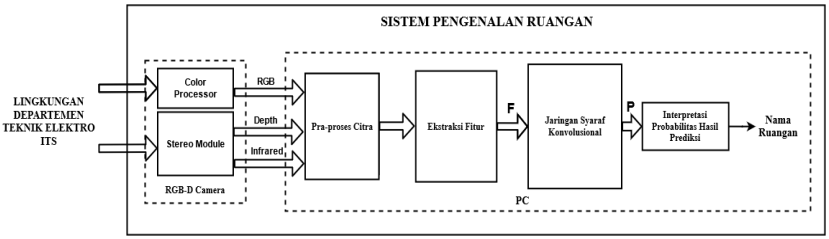
OpenCV adalah suatu perpustakaan untuk visi computer dan perangkat lunak machine learning yang bersifat terbuka. OpenCV dibuat untuk menyediakan infrastruktur bagi aplikasi-aplikasi visi computer dan untuk mengakselerasi pemakaian pengelihatan mesin pada produk-produk komersil. OpenCV-python merupakan perpustakaan openCV namun terikat dengan bahasa pemrograman python. OpenCV-Python ini menggunakan Numpy, yaitu suatu perpustakaan untuk menjalankan operasi-operasi numerik yang sudah sangat dikenal dan cukup teroptimisasi. Semua struktur array pada OpenCV akan terkonversi menjadi Numpy array. Hal ini dapat memudahkan pengguna untuk mengintegrasikan perpustakaan-perpustakaan lainnya yang menggunakan numpy, seperti scipy dan Matplotlib.

BAB 3
PERANCANGAN SISTEM

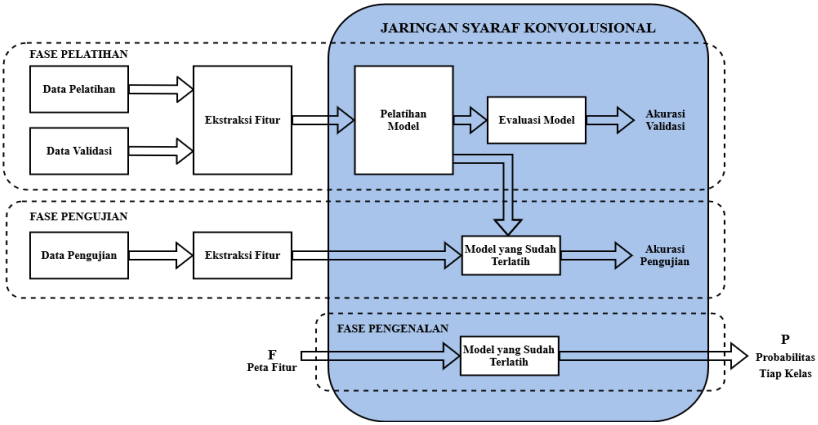
Pada bab ini akan dibahas perancangan dari sistem pengenal ruangan berbasis *convolutional neural network* menggunakan data visual. Hal ini meliputi tahap-tahap perancangan, perangkat-perangkat yang digunakan, serta penjelasan program dari sistem utama maupun jaringan syaraf konvolusional.

Perancangan sistem ini dapat dikategorikan menjadi 2 bagian, pemilihan model jaringan syaraf konvolusional yang akan diimplementasikan serta pembuatan program utama.

Pemilihan dilakukan antara 3 jenis jaringan syaraf konvolusional, model jaringan AlexNet, DeepSpace, serta satu buah model yang dirancang oleh penulis (selanjutnya akan disebut dengan model-A). Demi memilih model mana yang akan diimplementasikan, pertama dilakukan pengambilan data RGB, Infrared, dan Depth dari 5 buah ruangan yang akan dikenali sistem. Data-data ini akan membentuk suatu database yang terdiri dari data pelatihan, data validasi, dan data pengujian, dan akan digunakan sebagai input model jaringan. Penjelasan lebih dalam mengenai data yang digunakan akan dibahas pada subbab 3.1. Kedua, dilakukan pelatihan jaringan dengan input data yang berbeda-beda. Ketiga model dilatihkan sebanyak enam kali meggunakan jenis data yang berbeda-beda. Secara berurutan adalah menggunakan jenis data RGB, Infrared, Depth, gabungan RGB dan Infrared, gabungan RGB dan Depth, dan gabungan Infrared dan Depth, sehingga didapatkan total sebanyak 18 model yang sudah terlatih pada data yang berbeda-beda. Ketiga, dilakukan pengujian seluruh model yang sudah terlatih dari tahap 2. Dari pengujian 18 model tersebut kemudian akan dipilih model yang memiliki tingkat akurasi tertinggi untuk diimplementasikan ke sistem akhirnya. Hal ini dilakukan agar sistem pada akhirnya dapat berjalan dengan tingkat akurasi semaksimal mungkin, dengan melihat model jaringan manakah yang dapat melakukan pengenalan terbaik, dan dengan jenis data manakah model jaringan dapat paling mudah mengenali ruangan. Penjelasan mengenai pelatihan jaringan syaraf konvolusional akan dibahas lebih dalam pada subbab 3.2.



Gambar 3. 2. Diagram Blok Sistem



Gambar 3. 1. Diagram Blok Jaringan Syaraf Konvolusional

Pada bagian perancangan yang kedua, pembuatan program utama, program yang dibuat adalah program yang mampu membuat sistem berjalan dengan skenario sebagai berikut: 1) Prototipe robot yang sudah memiliki kamera digerakkan menuju/diletakkan pada suatu ruangan, 2) Ketika program dijalankan, sistem akan memulai mengambil informasi dari ruangan letak robot berada dengan mengambil citra (dapat berupa RGB, Depth, ataupun NIR) dari kamera. 3) Kemudian citra tersebut diolah, yaitu dengan dilakukan perubahan ukuran menjadi 227x227 pixel. 4) Citra tersebut dijadikan masukan bagi suatu model jaringan syaraf

konvolusional, 5) Model jaringan akan melakukan proses prediksi dan akan mengeluakan keluaran berupa probabilitas nama ruangan, 6) Nama ruangan dengan probabilitas terbesar akan ditampilkan pada layar sebagai prediksi nama ruangan. Penjelasan lebih dalam mengenai program utama akan dijelaskan pada subbab 3.3.

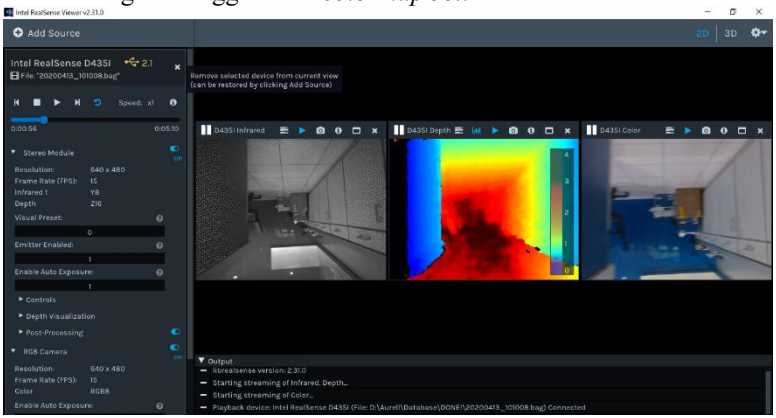
3.1. Data

Sebelum melatih setiap model jaringan, diperlukan data-data yang akan digunakan sebagai masukannya, sehingga dibutuhkan pengambilan data. Data yang diambil adalah berupa data RGB, depth dan infrared dari 5 buah ruangan di departemen Teknik Elektro ITS, yaitu ruangan C215, ruangan C216, ruangan lobby tata usaha S-1, ruangan lobby tata usaha pasca sarjana, serta ruangan pertemuan sekretariat Departemen Teknik Elektro ITS. Data-data tersebut merupakan data yang diambil sendiri dikarenakan kemiripan dari ruangan-ruangan yang dijadikan sebagai kategori (semua ruangan masuk dalam kategori ruangan kantor), sehingga pada pembuatan database serta pelatihan tidak dapat mengambil dari sumber yang sudah ada ataupun melakukan *transfer-learning*.

