



**PROYEK AKHIR**

**SISTEM PEMANTAU LINGKUNGAN UNTUK  
MENGANTISIPASI PENYEBARAN PENYAKIT  
PENYAKIT MENULAR**

*ENVIRONMENTAL MONITORING SYSTEM TO ANTICIPATE  
THE SPREAD OF DISEASES*

**Oleh:**

**Hanif Izzudin Rahman**  
**NRP. 1110161043**

**Dosen Pembimbing :**

**Ir. Wahjoe Tjatur Sesulihatien, MT., Ph.D**  
**NIP. 196506221991032003**

**Ir. Dadet Pramadihanto, M. Eng., Ph.D**  
**NIP. 196202111988111001**

**PROGRAM STUDI D4 TEKNIK ELEKTRONIKA  
DEPARTEMEN TEKNIK ELEKTRO  
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA  
SURABAYA  
2020**

**PROYEK AKHIR**



## **PROYEK AKHIR**

### **SISTEM PEMANTAU LINGKUNGAN UNTUK MENGANTISIPASI PENYEBARAN PENYAKIT MENULAR**

*ENVIRONMENTAL MONITORING SYSTEM TO ANTICIPATE THE  
SPREAD OF DISEASES*

**Oleh:**

**Hanif Izzudin Rahman**

**NRP. 1110 161 043**

**Dosen Pembimbing :**

**Ir. Wahjoe Tjatur Sesulihatien, MT., Ph.D**

**NIP. 196506221991032003**

**Ir. Dadet Pramadihanto, M. Eng., Ph.D**

**NIP. 196202111988111001**

**PROGRAM STUDI D4 TEKNIK ELEKTRONIKA  
DEPARTEMEN TEKNIK ELEKTRO  
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA  
SURABAYA  
2020**

# SISTEM PEMANTAU LINGKUNGAN UNTUK MENGANTISIPASI PENYEBARAN PENYAKIT MENULAR

Oleh :

Hanif Izzudin Rahman  
NRP. 1110161043

Proyek Akhir ini Diajukan sebagai Salah Satu Syarat untuk  
Memperoleh Gelar Sarjana Terapan Teknik (S.Tr.T.)  
di  
Program Studi D4 Teknik Elektronika  
Departemen Teknik Elektro  
Politeknik Elektronika Negeri Surabaya

Disetujui dan disahkan pada tanggal 18 Agustus 2020 oleh:

Dosen Pembimbing:

1. Ir. Wahjoe Tjatur Sesulihatien, MT., Ph.D.  
NIP. 196506221991032003
2. Ir. Dadet Pramadihanto, M.Eng, Ph.D  
NIP. 196202111988111001

Dosen Penguji:

1. Dr.. Arif Irwansyah, S.T., M.Eng  
NIP. 197703182001121002
2. Taufiqurrahman, S.ST, MT  
NIP. 198309202008121001
3. Ir. Rika Rokhana, MT  
NIP. 196909051998022001



Mengetahui,  
Ketua Program Studi D4 Teknik Elektronika

Dr..Arif Irwansyah,S.T., M.Eng  
NIP. 197703182001121002

## **PERNYATAAN ORISINILITAS**

Saya selaku penulis menyatakan bahwa Proyek Akhir ini adalah benar-benar hasil karya saya sendiri, dan semua sumber/referensi baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Surabaya, 3 Agustus 2020  
Penulis yang menyatakan,

Hanif Izzudin Rahman  
NRP. 1110161043

## ABSTRAK

Tugas akhir ini berkaitan dengan sistem yang dipakai untuk membantu penanggulangan penyebaran penyakit menular. Cara penanggulangan difokuskan pada partisipasi masyarakat. Dalam tugas akhir ini ada dua hal yang dikerjakan yaitu membuat sistem kamera untuk memantau lingkungan dan membuat perangkat lunak pemantau lingkungan. Sistem kamera yang dibuat adalah sebuah kamera spectral, yaitu kamera RGB dan kamera NoIR dengan 3 band filter (770nm-790nm, 855nm-890nm dan 928nm-955nm). Kamera ini dioperasikan menggunakan raspberry pi 3 untuk mengambil suatu gambar sekaligus untuk komunikasi antar module telemetry. Kamera ini akan diterbangkan di daerah kerumunan menggunakan drone dan gambarnya akan dianalisis di bagian perangkat lunak. Sistem perangkat lunak pemantau lingkungan terdiri dari software GUI komunikasi telemetry dan program pemantau lingkungan. Software GUI bertujuan untuk mengirim suatu perintah untuk mengoperasikan sistem kamera sekaligus menerima informasi darinya. Pada bagian program pemantau lingkungan, menggunakan CNN dengan arsitektur CSRNet, yaitu tehnik yang akan digunakan untuk mengenali berapa jumlah kerumunan yang ada pada sebuah gambar. Dataset yang telah diambil menggunakan kamera spectral menghasilkan gambar yang kurang baik, dalam artian tidak mengandung banyak kerumunan. Alhasil, disini menggunakan Dataset Shanghai yang mempunyai 1198 gambar dengan kombinasi kerumunan 330.165 orang. Dari hasil testing yang didapat, didapatkan error besar hingga 170%. Ini dikarenakan gambar yang di testing bukanlah gambar yang berkerumun. Pada saat proses training model menggunakan CSRNet, tidak diinputkan juga sebuah dataset kerumunan yang diambil menggunakan drone.

**Kata Kunci:** Penyakit Menular; Kamera; Deteksi Kerumunan; CSRNet

## **ABSTRACT**

*This final project deals with the system used to help control the spread of infectious diseases. Prevention measures are focused on community participation. In this final project, there are two things that are done, namely creating a camera system to monitor the environment and making environmental monitoring software. The camera system created is a spectral camera, namely an RGB camera and a NoIR camera with 3 band filters (770nm-790nm, 855nm-890nm and 928nm-955nm). This camera is operated using Raspberry Pi 3 to take an image as well as to help communication between telemetry modules. This camera will be flown in a crowd area using a drone to pick up a crowd object which will be analyzed in the software section. The environmental monitoring software system consists of telemetry communication GUI software and an environmental monitoring program. The GUI software aims to send a command to operate the camera system as well as receive information from it. In the environmental monitoring program, using CNN with the CSRNet architecture, a technique that will be used to identify how many crowds there are in an image. The dataset that has been captured using a spectral camera results in a poor image, in the sense that the data captured does not contain a lot of crowds. As a result, here using the Shanghai Dataset which has 1198 images with a combined crowd of 330,165 people. From the test results obtained, a large error of up to 170% was obtained. This is because the image being tested is not a clustered image. During the model training process using CSRNet, a crowd dataset that was taken using drones was not entered.*

**Keyword:** *Infectious diseases; Cameras; Crowd Detection; CSRNet*

## KATA PENGANTAR



Alhamdulillah, segala puji syukur bagi Allah SWT karena berkat rahmat dan hidayah-Nya, penulis dapat menyelesaikan proyek akhir yang berjudul :

### **“SISTEM PEMANTAU LINGKUNGAN UNTUK MENGANTISIPASI PENYEBARAN PENYAKIT MENULAR”**

Pembuatan dan penyusunan proyek akhir ini diajukan sebagai salah satu syarat untuk menyelesaikan studi Diploma-4 (D4) dan memperoleh gelar Sarjana Sains Terapan (S. ST) di jurusan Teknik Elektronika Politeknik Elektronika Negeri Surabaya.