Pengambilan data dilakukan beberapa kali. Beberapa kali pertama adalah dengan cara merekam video RGB, depth, dan NIR menggunakan kamera Intel Real Sense d435i dengan menggunakan kamera yang dipegang tangan pada ketinggian sekitar 60cm dari permukaan tanah, lalu dibawa mengelilingi 5 ruangan yang telah disebutkan sebelumnya. Pada kali lainnya, pengambilan data dilakukan menggunakan kamera yang sudah terpasang pada suatu prototipe robot. Robot digerakkan ke lima ruangan yang sudah disebutkan sebelumnya, dan ketika sudah sampai di ruangan yang dituju, perekaman dimulai. Perekaman-perekaman ini dilakukan dengan menggunakan aplikasi bawaan dari kamera IntelRealSense yang disebut dengan Intel RealSense Viewer, dimana video yang terekam akan tersimpan pada file jenis .bag.

Perekaman video adalah dengan kecepatan 15 *frames per second,* dan resolusi 640x480 pixel, baik itu pada modul strereo maupun pada *color imaging*. Dari kedua modul stereo yang ada pada kamera Intel RealSense d435i, modul yang digunakan untuk mengambil citra infrared adalah

modul kiri, atau disebut dengan Y8. Sementara itu pemrosesan data depth adalah dengan menggunakan *colormap Jet.*



Gambar 3. 3. Tampilan Intel RealSense Viewer

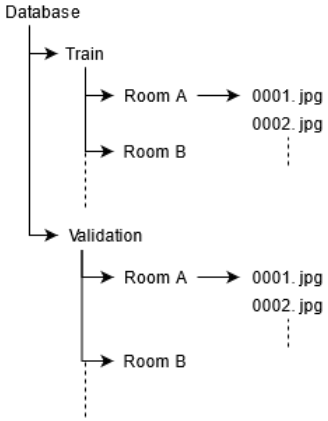
File jenis .bag ini kemudian diekstraksikan *frame-per-frame*nya menggunakan perintah dari *library* RealSense SDK 2.0 yang disebut dengan “rs-convert”. Perintah ini dijalankan pada Command Prompt PC dengan contoh sebagai berikut:

```
C:\Users\Aurell>rs-convert.exe -i <directoryfile>/file.bag -p <directory img>
```

Gambar 3. 4. Perintah rs-convert

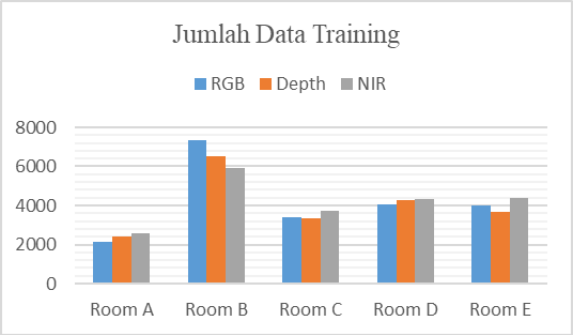
Dengan menjalankan perintah tersebut kemudian didapatkan citra RGB, Infrared, dan depth yang berukuran 640x480 pixel. Setiap citra tersebut kemudian diubah ukurannya menjadi ukuran yang sesuai dengan kebutuhan masukan model jaringan, yaitu 227x227 pixel. Seluruh citra yang didapatkan kemudian dipisahkan menjadi dua bagian, dataset training (digunakan pada saat pelatihan model jaringan) dan dataset testing (digunakan pada saat pengujian). Citra-citra tersebut juga dipisahkan berdasarkan ruangannya menjadi 5 kategori ruangan, yang pada kode program secara berurutan disebut dengan “RoomA”, “RoomB”, “RoomC”, “RoomD”, dan “RoomE”. Semua data-data ini kemudian

disusun dengan menggunakan format database Keras, yaitu sebagai berikut:

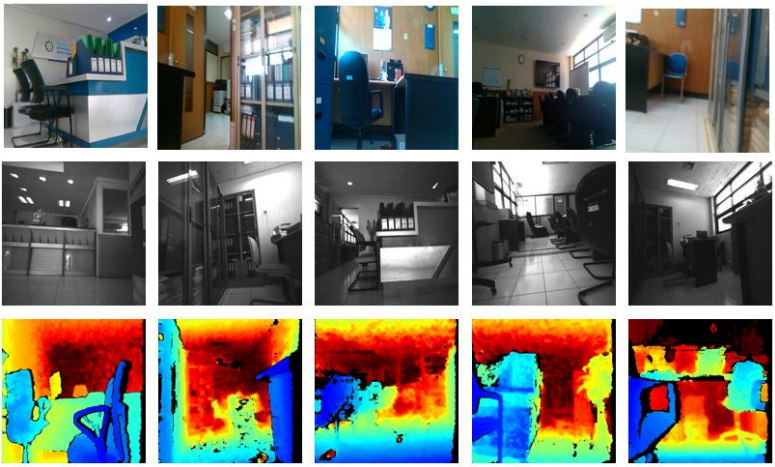


Gambar 3. 5. Format Database Keras

Pada akhirnya didapatkan total sebanyak 87.600 citra secara keseluruhan, dengan 29.200 citra untuk setiap jenis data RGB, Infrared, dan Depth. Sebanyak 75% dari data (65.700 citra) digunakan untuk pelatihan serta sebanyak 25% dari data (21.900 citra) untuk pengujian.



Tabel 3. 1 Grafik Jumlah Data per Kategori



Gambar 3. 6. Contoh Citra RGB, infrared, dan depth yang digunakan

3.2. Pelatihan Jaringan

Seperti yang sudah dijelaskan sebelumnya, pelatihan jaringan dilakukan pada tiga jenis model jaringan syaraf konvolusional – AlexNet, DeepSpace, dan Model-A. Pelatihan jaringan dilakukan sebanyak 6 kali untuk setiap jenis model jaringan, yaitu menggunakan jenis data yang berbeda-beda sebagai masukannnya. Secara berurutan adalah menggunakan data RGB, data infrared, data depth, gabungan RGB dan Infrared, gabungan RGB dan depth, dan gabungan Infrared dan Depth. Perangkat yang digunakan untuk melakukan pelatihan serta pengujian jaringan syaraf konvolusi ini adalah notebook dengan sistem Windows 10 64-bit, dengan prosesor Intel Core i7 CPU yang terpasang pada 2.50GHz dan 12GB RAM dan juga beberapa model jaringan dilakukan pada *Amazon Web Service SageMaker.* Untuk perangkat lunak yang digunakan adalah bahasa pemrograman Python 3.7.6 dan dilengkapi dengan library Keras, TensorFlow, dan Numpy.

3.2.1. Pelatihan Jaringan dengan 1 Jenis Data

Pelatihan jaringan dengan 1 jenis data adalah pelatihan model jaringan dengan menggunakan hanya data RGB, hanya data infrared, dan hanya data depth. Pelatihan ini dilakukan dengan *epoch* sebanyak 30 dan *learning rate* sebesar 0.1, untuk semua model jaringan.

AlexNet

Model Alex-Net[3] yang digunakan pada tahap ini merupakan versi yang telah disederhanakan dan diadaptasikan agar berjalan menggunakan Keras. Model ini menggunakan citra berukuran 227x227 px sebagai masukannya. Model ini terdiri dari 5 lapisan konvolusional yang diikuti oleh 3 lapisan terhubung penuh yang ditutup oleh lapisan aktivasi softmax untuk mengeluarkan outputnya. *Pseudocode* dari model ini adalah sebagai berikut:

```
model = Sequential()
# 1st Convolutional Layer
model.add(Conv2D(filters=96,input_shape=input_shape,
kernel_size=(11,11), strides=(4,4), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2),
padding='valid'))

# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1),
padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2),
padding='valid'))

# 3rd Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
padding='valid'))
```

```
model.add(Activation('relu'))
# 4th Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
padding='valid'))
model.add(Activation('relu'))
# 5th Convolutional Layer
model.add(Conv2D(filters=256,kernel_size=(3,3),strides=(1,1),
padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2),
padding='valid'))
```

Seperti yang dapat dilihat dari *pseudocode* tersebut, pada beberapa lapisan konvolusional selalu diakhiri oleh lapisan pooling dengan ukuran sebesar 2x2 dan dengan langkah = 2. Keluaran dari kelima lapisan konvolusional ini adalah peta fitur dengan ukuran 1 x 1 x 256. Dimana kemudian peta fitur tersebut diratakan menjadi 1 kanal saja dan diberikan kepada layer terhubung penuh, seperti pada pseudocode dibawah ini:

```
# Passing it to a Fully Connected layer
model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(9216, input_shape=(256,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))

# 2nd Fully Connected Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
```

```
# 3rd Fully Connected Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))

# Output Layer
model.add(Dense(4))
model.add(Activation('softmax'))
```

Pada setiap akhir lapisan terhubung penuh, selalu diikuti oleh *dropout.* Hal ini dilakukan untuk mengurangi *overfitting* pada jaringan. Setelah ketiga lapisan terhubung penuh, peta fitur diberikan kepada lapisan *output* berupa softmax. Dari model ini, total parameter yang digunakan adalah sebanyak 63.025.540 parameter, cukup banyak bagi suatu model yang berfungsi untuk mengenali ruangan kecil.

DeepSpace

Model DeepSpace merupakan implementasi Keras dari model yang dikembangkan oleh Zhang,dkk[6]. Model ini, seperti model AlexNet, menggunakan citra berukuran 227x227 pixel sebagai masukannya. Model ini terdiri dari 7 lapisan yang terdiri dari lapisan MLPConv, konvolusional, dan lapisan terhubung penuh. Untuk mengimplementasikan lapisan MLPConv pada Keras, dilakukan dengan cara menumpuk 2 lapisan konvolusional dengan ukuran filter yang sama dengan lapisan konvolusional sebelumnya dan ukuran kanal sebesar 1x1 seperti pada contoh dibawah ini:

```
#MLPConv Layer
model.add(Conv2D(filters=96,input_shape=input_shape,
kernel_size=(11,11), strides=(4,4), padding='valid'))
model.add(Activation('relu'))
model.add(Conv2D(96, (1, 1)))
model.add(Activation('relu'))
```

```
model.add(Conv2D(96, (1, 1)))
model.add(Activation('relu'))
```

Model DeepSpace ini menggunakan total parameter sebanyak 1.329.349, yang berarti model ini tidak "melihat" fitur secara sangat detil seperti pada model AlexNet sebelumnya, namun hal ini bukan berarti model tidak dapat memberikan akurasi yang lebih baik.

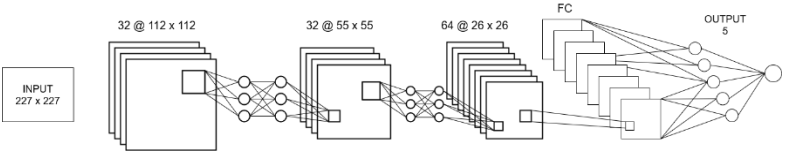
Model-A

Model-A merupakan jaringan syaraf konvolusional yang dirancang khusus untuk keperluan mengenali ruangan di Departemen Teknik Elektro ITS. Model-A ini menggunakan lapisan MLPConv pada 2 lapisan pertamanya, lalu diikuti dengan satu lapisan konvolusi, dan kemudian diikuti dengan lapisan terhubung dan fungsi aktivasi. Penggunaan MLPConv sebagai dua lapisan pertama diharapkan dapat mempermudah model dalam mempelajari fitur-fitur non-linear. Model ini menggunakan parameter sebanyak 2.800.037 dan menggunakan optimizer "rmsprop" pada penyusunan modelnya.

Model ini menggunakan citra berukuran 227x227 pixel sebagai masukannya. Hal ini dilakukan demi kemudahan dalam pemrosesan data dikarenakan kebutuhan ukuran masukan model lain yang juga berukuran 227x227 pixel. Selain itu dengan ukuran masukan yang lebih besar, waktu pelatihan juga akan menjadi lebih lama, namun dengan ukuran ini dianggap sudah menghabiskan waktu yang cukup seimbang dengan pembelajaran model.

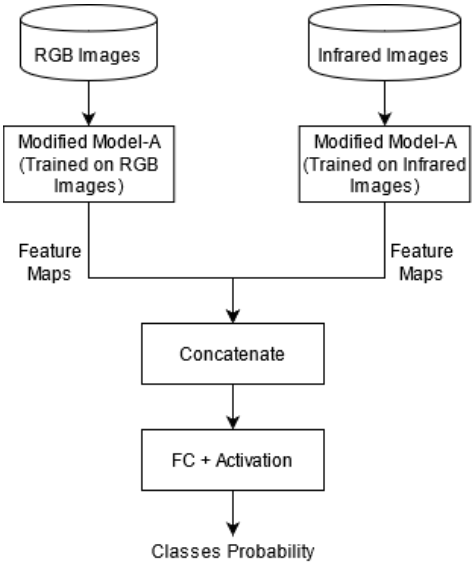
Konfigurasi dari model ini adalah sebagai berikut. Pada layer pertama, MLPConv memfilter image yang berukuran 227x277x3 tersebut dengan 32 kanal yang berukuran 3x3. Peta fitur yang didapatkan tersebut kemudian di pool (max pool berukuran 2x2) sehingga didapatkan 32 peta fitur berukuran 112x112. Hal yang sama dilakukan kembali pada layer kedua, yaitu layer MLPConv layer kedua, sehingga didapatkan 32 fitur peta berukuran 55x55. Kemudian fitur peta tersebut menjadi masukkan bagi lapisan konvolusional dengan filter sebanyak 64 kanal berukuran 3x3, dan dilanjutkan dengan max pool (berukuran 2x2) sehingga didapatkan output berupa 64 peta fitur berukuran 26x26. Lapisan berikutnya

merupakan lapisan fully-connected, dimana fitur map yang dihasilkan pada lapisan ini menjadi input bagi lapisan terakhir berupa aktivasi softmax. Output dari lapisan terakhir ini berupa *array* berisi probabilitas dari 5 label kelas.



Gambar 3. 7. Arsitektur Model-A

3.2.2. Pelatihan Jaringan dengan 2 Jenis Data



Gambar 3. 8. Contoh Arsistektur Model-A dengan Input RGB dan Infrared

Pada pelatihan dengan menggunakan gabungan 2 jenis data sebagai masukkannya, model dimodifikasi agar memiliki 2 cabang masukan. Lalu model ini menggunakan kembali 9 model yang sudah terlatih pada tahap sebelumnya. Model-model ini digunakan sebagai ekstraktor fitur bagi jenis data yang sesuai dengan pelatihannya. Hal ini dilakukan dengan cara memuat model itu kembali pada program dan menghapus lapisan terhubung penuh serta aktivasi yang terakhir, sehingga keluaran dari model bukan berupa prediksi seperti biasanya, namun berupa matriks fitur. Matriks fitur dari kedua cabang kemudian digabungkan lalu dijadikan masukan bagi lapisan terhubung penuh dan aktivasi untuk mendapatkan outputnya. Contoh arsitektur dari model dengan 2 masukan ini, dapat dilihat pada gambar 3.7., dimana digunakan citra RGB sebagai masukan cabang satu dan citra Infrared sebagai masukan cabang dua. Citra RGB menjadi masukan bagi model-A yang sudah dilatih dengan citra RGB, dan citra infrared menjadi masukan bagi model-A yang sudah dilatih dengan citra infrared. Kedua peta fitur yang didapatkan kemudian digabungkan dengan lapisan "*concatenate*", dan hasil peta fitur yang sudah digabungkan tersebut menjadi masukan bagi lapisan terhubung penuh dan aktivasi untuk mendapatkan keluaran berupa probabilitas tiap kelas.