Penulis menyadari bahwa buku proyek akhir ini masih memiliki banyak kekurangan. Untuk itu penulis memohon maaf sebesar-besarnya atas segala kekurangan dalam penyusunannya. Penulis juga mengharapkan saran dan kritik dari semua pihak demi kesempurnaan buku ini.

Akhirnya, penulis berharap semoga buku proyek akhir ini memiliki manfaat yang besar khususnya bagi penulis dan pembaca pada umumnya sebagai sarana ilmu, wawasan, dan pengetahuan

Surabaya, 3 Agustus 2020

**Penulis**

## UCAPAN TERIMA KASIH

Alhamdulillah, atas segala limpahan rahmat, taufik, serta hidayah yang diberikan oleh Allah SWT sehingga proyek akhir ini dapat terselesaikan sesuai dengan jadwal yang telah ditentukan.

Di samping itu, penulis ingin mengucapkan rasa terima kasih kepada semua pihak yang telah memberikan bantuan, bimbingan, dan dorongan serta fasilitas sarana dan prasarana, baik material maupun spiritual sehingga penulis dapat menyusun buku laporan proyek akhir ini tepat pada waktunya, di antaranya adalah:

1. Ibu, Ayah, dan keluarga tercinta. Terima kasih atas segala pengorbanan, doa, dan semangat yang telah diberikan. Semoga segera tercapai segala angan dan cita-cita.
2. Bapak **Dr. Zainal Arief, S.T., M.T.** selaku direktur PENS.
3. Bapak **I Gede Puja Astawa, S.T, M.T.** selaku Kepala Departemen Teknik Elektro PENS.
4. Bapak **Dr. Arif Irwansyah, S.T. M.T.**, selaku Ketua Program Studi Diploma IV Teknik Elektronika PENS.
5. Ibu **Ir. Wahjoe Tjatur Sesulihatien, MT., Ph.D,Ir. Dadet Pramadihanto, M. Eng., Ph.D.** selaku dosen pembimbing yang telah banyak memberi pengarahan serta bimbingan baik teknis maupun non-teknis kepada penulis dalam menyelesaikan proyek akhir ini
6. Bapak **Dosen** selaku dosen penguji, yang telah memberi banyak saran dan masukan dalam menyelesaikan proyek akhir ini.
7. **Teman Teman D4 Elka B 2016** yang selama 4 tahun ini telah menemani dalam suka maupun duka.
8. Dan juga semua pihak yang tidak bisa penulis sebutkan satu persatu yang telah membantu dalam penyelesaian proyek akhir.

Semoga Allah S.W.T selalu memberikan perlindungan, rahmat dan nikmat-Nya bagi kita semua. Amin!



## DAFTAR ISI

<b>HALAMAN JUDUL .....</b>	<b>i</b>
<b>KATA PENGANTAR.....</b>	<b>vi</b>
<b>UCAPAN TERIMA KASIH.....</b>	<b>viii</b>
<b>DAFTAR ISI.....</b>	<b>viii</b>
<b>DAFTAR GAMBAR.....</b>	<b>ix</b>
<b>DAFTAR TABEL.....</b>	<b>xii</b>
<b>BAB I PENDAHULUAN .....</b>	<b>1</b>
1.1 LATAR BELAKANG .....	2
1.2 TUJUAN .....	2
1.3 PERUMUSAN MASALAH .....	2
1.4 MANFAAT PROGRAM .....	2
1.5 BATASAN MASALAH.....	3
1.6 SISTEMATIKA PEMBAHASAN .....	3
<b>BAB II TEORI PENUNJANG .....</b>	<b>5</b>
2.1 <i>Multispectral Imaging</i> .....	4
2.2 Radio Telemetry Module .....	5
2.3 CNN .....	6
<b>BAB III DISAIN SISTEM.....</b>	<b>11</b>
3.1 Perancangan Dan Pembuatan Sistem Kamera.....	11
3.2 Perancangan Hardware.....	13
3.3 Perancangan Software .....	16
<b>BAB IV PENGUJIAN DAN ANALISA .....</b>	<b>26</b>
4.1 Pengujian Range Komunikasi Telemetry.....	26
4.2 Pengambilan Gambar Kamera Spectral.....	30
4.3 Pengujian Model CSRNet .....	33

4.4	Pengujian Model (Ketinggian 5 meter) .....	49
<b>BAB V PENUTUP</b> .....		52
5.1	Kesimpulan .....	52
5.2	Saran .....	52
<b>DAFTAR PUSTAKA</b> .....		53
<b>LAMPIRAN A</b> .....		55
<b>Lampiran B (Hasil Gambar Camera Spectral)</b> .....		84
<b>Lampiran C (Hasil Gambar Peta Kerapatan)</b> .....		86

## DAFTAR GAMBAR

Gambar 2.1	3x3 kernel convolusi dengan dilation rate yang berbeda ..7
Gambar 2.2	Perbedaan dilated convolution dan max-pooling, convolution dan upsampling ..... 8
Gambar 3.1	Diagram Umum Sistem ..... 11
Gambar 3.2	Schematic Raspberry Pi dengan Camera Adaopter ..... 14
Gambar 3.3	Tampak depan Kamera Spectral ..... 14
Gambar 3.4	Tampak samping Kamera Spectral ..... 15
Gambar 3.5	Design alas camera spectral 30 derajat .....15
Gambar 3.6	Gambar kamera spectral beserta dronanya ..... 15
Gambar 3.7	Setting Konfigurasi Telemetry Module Ardu Pilot ..... 16
Gambar 3.8	GUI pada PC mengirimkan perintah ambil gambar ..... 17
Gambar 3.9	Hasil Gambar dari 1 Folder ..... 19
Gambar 3.10	Hasil RGB dan Noir dengan 3 Band dalam 1 folder ..... 20
Gambar 3.11	91 Gambar Kerumunan RGB ..... 20
Gambar 3.12	ROI Kerumunan Training Image Labeler ..... 21
Gambar 3.13	Informasi banyaknya kerumunan beserta lokasinya ..... 21
Gambar 3.14	Dataset Shanghai part A (kiri) dan part B (kanan) ..... 22
Gambar 3.15	Gambar kerumunan RGB dan Peta Kerapatan ..... 23
Gambar 3.16	File Model A dan Model B hasil modelling ..... 24
Gambar 3.17	Gambar RGB dan Peta Kepadatannya (61 orang) ..... 25
Gambar 3.18	Hasil prediksi jumlah kerumunan dengan Model A ..... 25