Pada penulisan program untuk tahap ini, demi memanggil 2 jenis data yang berbeda, digunakan 2 buah generator. Generator untuk memanggil jenis data 1 dan generator untuk memanggil jenis data 2. Generator ini menggunakan "flow_from_directory"

Kedua generator tersebut kemudian dijadikan satu kesatuan dengan *pseudocode* berikut:

```
def train_gen_flow_for_two_inputs(X1, X2):
    genX1 = gen.flow_from_directory(X1,
                                    target_size=(img_width, img_height),
                                    batch_size=batch_size,
                                    class_mode='categorical')
    genX2 = gen.flow_from_directory(X2,
                                    target_size=(img_width, img_height),
                                    batch_size=batch_size,
                                    class_mode='categorical')
```

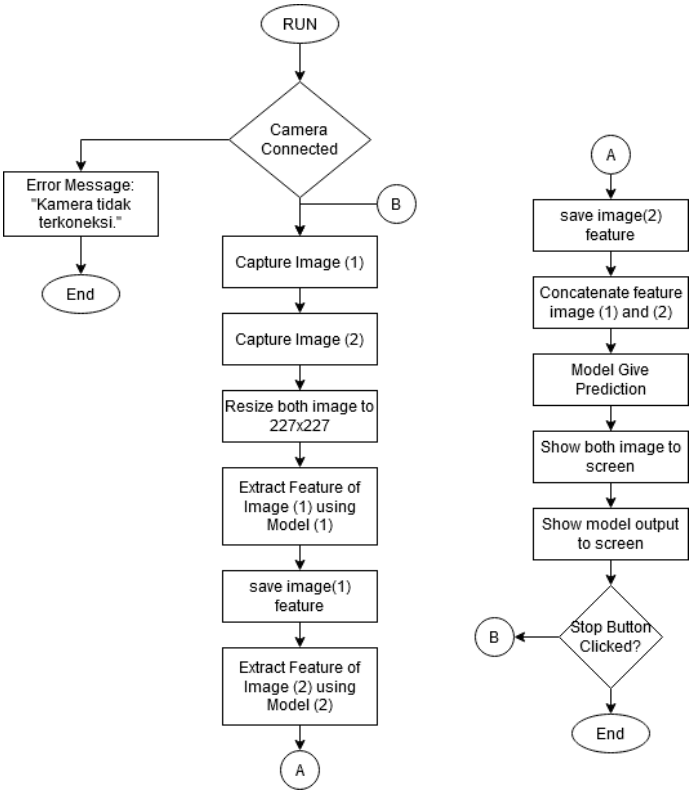
```
while True:
        X1i = genX1.next()
        X2i = genX2.next()
        yield [X1i[0], X2i[1]], X1i[1]
```

Dimana X1 merupakan direktori letak database jenis data 1 dan X2 merupakan direktori letak database jenis data 2. Generator data inilah yang akan secara langsung memberikan data untuk diberikan kepada masing-masing ekstraktor fitur. Masing-masing ekstraktor fitur kemudian akan mengeluarkan matriks fitur. Kedua matriks fitur ini akan memiliki dimensi yang sama karena kedua ekstaktor fitur merupakan jenis model yang sama, hanya saja dilatihkan pada jenis data yang berbeda. Jika kita misalkan masing-masing dimensi matriks fitur keluaran ekstraktor fitur adalah berukuran [i,j]. Kedua matriks ini digabungkan menggunakan layer.concatenate sehingga didapatkan matriks fitur berukuran [2i,j]. Matriks tersebut kemudian diteruskan kepada lapisan-lapisan berikutnya, yang adalah lapisan terhubung penuh dan aktivasi. Keluaran dari model secara keseluruhan adalah prediksi nama ruangan.

Sebagai contohnya, jika dilakukan pelatihan model-A menggunakan data gabungan antara RGB dan NIR, data RGB diberikan kepada ekstraktor fitur yang adalah model-A yang sudah dilatih pada data RGB sebelumnya untuk mendapatkan fiturnya. Hal yang sama dilakukan untuk data NIR-nya, diberikan kepada ekstraktor fitur yang adalah model-A yang sudah dilatih pada data NIR. Kedua fitur yang didapatkan ini kemudian digabungkan menggunakan lapisan *concatenate* yang akan menjadi masukan bagi lapisan terhubung dan aktivasi linear. Hal ini dilakukan terus menerus bagi gabungan data RGB dan NIR, gabungan data RGB dan depth, dan gabungan data depth dan NIR.

3.3. Pembuatan Program Utama

Sistem ini mengambil citra dari kamera Intel RealSense d435i dan kemudian mengubah ukurannya menjadi sebesar 227x227 px sebelum citra tersebut diberikan sebagai masukan bagi model jaringan syaraf konvolusional. Sebagai keluarannya, sistem akan mengeluarkan prediksi dari ruangan tersebut dan menampilkannya pada layar *notebook* beserta



Gambar 3. 9. Flowchart Program Utama

dengan citra yang terambil. Pada flowchart pada gambar 3.9., digambarkan alur dari program sistem pengenal ruangan, dengan penjelasan sebagai berikut:

- Image (1) = jenis data 1
- Image (2) = jenis data 2
- Model (1) = model yang sudah dilatih menggunakan jenis data 1

- Model (2) = model yang sudah dilatih menggunakan jenis data 2
- Model terakhir yang memberikan prediksi merupakan model yang terdiri dari lapisan terhubung penuh serta aktivasi, seperti yang sudah dijelaskan pada subbab 3.2.2.

Flowchart pada gambar 3.8 dinyatakan dalam program dengan bahasa pemrograman python 3.7.6 dan dilengkapi dengan library keras, pyrealsense2, cv2, json, dan h5py. Pertama program akan melakukan koneksi dengan kamera Intel RealSense yang terpasang pada robot dan dihubungkan kepada notebook menggunakan kabel type-C. Lalu sistem akan mengambil 2 buah citra, yaitu citra RGB serta infrared. Karena digunakan modul stereo kiri maka pada dituliskan jenis modul yaitu Y8. Sistem akan mengambil citra menggunakan perintah "get_frame", dan agar antara kedua frame RGB dan infrared sinkron, maka sebelum dilakukan perintah get_frame akan digunakan "pipeline.wait_for_frames". Dengan ini sistem akan menunggu hingga pipeline menerima kedua citra RGB dan infrared baru kemudian melanjutkan jalannya program.

```
pipeline = rs.pipeline()
config = rs.config()

config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
config.enable_stream(rs.stream.infrared, 1, 640,480, rs.format.y8, 30)

cfg = pipeline.start(config)
dev = cfg.get_device()
try:
    while True:
        frames = pipeline.wait_for_frames()
        color_frame= frames.get_color_frame()
        infrared_frame = frames.get_infrared_frame()

        color_image = np.asanyarray(color_frame.get_data())
        infrared = np.asanyarray(infrared_frame.get_data())
```

```
infrared_image = cv2.cvtColor(infrared, cv2.COLOR_GRAY2BGR)
```

Setelah mendapatkan kedua citra RGB dan infrared, citra tersebut kemudian diubah ukurannya dan dikonversikan menjadi dalam bentuk numpy array. Lalu program akan membuka kedua file .json yang merupakan struktur model yang digunakan, serta program akan memuat weight dan biasnya yang berasal dari file .h5.

```
with open(<directory file ,json>,'r') as f:
        model_json = json.load(f)

    model = model_from_json(model_json)
    model.load_weights(<directory file .h5>)
```

Hal ini dilakukan sebanyak 3 kali, yaitu untuk memuat model(1), model(2), dan model yang menggabungkan fitur tersebut. Lalu kemudian secara berkesesuaian, akan dijalankan model.predict untuk mengekstraksi fitur dari Image(1) dan Image(2). Hasil fitur tersebut kemudian diberikan kepada model yang menggabungkan fitur tersebut dengan code model.predict juga. Setelah itu, akan didapatkan keluaran berupa array berukuran 1 x 5 yang merupakan probabilitas tiap ruangan. Dari nilai-nilai pada array tersebut dicari nilai terbesarnya dan kemudian lokasi array terbesar menentukan label yang mana yang dimaksud. Pada program ini label diurutkan sebagai berikut:

```
label = (['C215', 'C216','TU S1','TU Pasca','R. Meeting])
```

Serta program untuk mencari lokasi array dengan nilai terbesar adalah sebagai berikut:

```
ind = np.unravel_index(output.argmax(), output.shape)
```

Kemudian label yang dimaksud akan dituliskan kepada layar

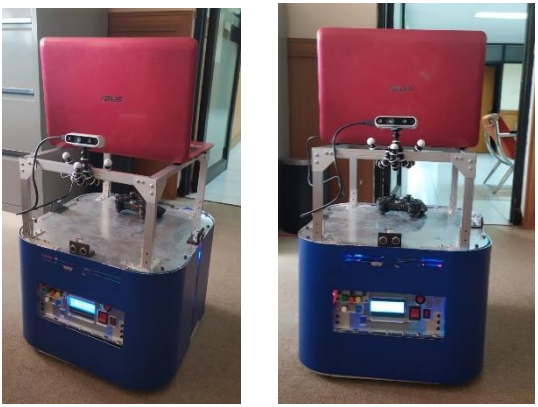
menggunakan code sebagai berikut:

```
cv2.putText(images, label[ind[1]], (60,80), cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255),2,cv2.LINE_AA)
cv2.namedWindow('RealSense', cv2.WINDOW_AUTOSIZE)
cv2.imshow('RealSense',images)
```

Dimana hasil akhir dari sistem adalah menampilkan kedua gambar yang menjadi masukan model, dan juga nama ruangan yang merupakan prediksi dari model. Tampilan dari sistem ini dapat dilihat pada gambar 3.8.