## DAFTAR TABEL

Tabel 3.1 GPIO pada Raspberry untuk switch Camera Raspberry .....	14
Tabel 3.2 Alamat I2C untuk menentukan Camera mana yang aktif ...	14
Tabel 3.3 Libray yang digunakan saat modelling .....	24
Tabel 3.4 Perbandingan Komunikasi Telemetry antara Radius dan Ketinggi .....	30

# **BAB I**

## **PENDAHULUAN**

### **1.1 LATAR BELAKANG**

Penyakit menular dapat ditularkan secara langsung maupun tidak langsung. Penularan secara langsung terjadi ketika kuman pada orang yang sakit berpindah melalui kontak fisik, misalnya lewat sentuhan, melalui udara saat bersin dan batuk, atau melalui kontak dengan cairan tubuh seperti urine dan darah. Penyakit menular juga dapat berpindah secara tidak langsung. Misalnya saat menyentuh kenop pintu, keran air, atau tiang besi pegangan di kereta yang terkontaminasi.

Untuk memantau terjadinya kontak langsung antar manusia, dibuatlah sebuah alat yang dapat memprediksi jumlah orang yang tertular berbasis citra. Alat ini menggunakan kamera *spectral* yang akan diterbangkan di sebuah kerumunan dan data yang didapat akan dianalisa menggunakan software.

Kamera *spectral* yang digunakan adalah kamera RGB, untuk kamera NoIR dengan 3 filternya tidak digunakan. Ini dikarenakan dampak dari masa awal merebaknya virus covid-19 di Indonesia, terutama di Surabaya. Kamera NoIR ini awalnya digunakan untuk menemukan fitur-fitur pada sebuah lingkungan yang menyebabkan penyakit demam berdarah, dengan adanya covid-19, alhasil tidak didapatkan sebuah data. Oleh karena itu, dengan *hardware* yang sudah jadi, maka judul proyek akhir ini diganti dengan “Sistem Pemantau Lingkungan Untuk Mengantisipasi Penyebaran Penyakit Menular”

### **1.2 TUJUAN**

Tujuan dari proyek akhir ini adalah :

1. Membuat kamera yang dapat dipakai untuk memantau lingkungan, dalam hal ini manusia
2. Membuat perangkat lunak pendeteksi manusia berbasis citra

### **1.3 PERUMUSAN MASALAH**

Adapun permasalahan yang dibahas dalam proyek akhir ini adalah sebagai berikut:

1. Membuat kamera dengan feature yang representative untuk mendeteksi kerumunan
2. Membuat sistem embedded kamera pada drone

3. Membuat sistem telemetri untuk pengontrolan pengambilan data pada drone
4. Membuat perangkat lunak pendeteksi lingkungan yaitu kerumunan manusia

#### **1.4 MANFAAT PROGRAM**

Manfaat yang diharapkan dari hasil pengerjaan proyek akhir ini adalah dapat memantau lingkungan, yaitu manusia menggunakan perangkat lunak berbasis citra.

#### **1.5 BATASAN MASALAH**

Adapun batasan – batasan masalah yang dibuat agar dalam pembuatan proyek akhir ini dapat berjalan dengan baik adalah sebagai berikut :

1. Sistem kamera yang digunakan untuk saat ini adalah kamera RGB saja
2. Sistem telemetri terbatas pada ketinggian 15 meter dan radius 30 meter
3. Sistem perangkat lunak dibatasi dengan error yang masih besar, yaitu 170%

#### **1.6 SISTEMATIKA PEMBAHASAN**

Sistematika pembahasan dalam penyusunan buku proyek akhir ini adalah sebagai berikut :

##### **A. BAB I PENDAHULUAN**

Bab ini membahas pendahuluan yang terdiri dari latar belakang, tujuan, batasan masalah, sistematika pembahasan Proyek Akhir dan tinjauan pustaka.

##### **B. BAB II TEORI PENUNJANG**

Bab ini membahas teori-teori yang menunjang dan berkaitan dengan penyelesaian Proyek Akhir, antara lain teori Penyakit Menular, Kamera, Deteksi Kerumunan dan *CSRNet*

##### **C. BAB III PERANCANGAN & PEMBUATAN**

Bab ini membahas tahap perencanaan dan proses pembuatan perangkat keras Proyek Akhir.

##### **D. BAB IV ANALISA HASIL PENGUJIAN**

Bab ini membahas secara keseluruhan dari sistem dan dilakukan pengujian serta analisa pada setiap percobaan perangkat keras. Mengintegrasikan seluruh sistem dan pengujian, kemudian berdasarkan data hasil pengujian dan dilakukan analisa terhadap keseluruhan sistem.

#### **E. BAB V KESIMPULAN DAN SARAN**

Bab ini membahas kesimpulan dari pembahasan, perencanaan, pengujian dan analisa berdasarkan data hasil pengujian sistem. Untuk meningkatkan hasil akhir yang lebih baik diberikan saran-saran terhadap hasil pembuatan Proyek Akhir

#### **F. DAFTAR PUSTAKA**

Pada bagian ini berisi tentang referensi-referensi yang telah dipakai oleh penulis sebagai acuan dan penunjang serta parameter yang mendukung penyelesaian proyek akhir ini baik secara praktis maupun teoritis.

#### **G. LAMPIRAN**

Pada bagian ini berisi tentang data penunjang hasil pengujian sistem.

## **BAB II**

### **TEORI PENUNJANG**

#### **2.1 *Multispectral Imaging***

Mata manusia hanya dapat melihat dalam *spectrum elektromagnetik* yang terbatas dan dapat membedakan objek berdasarkan respons spectral yang berbeda dalam batas spectral sempit itu . Namun, sensor citra multispectral telah mengalami perkembangan yang signifikan sehingga dapat memperoleh gambar dalam spectrum inframerah (*infrared*) dan spectrum yang terlihat (*visible*) dari sebuah susunan spectrum elektromagnetik. Hal ini memungkinkan kita mengidentifikasi benda berdasarkan *spectral* unik mereka dalam rentang spectrum yang luas. Citra *multispectral* dapat mengetahui ciri khas atau keunikan sendiri pada tiap benda berdasarkan indikasi spectrum yang ada. Spectrum dari sebuah piksel dalam gambar *multispectral* memberikan informasi tentang komponen penyusun dan jenis permukaan materialnya

##### **2.1.1 Indikasi Spektral Pada Citra Multispektral**

Benda yang berada di permukaan bumi menyerap, mengirimkan dan memantulkan radiasi elektromagnetik dari matahari dengan cara yang unik. Sensor *multispectral* memungkinkan kita untuk mengukur semua jenis radiasi elektromagnetik dalam rentang tertentu pada sebuah benda di permukaan bumi. Sehingga memungkinkan kita untuk mengamati perbedaan fitur atau ciri-ciri dari benda serta perubahannya. Reflektansi adalah banyaknya sebuah energy elektromagnetik yang memantul kembali dari sebuah permukaan benda ke sebuah sensor citra. Atau secara matematis, reflektansi adalah rasio *energy* yang dipantulkan terhadap kejadian *energy* sebagai fungsi panjang gelombang. Ukuran reflektansi adalah 0-100%. Apabila reflektansi bernilai 100% maka cahaya yang menyentuh benda dengan panjang gelombang tertentu akan dipantulkan seluruhnya menuju sensor citra. Sedangkan apabila reflektansi bernilai 0% maka cahaya dengan panjang gelombang tertentu akan diserap seluruhnya oleh benda. Dalam rentang *spectrum* elektromagnetik tertentu, nilai reflektansi dari berbagai benda yang ada pada permukaan bumi



seperti tanah, hutan, air dan mineral bisa diplot dan dibandingkan. Gambar 2.1 menunjukkan model dari indikasi citra spectral dengan berbagai benda berbeda yang ada di permukaan bumi.

Citra yang diambil dengan jarak yang jauh dapat diklasifikasikan menggunakan indikasi *spectral* ini, karena setiap benda yang ada memiliki indikasi *spectral* yang unik. Semakin banyak resolusi spectral dari sensor citra maka semakin banyak informasi klasifikasi yang bisa didapat dari indikasi *spectral*.

### **2.1.2 Image Processing**

Image processing adalah suatu bentuk pengolahan atau pemrosesan sinyal dengan input berupa gambar (image) dan ditransformasikan menjadi gambar lain sebagai keluarannya dengan teknik tertentu. Image processing dilakukan untuk memperbaiki kesalahan data sinyal gambar yang terjadi akibat transmisi dan selama akuisisi sinyal, serta untuk meningkatkan kualitas penampakan gambar agar lebih mudah diinterpretasi oleh sistem penglihatan manusia baik dengan melakukan manipulasi dan juga penganalisisan terhadap gambar.

## **2.2 Radio Telemetry Module**

Sistem telemetry Radio 3DR dirancang sebagai perangkat radio pengganti Xbee *open source*, menawarkan harga yang lebih rendah, jarak yang lebih jauh, dan kinerja yang unggul dari radio Xbee. Ini tersedia dalam 433Mhz dan dalam konfigurasi berikut: papan serial (untuk udara) dan USB (untuk *ground*).

Modul Radio Udara digunakan untuk UAV atau peralatan atau peralatan kerja udara. Modul Radio Tanah digunakan untuk menghubungkan ke komputer atau menampilkan gambar pada peralatan lain di dalam tanah. Ketika dua modul diikat dengan sinyal nirkabel, kita dapat menggunakan stasiun darat APM (*Mission Planner*) untuk menyesuaikan dan meningkatkan parameter.

### **Features**

- Pita frekuensi 915 MHz. Sensitivitas penerima hingga -117 dBm

- Kecil dan ringan, dapat ditempatkan di sleeving dan juga diikat ke lengan mekanik.
- Antena dBi, ruang lingkup pengiriman dan penerimaan hingga 1000 meter.
- Kirim daya hingga 20dBm (100mW). Tautan serial transparan. Konektor antena: konektor RP-SMA.
- Pembingkatan protokol MAVLink dan pelaporan status.

## 2.3 CNN

Diperlukan sebuah CNN yang lebih mendalam agar dapat menangkap sebuah fitur *high-level* dengan bidang reseptif yang lebih besar dan menghasilkan peta kepadatan berkualitas tinggi tanpa memperluas kompleksitas jaringan secara besar-besaran. Digunakan lah sebuah arsitektur CSRNet.

### 2.3.1 Arsitektur CSRNet

Pada Arsitekturnya, menggunakan VGG-16 sebagai *front-end* karena kemampuannya untuk *transfer learning* dengan kuat dan arsitekturnya yang fleksibel agar memudahkan merangkai bagian *back-end* untuk pembuatan peta kerapatannya. Ukuran output pada bagian *front-end* ini adalah 1/8 dari ukuran aslinya. Pada bagian *back-end* digunakan *dilated convolutional network* untuk mengekstraksi informasi penting lebih dalam serta menjaga resolusi *output*.

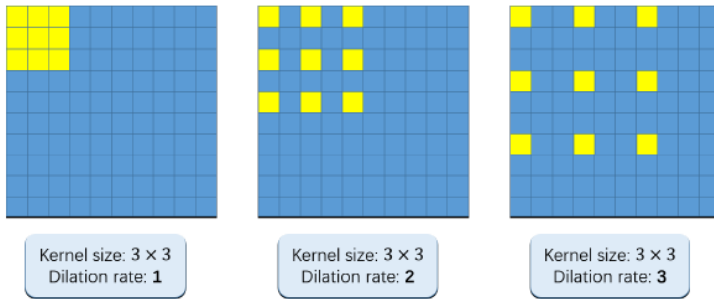
### 2.3.2 Dilated Convolution

Salah satu bagian kritis dalam CSRNet adalah bagian ini. 2-D *dilated convolution* didefinisikan sebagai berikut

$$y(m, n) = \sum_{i=1}^M \sum_{j=1}^N x(m + r \times i, n + r \times j) w(i, j)$$

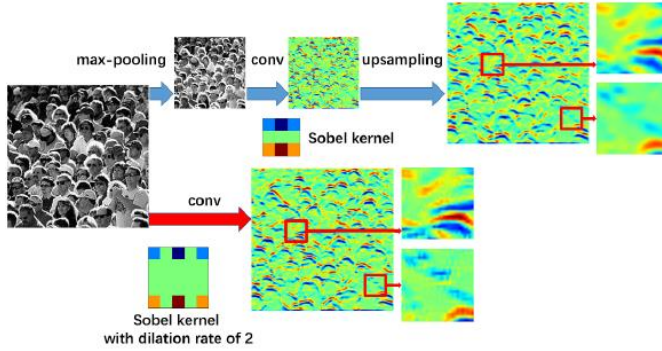
$y(m, n)$  adalah *output* dari *dilated convolution* dari *input*  $x(m, n)$  dan *filter*  $w(i, j)$  dengan lebar  $M$  dan  $N$ . Parameter  $r$  adalah *dilation* ratenya

*Dilated convolutional layers* sudah pernah dilakukan demonstrasi saat melakukan segmentasi dengan peningkatan akurasi dan merupakan alternative yang baik dari *pooling layer*. Layer ini merupakan pilihan terbaik, yang menggunakan kernel kecil untuk *alternative pooling* dan *convolutional layer*.