3.4. Realisasi Sistem

Perangkat yang digunakan untuk sistem pengenalan ruangan ini adalah *notebook* dengan sistem Windows 10 64-bit, dengan prosesor Intel Core i7 CPU yang terpasang pada 2.50GHz dan 12GB RAM. *Notebook* diletakan pada suatu prototipe robot yang sudah terpasang kamera Intel RealSense D435i pada ketinggian 60cm diatas permukaan tanah. Prototipe robot ini digerakan secara eksternal menggunakan *joystick.*



Gambar 3. 10. Realisasi Fisik Sistem

Sistem berjalan dengan cara menjalankan program menggunakan *command prompt* pada *notebook*. Kemudian saat program berjalan, pada layar akan tertampil gambar yang ditangkap oleh kamera serta nama ruangan yang terprediksi.

Gambar 3. 11. Tampilan *Interface* Sistem

BAB 4
PENGUJIAN DAN ANALISA

Pada bab ini akan dibahas metode pengujian yang dilakukan serta hasil dari pengujian tersebut. Dari setiap pembahasan juga akan meliputi analisa terhadap kegagalan ataupun keberhasilan dari suatu model ataupun sistem.

4.1. Metode Pengujian

Pengujian dilakukan dengan menggunakan *dataset testing* yang sudah dibagi di awal. *Dataset testing* ini menjadi masukan bagi program sistem yang didalamnya memanggil file berupa *weight* dan *bias* dari model yang sudah dilatih serta struktur dari model tersebut. Keluaran dari program tersebut adalah berupa persen akurasi top-1 dan top-2 dari prediksi model. Pengujian ini dilakukan pada sistem dengan model jaringan yang berbeda-beda, secara spesifik adalah 15 model jaringan yang sudah dijelaskan pada bab sebelumnya.

4.2. Pengujian Sistem

Pada subbab ini akan dilampirkan serta dijelaskan hasil pelatihan dan juga pengujian dari setiap sistem dengan tiap model jaringan. Hasil akurasi pelatihan merupakan akurasi model terhadap data validasi yang berjumlah 1000 (200 per kategori ruangan). Sementara hasil pengujian merupakan akurasi model terhadap data testing yang telah dijelaskan pada bab 2 dan dilakukan menggunakan metode yang telah dijelaskan pada subbab 4.1.

4.2.1. Sistem dengan Masukan 1 Jenis Data

Setelah melatih ketiga model pada 1 jenis data seperti yang telah dijelaskan pada bab sebelumnya, didapatkan hasil akurasi pelatihan seperti pada tabel 4.1. Dari tabel tersebut dapat dilihat bahwa model DeepSpace memiliki tingkat akurasi tertinggi pada 95.31% dan secara berdekatan diikuti oleh model-A yang hanya berbeda 0.41% dari akurasi model DeepSpace. Namun model AlexNet memiliki akurasi yang jauh

dibawah kedua model tersebut yaitu hanya sebesar 50.78%. Pada pelatihan menggunakan jenis data NIR, hasil akurasi ketiga model sangat mirip dengan hasil sebelumnya dimana DeepSpace menduduki peringkat pertama dan secara berdekatan diikuti oleh model-A dengan perbedaan sebesar 1.45%. Pada pelatihan menggunakan jenis data Depth, tingkat akurasi dari ketiga model menurun drastis jika dibandingkan dengan pada jenis data RGB dan NIR, dengan tingkat akurasi tertinggi adalah oleh model-A pada 90.72% diikuti oleh DeepSpace pada 86.28% dan AlexNet dengan hasil terburuk yaitu 29.41%.

Kemudian ketiga model yang telah dilatih tersebut diujikan. Pengujian ini adalah pengujian sistem yang menggunakan hanya data RGB, hanya data Infrared, dan hanya data Depth sebagai inputnya. Dapat dilihat pada hasil bahwa sistem yang menggunakan model AlexNet memiliki kesulitan terbesar dalam mengenali ruangan dibandingkan dengan sistem yang menggunakan model jaringan lainnya. AlexNet hanya mampu mencapai tingkat akurasi top-1 tertinggi sebesar 14.72% dan top-2 tertinggi sebesar 26.85%. Sementara itu model dengan akurasi top-1 tertinggi adalah model-A, dengan persentase akurasi sebesar 45.45% dengan menggunakan jenis data RGB, dan sebesar 44.02% dengan menggunakan jenis data Infrared. Model DeepSpace berada pada urutan berikutnya untuk akurasi top-1, namun memiliki akurasi top-2 yang sedikit lebih tinggi daripada model-A, dengan persentase akurasi tertinggi sebesar 69.82% pada model yang menggunakan hanya data RGB.

4.2.2. Sistem dengan Masukan 2 Jenis Data

Pada sistem dengan masukan 2 jenis data ini, ketiga model yang sudah dilatihkan kemudian menghasilkan akurasi validasi seperti pada tabel 4.3 Dari sini dapat dilihat bahwa tingkat akurasi tertinggi adalah pada model DeepSpace dengan jenis data RGB dan Infrared, yaitu sebesar 96.72%.

Kemudian ketiga model tersebut diujikan. Pengujian ini adalah pengujian sistem yang menggunakan kombinasi 2 jenis data antara 3 jenis data yang disediakan. Hasil akurasi Top-1 dan Top-2 dari pengujian ini dapat dilihat pada tabel 4.4.

Dari tabel tersebut dapat dilihat bahwa seperti pada hasil pengujian sebelumnya, model-A terbukti memiliki tingkat akurasi tertinggi, yaitu dengan akurasi top-1 tertinggi sebesar 48.23% pada model yang menggunakan kombinasi data RGB dan infrared. Pada pengujian ini juga dapat dilihat bahwa akurasi tertinggi dicapai ketika digunakan gabungan antara RGB dan infrared, dibandingkan dengan kombinasi data lainnya.

	Akurasi		
	RGB	NIR	Depth
AlexNet	50.78%	50.62%	29.41%
DeepSpace	95.31%	94.75%	86.28%
Model-A	94.90%	93.30%	90.27%

Tabel 4. 1. Akurasi Validasi Model dengan 1 Jenis Data

	RGB		Infrared		Depth	
	Top-1 Acc	Top-2 Acc	Top-1 Acc	Top-2 Acc	Top-1 Acc	Top-2 Acc
AlexNet	14.72%	26.85%	13.27%	21.33%	9.25%	19.79%
Deep Space	42.38%	69.82%	42.24%	62.65%	19.15%	33.12%
Model-A	45.45%	67.91%	44.02%	62.32%	22.93%	40.85%

Tabel 4. 2. Hasil Pengujian Model dengan 1 Jenis Data

	Akurasi		
	RGB+NIR	RGB + Depth	NIR+Depth
AlexNet	51.33%	36.72%	33.56%
DeepSpace	96.72%	94.75%	83.11%
Model-A	95.30%	93.30%	91.27%

Tabel 4. 3. Akurasi Validasi Model dengan 2 Jenis Data

	RGB+Infrared		Infrared+Depth		RGB+Depth	
	Top-1 Acc	Top-2 Acc	Top-1 Acc	Top-2 Acc	Top-1 Acc	Top-2 Acc
AlexNet	12.25%	17.67%	9.81%	12.13%	8.87%	11.23%
Deep Space	47.35%	72.81%	29.56 %	45.88%	30.76%	47.48%
Model-A	48.23%	71.23%	30.46 %	47.89%	31.19%	49.38%