**Gambar 2.1** 3x3 kernel convolusi dengan *dilation rate* yang berbeda

Outputnya menghasilkan dimensi yang sama dengan input, yang artinya tidak direkomendasikan menggunakan *pooling* dan *deconvolutional layer*. Yang paling penting, *output* dari *dilated convolution* berisi informasi yang lebih detail.



**Gambar 2.2** Perbedaan *dilated convolution* dan *max-pooling*, *convolution* dan *upsampling*.

### 2.3.3 Training Method

#### A. Ground truth generation

Mengikuti metode dari pembuatan peta kerapatan, disini menggunakan *geometry-adaptive kernels* untuk mengatasi bagian gambar yang sangat padat. Dengan memblurkan tiap kepala menggunakan *Gaussian kernel* (dengan normalisasi 1), didapatkan distribusi spasial dari setiap gambar di dataset. *Geometry-adaptive kernel* didefinisikan sebagai berikut

$$F(x) = \sum_{i=1}^N \delta(x - x_i) \times G_{\sigma_i}(x), \text{ with } \sigma_i = \beta \bar{d}_i$$

Untuk setiap target object  $x_i$  dan *ground truth*  $\bar{d}$ ,  $d_i$  sebagai jarak rata-rata dari  $k$  nearest neighbor. Untuk mendapatkan peta kerapatan, dikalikan dengan *kernel Gaussian* dengan parameter  $\sigma$  (standard deviasi), dimana  $x$  adalah posisi pixel pada gambar.

## B. Augmentasi Data

Gambar dipotong menjadi 9 bagian dengan ukuran  $\frac{1}{4}$  dari gambar asli. 4 bagian awal mengandung  $\frac{1}{4}$  dari gambar tanpa *overlapping* dan 5 bagian lainnya terpotong random. Setelah itu, gambar dimirror, menghasilkan *double training set*.

## C. Detail Training

Disini menggunakan *straightforward way* untuk melatih CSRNet. 10 *convolutional layer* adalah *fine-tuned* dari *well-trained* VGG-16. Untuk layer lainnya, diinisialisasi dari *Gaussian inilization* *standard deviasi* 0.01. *Stochastic Gradient Descent (SGD)* diaplikasikan dengan *learning rate*  $1e-6$ . Digunakan juga *Euclidean Distance* untuk mengukur perbedaan antara *ground truth* dan estimasi dari peta kerapatan. *Loss function* yang digunakan adalah

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^N \| Z(X_i; \Theta) - Z_i^{GT} \|_2^2$$

Dimana  $N$  adalah ukuran dari training dan  $Z(X_i)$  adalah output yang dihasilkan oleh CSRNet.  $X_i$  merepresentasikan gambar input dan  $Z_i$  adalah *ground truth* dari hasil input

### 2.3.4 Model A dan Model B pada CSRNet

Hasil training menggunakan CNN arsitektur CSRNet akan menghasilkan sebuah model. Model ini didasarkan dari input yang ditrainingnya

Model A adalah model CSRNet yang memprediksi jumlah kerumunan dengan jumlah kerumunan sangat padat. Input trainingnya menggunakan dataset kerumunan *Shanghai* part A. Disana terdapat 482 gambar.

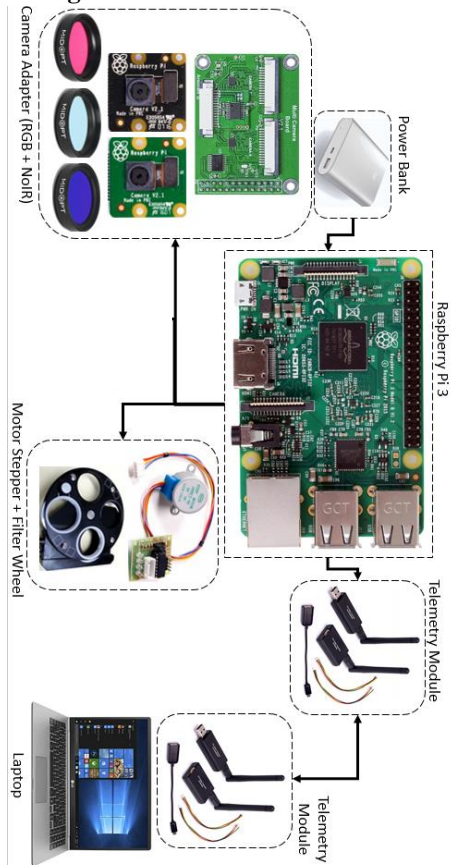
Model B adalah model CSRNet yang memprediksi jumlah kerumunan dengan jumlah kerumunan yang terpisah. Input trainingnya menggunakan dataset kerumunan *Shanghai* part B yang terdapat 716 gambar

## BAB III

### PERANCANGAN DAN PEMBUATAN SISTEM

Pada bab ini dibahas mengenai perencanaan dan perancangan dalam pembuatan kamera multispektral untuk mendeteksi kerumunan dalam lingkungan social distancing *Covid-19*.

#### 3.1 Perancangan Dan Pembuatan Sistem Kamera



**Gambar 3.1** Diagram Umum Sistem

Keterangan:

### **Powering System**

#### 1. Power Bank

Power Bank yang digunakan memiliki spesifikasi 5V/2A, 5000 mAh dan Multi Camera Adapter V2.1

Module dapat menyalakan sistem kurang lebih 2 Jam

### **Camera System**

Multiplexer yang dapat digunakan untuk kebutuhan jenis camera raspberry lebih dari 1.

#### 2. Camera Raspberry (RGB)

Kamera berjenis sensor RGB ini digunakan untuk mendapatkan gambar spectral pada range 600nm-700nm.

#### 3. Camera NoIR Raspberry

Kamera berjenis Raspberry Pi NOIR Camera ini yang akan dipasangkan pada spektral filter.

#### 4. Spektral Filter

3 Macam jenis filter spektral yang digunakan adalah jenis dengan range 770nm-790nm;855nm-890nm;928nm-955nm.

#### 5. Motor Stepper

Motor yang digunakan adalah motor stepper tipe 28BYJ dengan driver motor tipe ULN2003. Motor stepper digunakan untuk memutar wheel yang membawa filter

#### 6. Filter Wheel

Roda filter digunakan untuk tatakan filter spectral dan sebagai sarana untuk memudahkan motor stepper mengganti jenis filternya.

### **Telemetry System**

#### 7. Telemetry Module

Module yang dapat berfungsi sebagai receiver dan transmitter untuk mengirimkan sebuah data melewati serial UART



## Processing System

### 8. Raspberry Pi 3 B

Sebuah mini PC yang diletakan pada camera multispectral

### 9. Laptop

Sebagai Trasnmmitter gelombang radio yang akan dikirmkan ke camera spectral dan memproses data gambar

## Hardware System

### 10. Drone DJI Phantom 4

Untuk membantu mengambil data gambar pada camera spectral dan melihat lingkungan sekitar

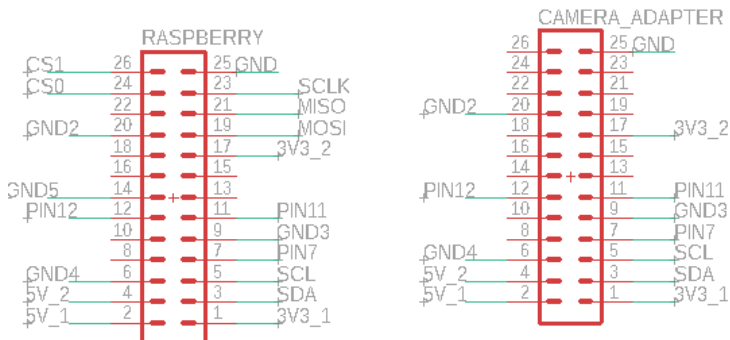
## 3.2 Perancangan Hardware

### 3.2.1 Perancangan dan Pembuatan Sistem Pengambilan Gambar

Pada sistem ini, akan dibuat board shield yang berguna untuk mengkombinasikan module-module yang terkoneksi dengan pin-pin GPIO pada Raspberry

#### a. Raspberry Pi 3 + Multi Camera Adapter

Untuk mengkoneksikan Multi Camera Adapter pada Raspberry Pi, dibutuhkan pin-pin yang terhubung ke Raspberry, yaitu 3 Pin GPIO, SDA, SCL, 2 Pin 5V dan 1 Pin 3,3 V.



**Gambar 3.2** Schematic Raspberry Pi dengan Camera Adaopter

3 Pin GPIO yaitu Pin 7, Pin 11 dan Pin 12 digunakan untuk menswitch, camera raspberry mana yang akan aktif.

Pin 7	Pin 11	Pin 12	Camera Yang Aktif
0	0	1	Camera A (Camera NoIR)
0	1	0	Camera B (Camera RGB)

**Tabel 3.1** GPIO pada Raspberry untuk switch Camera Raspberry

Pin I2C, yaitu SDA dan SCL juga berguna sebagai alamat mana yang akan diaktifkan sebagai Camera yang Online

Alamat I2C	Camera Yang Aktif
0x70 0x00 0x04	Camera A (Camera NoIR)
0x70 0x00 0x06	Camera B (Camera RGB)

**Tabel 3.2** Alamat I2C untuk menentukan Camera mana yang aktif

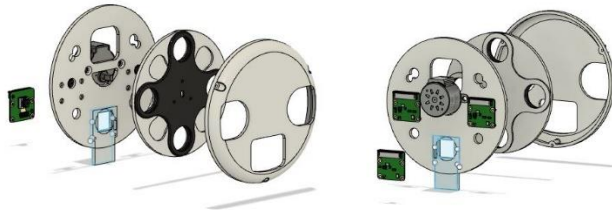
### 3.2.2 Perancangan dan Pembuatan Mekanik

Terdapat 3 Lubang Utama pada Camera, yaitu di sebelah kiri adalah camera NoIR, di kanan adalah camera RGB dan ditengah bagian bawah adalah camera thermal. Jarak antar titik tengah camera dengan titik tengah camera lainnya adalah 7 cm



**Gambar 3.3** Tampak depan Kamera Spectral

Di dalamnya lagi, terdapat 4 lubang yang berfungsi untuk penempatan filter, yaitu filter Midopt 770nm – 790nm, filter Midopt 855nm – 890nm, filter 929nm – 955nm dan tanpa filter.



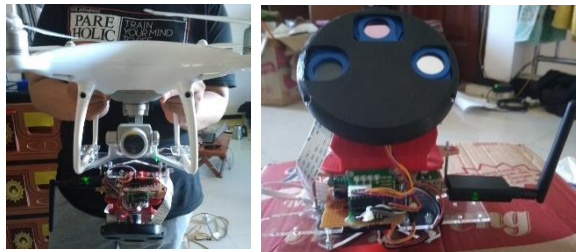
**Gambar 3.4** Tampak samping Kamera Spectral

Pengambilan gambar tidak 0 derajat vertical menghadap bawah, melainkan dengan sudut 30 derajat.



**Gambar 3.5** Design alas camera spectral 30 derajat

Dan di bawah ini merupakan gambar asli dari kamera spectral secara keseluruhan.

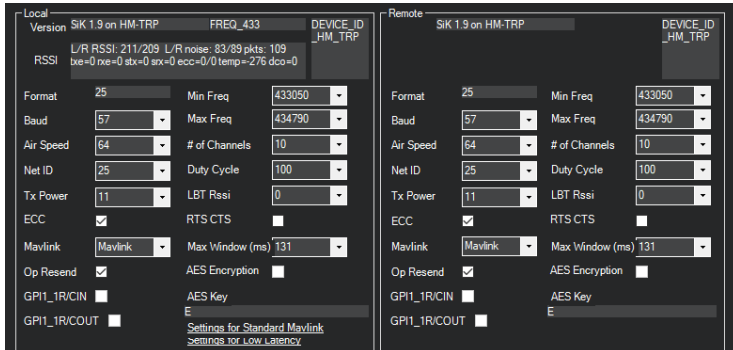


**Gambar 3.6** Gambar keseluruhan kamera spectral beserta dronnya

### 3.3 Perancangan Software

#### 3.3.1 Perancangan Pengiriman Perintah Via Module Telemetry

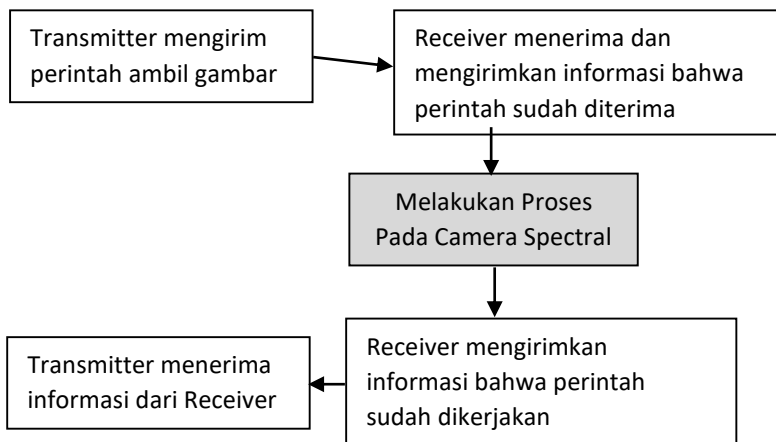
Terdapat 2 Module Telemetry yang keduanya dapat berkomunikasi ketika memiliki jalur yang sama. Module telemetry ini dapat disetting terlebih dahulu menggunakan software Ardu Pilot.