Tabel 4. 4. Hasil Pengujian 2 Jenis Data

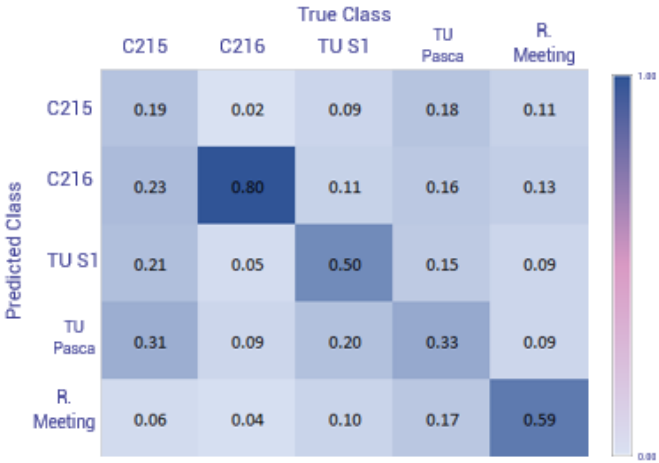
4.3. Analisa

Dari hasil pengujian-pengujian tersebut, dapat dilihat bahwa urutan tingkat akurasi model dari tertinggi hingga terendah secara berturut-turut untuk setiap jenis data adalah sebagai berikut: 1)Model-A, 2)DeepSpace, 3) AlexNet. Model AlexNet dengan jumlah parameter terbesar, sekitar 60.000.000 parameter, nampaknya kurang cocok untuk diimplementasikan sebagai sistem pengenal ruangan di lingkungan departemen teknik elektro ITS, hal ini dikarenakan model AlexNet bersifat sangat detil dan mengambil terlalu banyak parameter untuk dijadikan fitur, sehingga justru tidak mampu melihat obyek yang lebih besar, sehingga gagal dalam mencari korelasi dari data-data yang

diberikan. Dari seluruh pengujian juga didapatkan bahwa tingkat akurasi tertinggi adalah pada sistem yang menggunakan model-A dengan input gabungan data RGB dan Infrared. Menggunakan model tersebut, dilakukan analisa tambahan untuk mempelajari lebih dalam ruangan mana yang paling mudah serta paling sulit dikenali oleh sistem. Analisa ini dilakukan dengan membuat suatu *confusion matrix*.

Dari confusion matrix pada gambar 4.1. dapat kita lihat bahwa ruangan yang menghasilkan *true positive* tertinggi adalah ruangan C216, kemudian diikuti dengan ruangan meeting secretariat DTE. Sementara itu ruangan dengan *true positive* terendah adalah ruangan C215, dengan tingkat kebenaran hanya sebesar 0.19.

Berdasarkan confusion matrix ini kemudian dihitung nilai presisi model dalam mengidentifikasi tiap ruangan. Didapatkan dari tabel 4.3. bahwa nilai presisi tertinggi adalah pada ruang meeting secretariat DTE. Hal ini menyatakan bahwa ruang meeting secretariat DTE merupakan ruangan dengan fitur-fitur paling distingtif atau paling mudah dibedakan dibandingkan dengan ruangan-ruangan lainnya.



Gambar 4. 1. Confusion Matrix Model-A

Class	Precision
C215	31.80%
C216	55.64%
Lobby TU S1	49.69%
Lobby TU Pasca	32.93%
R. Meeting Sekre DTE	62.15%

Tabel 4. 5 Presisi Model-A Terhadap 5 Kategori Ruangan

Berdasarkan hipotesa penulis, hal ini dapat terjadi karena diantara ruangan-ruangan lainnya, ruang meeting sekretariat Departemen Teknik Elektro ITS memiliki kesamaan yang terbesar antar tiap gambar pada sudut yang berbeda. Dapat dilihat pada gambar 4.2. , walaupun kamera mengambil gambar dari sudut yang berbeda, terdapat kesamaan yang cukup mudah ditemukan, yaitu dinding ruangan. Pada kebanyakan data gambar dari ruangan ini, sekitar 80% mengandung gambar dinding tersebut. Hal ini dapat berperan cukup besar bagi sistem untuk mempelajari hubungan antar gambar pada sudut yang satu dengan lainnya, sehingga pada akhirnya sistem dapat mengenali ruangan ini dengan presisi tertinggi.

Sementara itu, pada ruangan C215, sedikit adanya kesamaan antar gambar yang tertangkap dari sudut yang berbeda-beda. Pada gambar 4.3. dapat dilihat bahwa ketika kamera menghadap sudut yang berbeda, barang-barang yang tertangkap kamera pun juga sangat berbeda. Hal ini menyebabkan sistem gagal mengetahui hubungan antar gambar-gambar tersebut sehingga salah dalam memprediksi.

Gambar 4. 3. Contoh gambar ruangan meeting sekretariat DTE yang diambil pada sudut yang berbeda-beda.

Gambar 4. 2. Contoh gambar ruangan C215 yang diambil pada sudut yang berbeda-beda.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa untuk mengenali ruangan di Departemen Teknik Elektro (DTE) ITS secara visual, dapat digunakan sistem yang berbasis jaringan syaraf konvolusional. Sistem terbukti mampu mengenali ruangan di Departemen Teknik Elektro dengan tingkat akurasi 48.23%. Sistem ini menggunakan model-A untuk melakukan prediksinya, dan menggunakan data masukan berupa citra RGB dan Infrared ruangan. Berdasarkan hasil analisa, sistem ini memiliki presisi tertinggi dalam mengenali ruang meeting sekretariat DTE yaitu sebesar 62.15%, serta memiliki presisi terendah dalam mengenali ruang C215 yaitu sebesar 31.80%. Hal ini menyatakan bahwa diantara kelima ruangan yang dijadikan sebagai *dataset*, ruangan meeting sekretariat DTE memiliki fitur-fitur yang paling mudah dibedakan oleh model, sementara itu ruang C215 paling sulit dibedakan oleh model. Berdasarkan hipotesa penulis, hal ini dikarenakan pada data ruangan meeting sekretariat DTE 80% mengandung gambar dinding ruangan. Hal ini menimbulkan adanya kesamaan yang jelas antara gambar yang diambil pada sudut yang satu dengan sudut lainnya. Sementara itu, pada ruangan C215 perbedaan sudut pengambilan gambar menghasilkan gambar yang sangat berbeda juga. Hal ini menyebabkan sistem gagal mengetahui hubungan antar gambar-gambar tersebut sehingga salah dalam memprediksi.

5.2. Saran

Penelitian ini dapat dikembangkan lebih lagi dengan meningkatkan akurasi sistem dalam mengenali ruangan-ruangan di DTE. Selain itu, ruangan-ruangan yang dikenali juga dapat diperbanyak lagi. Diharapkan pada penelitian berikutnya juga dapat mengembangkan sistem ini agar dapat mengenali bukan hanya pada tingkat ruangan saja tetapi juga area tertentu dalam ruangan.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Evan Ackerman, 2019, “Remotely Operated Home Robot Can Do Your Laundry”, https://spectrum.ieee.org/automation/robots/home-robots/remotely-operated-home-robot-can-do-your-laundry, diakses pada 5 Desember 2019 pukul 13.05

[2] Rich McCormick, 2016, “The Asus Zenbo is a ridiculous home robot for $599”, https://www.theverge.com/2016/5/30/11810102/asus-zenbo-home-helper-robot, diakses pada 5 Desember 2019 pukul 12.55

[3]A Krizhevsky, I Sutskever, GE Hinton. ImageNet Classification with Deep ConvolutionalNeural Networks. Advances in neural information processing systems, 2012.

[4] Othman, Kamal & Rad, Ahmad. (2019). An Indoor Room Classification System for Social Robots via Integration of CNN and ECOC. Applied Sciences. 9. 470. 10.3390/app9030470.

[5]Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 2015, arXiv:1409.1556.

[6]Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Patter Recognition, Boston, MA, USA, 7-12 June 2015; pp. 1-9.

[7] Zhang, Fan, Fabio Duarte, Ruixian Ma, Dimitrios Milioris, Hui Lin, and Carlo Ratti. “Indoor Space Recognition Using Deep Convolutional Neural Network: A Case Study at MIT Campus,” 2016.

[8] Sahdev, Raghavender, and Tsotsos John. Indoor Place Recognition System for Localization of Mobile Robots. Vol. 10.1109/CRV.2016.38. 53 60, 2016.

[8]OpenCV, “About”, https://opencv.org/about/, diakses pada 23 Juni 2020 pukul 14.00.

[9]OpenCV, “Introduction to OpenCV-Python Tutorials”, https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html , diakses pada 23 Juni 2020 pukul 03:18

[10]Keras, “About”, https://keras.io/about/, diakses pada 23 Juni 2020 pukul 13.00

[11]Martin Heller, 2019, “What is Keras? The deep neural network API explained”, https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html, diakses pada 23 Juni 2020 pukul 13.00
[12] Joydwip Mohajon, 2020, “Confusion Matrix for Your Multi-Class Machine Learning Model”, https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826, diakses pada 23 Juni 2020 pukul 13.10
[13]intelrealsense.com, “D435i”, https://www.intelrealsense.com/depth-camera-d435i/, diakses pada 23 Juni 2020 pukul 13.15

LAMPIRAN A

Pseudocode pelatihan Model AlexNet

```
# (1) Importing dependency

import keras

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout, Flatten,\

Conv2D, MaxPooling2D

from keras.layers.normalization import BatchNormalization

import numpy as np

np.random.seed(1000)



# (2) Get Data

import tflearn.datasets.oxflower17 as oxflower17

x, y = oxflower17.load_data(one_hot=True)



# (3) Create a sequential model

model = Sequential()
```

```
# 1st Convolutional Layer

model.add(Conv2D(filters=96, input_shape=(224,224,3),
kernel_size=(11,11),\

strides=(4,4), padding='valid'))

model.add(Activation('relu'))

# Pooling

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2),
padding='valid'))

# Batch Normalisation before passing it to the next layer

model.add(BatchNormalization())



# 2nd Convolutional Layer

model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1),
padding='valid'))

model.add(Activation('relu'))

# Pooling

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2),
padding='valid'))

# Batch Normalisation
```

```
model.add(BatchNormalization())



# 3rd Convolutional Layer

model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
padding='valid'))

model.add(Activation('relu'))

# Batch Normalisation

model.add(BatchNormalization())



# 4th Convolutional Layer

model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
padding='valid'))

model.add(Activation('relu'))

# Batch Normalisation

model.add(BatchNormalization())



# 5th Convolutional Layer

model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
```

```
padding='valid'))

model.add(Activation('relu'))

# Pooling

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2),
padding='valid'))

# Batch Normalisation

model.add(BatchNormalization())



# Passing it to a dense layer

model.add(Flatten())

# 1st Dense Layer

model.add(Dense(4096, input_shape=(224*224*3,)))

model.add(Activation('relu'))

# Add Dropout to prevent overfitting

model.add(Dropout(0.4))

# Batch Normalisation

model.add(BatchNormalization())
```

```
# 2nd Dense Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Batch Normalisation
model.add(BatchNormalization())

# 3rd Dense Layer
model.add(Dense(1000))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Batch Normalisation
model.add(BatchNormalization())

# Output Layer
```

```
model.add(Dense(17))

model.add(Activation('softmax'))


model.summary()


# (4) Compile

model.compile(loss='categorical_crossentropy', optimizer='adam',\

metrics=['accuracy'])


# (5) Train

model.fit(x, y, batch_size=64, epochs=1, verbose=1, \

validation_split=0.2, shuffle=True)
```

LAMPIRAN B

Pseudocode pelatihan model DeepSpace

```
import tensorflow as tf
import pandas as pd

from tensorflow.keras import layers
from tensorflow.keras import preprocessing
from tensorflow.keras import models

from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping
from tensorflow.keras.layers import Conv2D, Dense, Flatten,
MaxPooling2D, Activation, Dropout, GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt
import json
from tensorflow.keras.models import model_from_json
# dimensions of our images.
img_width, img_height = 227, 227

train_data_dir = 'data/train'
validation_data_dir = 'data/validation'
epochs = 30
batch_size = 15

input_shape = (img_width, img_height, 3)
```

```
model = Sequential()

# 1st MlpConv
model.add(Conv2D(filters=96, input_shape=input_shape,
kernel_size=(11,11), strides=(4,4), padding='valid', activation='relu'))
model.add(Conv2D(96, (1, 1),activation='relu'))
model.add(Conv2D(96, (1, 1),activation='relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2),
padding='valid'))

# 2nd MlpConv
model.add(Conv2D(filters=256, input_shape=input_shape,
kernel_size=(1,1), strides=(1,1), padding='valid',activation='relu'))
model.add(Conv2D(256, (1, 1),activation='relu'))
model.add(Conv2D(256, (1, 1),activation='relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2),
padding='valid'))

#3rd MlpConv
model.add(Conv2D(filters=384, input_shape=input_shape,
kernel_size=(1,1), strides=(1,1), padding='valid',activation='relu'))
model.add(Conv2D(384, (1, 1),activation='relu'))
model.add(Conv2D(384, (1, 1),activation='relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(1,1),
padding='valid'))


#Convolutional Layer
model.add(Conv2D(filters=512, input_shape=input_shape,
kernel_size=(1,1), strides=(1,1), padding='valid',activation='relu'))
model.add(Conv2D(filters=512, input_shape=input_shape,
```

```
kernel_size=(1,1), strides=(1,1), padding='valid',activation='relu'))
#Global Average Pooling
model.add(GlobalAveragePooling2D())

model.add(Flatten())

# 1st Fully Connected Layer
model.add(Dense(512, input_shape=(512,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))

# Output Layer
model.add(Dense(5))
model.add(Activation('softmax'))


model.summary()

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
train_datagen = ImageDataGenerator(
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=False,
        fill_mode='nearest')

test_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
```

```
        class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical')

checkpoint = ModelCheckpoint("deepspace-best.h5",
monitor='val_accuracy', verbose=1, save_best_only=True,
save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='val_accuracy', min_delta=0,
patience=15, verbose=1, mode='auto')

hist = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)

plt.plot(hist.history["accuracy"])
plt.plot(hist.history['val_accuracy'])
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation
Loss"])
plt.show()

model_json = model.to_json()
with open("deepspace.json", "w") as json_file:
```

```
        json.dump(model_json, json_file)
model.save_weights('deepspace.h5')
```

LAMPIRAN C

Pseudocode pelatihan model-A

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.callbacks import ModelCheckpoint, EarlyStopping
import matplotlib.pyplot as plt
import json

# dimensions of our images.
img_width, img_height = 227, 227

train_data_dir = <directory>
validation_data_dir = <directory>
epochs = 30
batch_size = 40

input_shape = (img_width, img_height, 3)

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(Conv2D(32, (1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(5))
model.add(Activation('softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator()

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
```

```
        batch_size=batch_size,
        class_mode='categorical')

checkpoint = ModelCheckpoint("modelB-best.h5",
monitor='val_accuracy', verbose=1, save_best_only=True,
save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='val_accuracy', min_delta=0,
patience=20, verbose=1, mode='auto')

hist = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size,
        callbacks= [checkpoint,early])

plt.plot(hist.history["accuracy"])
plt.plot(hist.history['val_accuracy'])
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation
Loss"])
plt.show()

model_json = model.to_json()
with open("modelA.json", "w") as json_file:
        json.dump(model_json, json_file)
model.save_weights('modelA.h5')
```

LAMPIRAN D

Pseudocode pelatihan menggunakan 2 jenis data

```
from keras.preprocessing.image import load_img, img_to_array,
ImageDataGenerator
from keras.models import model_from_json, Model, Sequential
from keras.layers import concatenate, Input
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.callbacks import ModelCheckpoint, EarlyStopping
import matplotlib.pyplot as plt
import numpy as np
import json