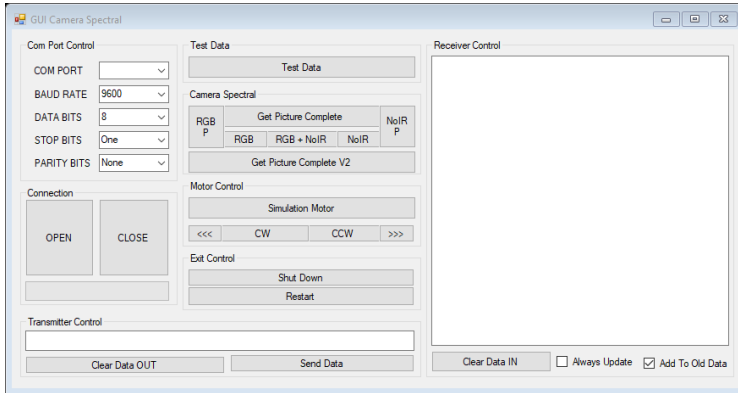
**Gambar 3.7** Setting Konfigurasi Telemetry Module pada Ardu Pilot

Dengan software ini, kita akan lebih mudah dalam mensetting module telemetry yang akan digunakan, seperti nilai baudranya, air speed, dan Net IDnya. Pastikan bahwa kedua module telemetry mempunyai settingan yang sama agar dapat bisa berkomunikasi

Setelah itu, baru bisa melaksanakan prosedur pengambilan gambar menggunakan telemetry module



Transmitter adalah Laptop/PC yang terpasang Module Telemetry menggunakan USB. Karena menggunakan serial dan membutuhkan perintah yang beragam kepada camera spectral, maka dibuatkan GUInya.



**Gambar 3.8** GUI pada PC untuk mengirimkan perintah ambil gambar

Untuk menggunakan GUI, diperlukan setting COM PORT yang sesuai dengan device manager pada laptop. Baudrate, Data Bits, Stop Bits dan Parity Bits juga diisi sesuai program Telemetry yang ada pada camera spectral.

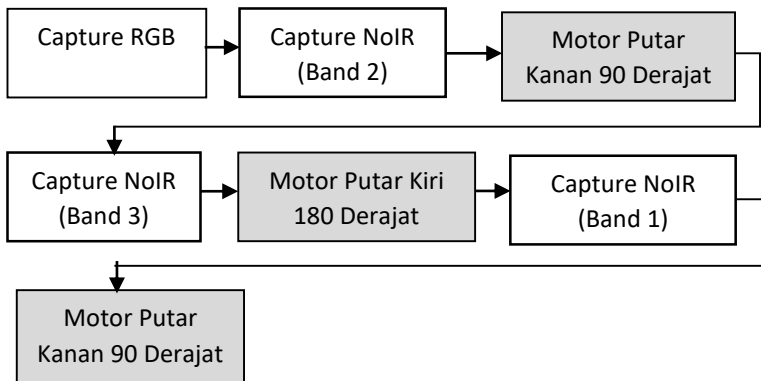
No.	Nama GUI	Fungsi
1	Test Data	Mengecheck koneksi antar module telemetry
2	Get Picture Complete	Ambil gambar camera RGB dan 3 band NoIR
3	RGB	Ambil gambar RGB saja
4	RGB + NoIR	Ambil gambar RGB dan NoIR
5	NoIR	Ambil gambar NoIR saja
6	RGB P	Preview camera RGB
7	NoIR P	Preview camera NoIR
8	Get Picture Complete V2	Ambil gambar camera RGB dan 3 band NoIR + NoIR tanpa filter
9	Simulation Motor	Menggerakan/mensimulasikan pergerakan motor filter

10	CW	Berputar searah jarum jam 90 derajat
11	CCW	Berputar berlawanan arah jarum jam 90 derajat
12	<<<	Berputar searah jarum jam untuk kalibrasi
13	>>>	Berputar berlawanan arah jarum jam untuk kalibrasi
14	Shut Down	Mematikan raspberry
15	Restart	Menrestart raspberry
16	Transmitter Control	Untuk mentransmikan variable tertentu
17	Receiver Control	Untuk menerima variable tertentu

### 3.3.2 Proses Pengambilan Gambar Pada Camera Spectral

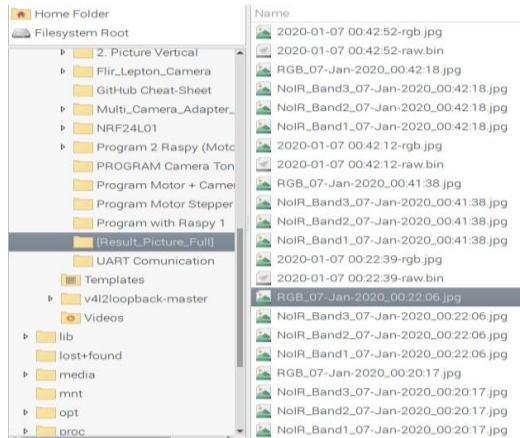
#### a. Camera RGB + 3 Band Camera NoIR

Pertama, pastikan dahulu posisi filter pada posisi yang tepat. Yaitu, filter bening (tanpa filter) berada di depan camera RGB



**Gambar** Flowchart get image

Hasil dari pengambilan gambar tersebut, akan disimpan dalam Raspberry dengan nama khusus sesuai dengan jenis camera yang digunakan juga tanggal pengambilan gambar



**Gambar 3.9** Hasil Gambar dari 1 Folder

### 3.3.3 Proses Image Processing Menggunakan CSRNet (Memprediksi Jumlah Kerumunan)

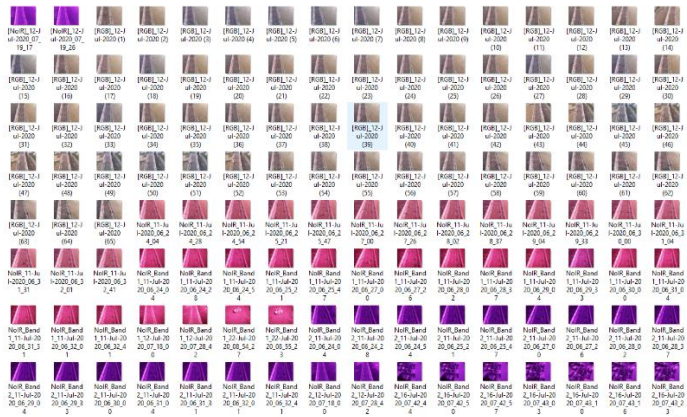
Dalam proses memprediksi jumlah kerumunan, disini menggunakan metode CNN yang lebih dalam untuk menangkap fitur yang high-level dan menghasilkan peta kepadatan yang berkualitas tinggi tanpa memperluas kompleksitas network besar-besaran.