##define DATAS##
img_width, img_height = 227, 227

X_train = <directory>
X_validation =    <directory>
Y_train =    <directory>
Y_validation =    <directory>
epochs = 30
batch_size = 10


###MODEL###
#RGB
with open('modelB.json','r') as f:
        model_json = json.load(f)
x = model_from_json(model_json)
x.load_weights('modelB-best.h5')
x.layers.pop()
x1 = Model(inputs=x.inputs, outputs=x.layers[-7].output)
```

```
for layer in x1.layers:
        layer.trainable = False
#x1.summary()

#IR
y = model_from_json(model_json)
y.load_weights('modelB-best.h5')
y.layers.pop()
y1 = Model(inputs=y.inputs, outputs=y.layers[-7].output)
for layer in y1.layers:
        layer.name = layer.name + str("_2")
        layer.trainable = False
#y1.summary()

combined = concatenate([x1.output, y1.output])
flat1 = Flatten()(combined)
class1 = Dense(64, activation='relu')(flat1)
dropout1= Dropout(0.5)(class1)
output = Dense(5, activation='softmax')(dropout1)
model = Model(inputs=[x1.input, y1.input], outputs=output)
model.summary()
model.compile(loss='categorical_crossentropy',
                        optimizer='rmsprop',
                        metrics=['accuracy'])

###DATA GENERATOR###
gen = ImageDataGenerator()

##TRAINING GENERATOR
# Here is the function that merges our two generators
# We use the exact same generator with the same random seed for
both the y and angle arrays
def train_gen_flow_for_two_inputs(X1, X2):
        genX1 = gen.flow_from_directory(X1,
```

```
                                                    target_size=(img_width,
img_height),
                                                    batch_size=batch_size,

class_mode='categorical')
        genX2 = gen.flow_from_directory(X2,
                                                    target_size=(img_width,
img_height),
                                                    batch_size=batch_size,

class_mode='categorical')
        while True:
                    X1i = genX1.next()
                    X2i = genX2.next()
                    #Assert arrays are equal - this was for peace of mind,
but slows down training
                  # np.testing.assert_array_equal(X1i[0],X2i[0])
                    yield [X1i[0], X2i[0]],X2i[1]

##VALIDATION GENERATOR
# Here is the function that merges our two generators
# We use the exact same generator with the same random seed for
both the y and angle arrays
def validation_gen_flow_for_two_inputs(Y1, Y2):
        genY1 = gen.flow_from_directory(Y1,
                                                    target_size=(img_width,
img_height),
                                                    batch_size=batch_size,

class_mode='categorical')
        genY2 = gen.flow_from_directory(Y2,
                                                    target_size=(img_width,
img_height),
                                                    batch_size=batch_size,
```

```
class_mode='categorical')
        while True:
                        Y1i = genY1.next()
                        Y2i = genY2.next()

train_generator = train_gen_flow_for_two_inputs(X_train, Y_train)
val_generator = validation_gen_flow_for_two_inputs(X_validation,
Y_validation)

##FIT TO MODEL##
checkpoint = ModelCheckpoint("modelB-mixed-best.h5",
monitor='val_accuracy', verbose=1, save_best_only=True,
save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='val_accuracy', min_delta=0,
patience=20, verbose=1, mode='auto')

hist = model.fit_generator(
        train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs,
        validation_data=val_generator,
        validation_steps=nb_validation_samples // batch_size,
        callbacks= [checkpoint,early])

plt.plot(hist.history["accuracy"])
plt.plot(hist.history['val_accuracy'])
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation
Loss"])
```

```
plt.show()

model_json = model.to_json()
with open("filename.json", "w") as json_file:
        json.dump(model_json, json_file)
model.save_weights(filename..h5')
```

LAMPIRAN E

Pseudocode Testing Jaringan

```
from keras.preprocessing import image
import numpy as np
import json
from keras.models import model_from_json
import random

matrix = []
label = (['C215', 'C216','S1','Pasca','Dalam'])

#TEST DIRECTORY
test_dir = <directory>
weight_file = <directory>
json_file = <directory>


#WRITE TO TEXT
file1 = open("modelA-NIR-roomC.txt","a")

for i in range(730):
	result = []
	#number = str(i+1)
	number = str(random.randint(1,8811))
	img = image.load_img(test_dir + "roomD-"+ number +"-
a.jpg",target_size=(227,227))
	img = np.asarray(img)
	img = np.expand_dims(img, axis=0)

	with open(json_file,'r') as f:
		model_json = json.load(f)
```

```
model = model_from_json(model_json)
model.load_weights(weight_file)
prediction = model.predict_proba(img)
output = list(prediction[0])
#print(output)

#top-1
indeks = output.index(max(output))
indextop1 = str(output.index(max(output)))
#pred1 = label[indextop1]
#print(indexmaks)
#print(pred1)

#top-2
new_list = output
new_list[indeks]= 0
indextop2 =str(output.index(max(new_list)))
#pred2 = label[output.index(max(new_list))]
#print(max(new_list))
result = [indextop1, indextop2]
#result.append(indextop1)
#result.append(indextop2)
print (result)
#write to text

file1.write(str(result)+ "\n")

#matrix.append(result)
#matrix = np.append(matrix, result, axis=1)
```

LAMPIRAN F

Pseudocode Sistem Pengenalan

```
import pyrealsense2 as rs
import numpy as np
import json
import cv2
from keras.models import model_from_json
import h5py

#Category Labels
label = (['C215', 'C216','TU S1','TU Pasca','TU Dalam'])

pipeline = rs.pipeline()
config = rs.config()

config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
config.enable_stream(rs.stream.infrared, 1, 640,480, rs.format.y8, 30)

cfg = pipeline.start(config)
dev = cfg.get_device()

try:
	while True:
		frames = pipeline.wait_for_frames()
		color_frame= frames.get_color_frame()
		infrared_frame = frames.get_infrared_frame()

		color_image = np.asanyarray(color_frame.get_data())
		infrared = np.asanyarray(infrared_frame.get_data())
		infrared_image = cv2.cvtColor(infrared,
cv2.COLOR_GRAY2BGR)
		images = np.hstack((color_image,	infrared_image))
```

```
            #resize the image
            img = cv2.resize(color_image, (227,227))
            img = np.asarray(img)
            img = np.expand_dims(img, axis=0)
            with open('C:/Users/Aurell/interference-
trial/modelB.json','r') as f:
                model_json = json.load(f)

            model = model_from_json(model_json)
            model.load_weights('C:/Users/Aurell/interference-
trial/modelB-best.h5')
            output = model.predict(img)
            print (output)
            ind = np.unravel_index(output.argmax(), output.shape)

            cv2.putText(images, label[ind[1]], (60,80),
cv2.FONT_HERSHEY_SIMPLEX, 2,
(255,255,255),2,cv2.LINE_AA)
            #cv2.putText(color_image, label[ind[1]], (60,80),
cv2.FONT_HERSHEY_SIMPLEX, 2,
(255,255,255),2,cv2.LINE_AA)
            cv2.namedWindow('RealSense',
cv2.WINDOW_AUTOSIZE)
            #cv2.namedWindow('RGB', cv2.WINDOW_AUTOSIZE)
            #cv2.namedWindow('Infrared',
cv2.WINDOW_AUTOSIZE)
            cv2.imshow('RealSense',images)
            #cv2.imshow('RGB',color_image)
            #cv2.imshow('Infrared',infrared_image)

            if cv2.waitKey(1) == ord("q"):
                break
finally:
    pipeline.stop()
```

BIODATA PENULIS

Penulis, Aurell Quinta Miranda, lahir di Jakarta pada 4 Januari 1999. Penulis merupakan lulusan dari SMPK 3 Penabur Jakarta tahun kelulusan 2013 dan SMA Negeri 77 Jakarta dengan tahun kelulusan 2016. Pada tahun 2016 penulis melanjutkan pendidikan di Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember dengan bidang studi Elektronika. Selama masa kuliah penulis aktif sebagai pengurus kaderisasi Fakultas Teknologi Elektro, asisten laboratorium B202, dan beberapa kepanitiaan. Penulis bercita-cita menggunakan ilmu yang didapatkan pada masa kuliah untuk membantu masyarakat dan menghasilkan inovasi demi meningkatkan kualitas edukasi di Indonesia.