Menggunakan CNN dengan arsitektur CSRNet, dibagi menjadi 3 tahap, data preprocessing, modelling dan yang terakhir predicting

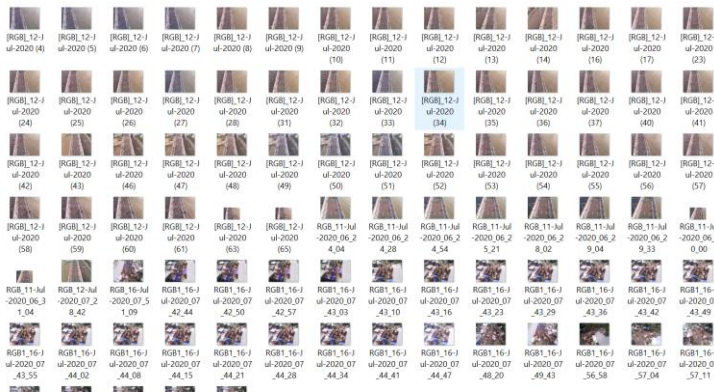
#### 3.3.3.1 Data Preprocessing

Gambar yang sebelumnya diambil menggunakan camera spectral akan ditandai terlebih dahulu posisi dari masing-masing kerumunan yang ada dalam tiap gambar serta jumlah kerumunan itu sendiri (ground\_truth)

Gambar yang didapat dari camera spectral dipisah menjadi 4 folder, yaitu untuk Gambar RGB, NoIR Band 1, NoIR Band 2 dan NoIR Band 3. Dalam proses pembuatan dataset, disini hanya menggunakan gambar hasil RGB saja yang berjumlah 91 Gambar yang diambil dengan 3 waktu berbeda



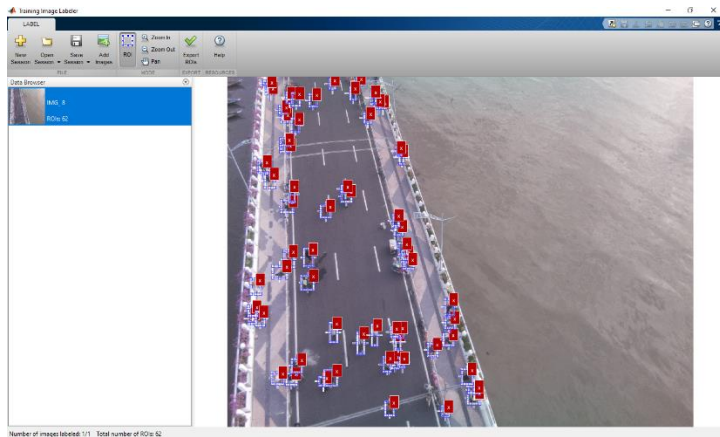
**Gambar 3.10** Hasil RGB dan Noir dengan 3 Band dalam 1 folder



**Gambar 3.11** 91 Gambar Kerumunan RGB

Gambar RGB kerumunan tersebut ditandai menggunakan fitur ROI (Region Of Interest) yang ada pada aplikasi Training Image Labeler – Mathlab 2015. Akan didapatkan informasi berapa jumlah kerumunan (ground\_truth) yang ada dalam gambar, beserta posisinya (x,y).





**Gambar 3.12** ROI Kerumunan menggunakan Training Image Labeler – Mathlab 2015

The screenshot shows the 'Variables - image\_infoX.location' window. It displays a variable named 'image\_infoX' with a sub-variable 'location' which is a 6x2 double matrix. The matrix contains the following data:

	1	2	3	4
1	1694	2224		
2	1702	2179		
3	1651	2057		
4	1514	2318		
5	1113	2287		
6	649	2113		

**Gambar 3.13** Informasi banyaknya kerumunan beserta lokasinya

Karena data kerumunan yang diambil menggunakan kamera spectral tidak begitu menghasilkan gambar kerumunan yang cukup padat, digunakan juga dataset kerumunan yang sangat populer yaitu Dataset Shanghai.

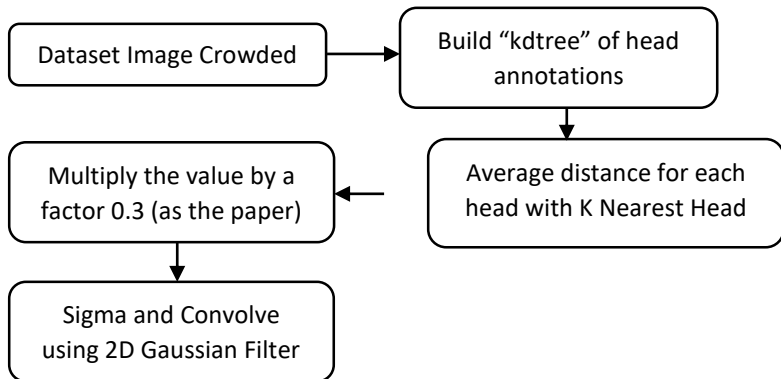
Dataset Shanghai ini dibagi menjadi 2, yaitu part A yang berisikan dataset kerumunan dengan tingkat kepadatan yang tinggi dan part B dengan tingkat kepadatan yang tersebar. Dataset part A ini

berisikan 482 gambar kerumunan dengan 241.667 orang di dalamnya. Pada dataset part B terdapat 716 gambar.

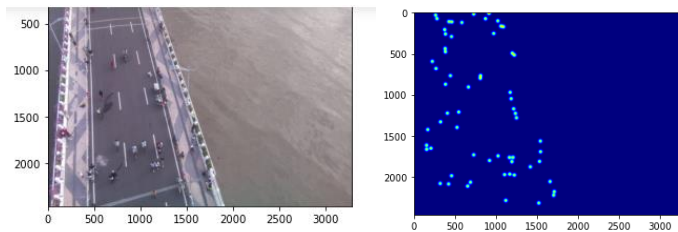


**Gambar 3.14** Dataset Shanghai part A (kiri) dan part B (kanan)

Karna dataset sudah siap, selanjutnya mengkonversi sebuah gambar menjadi sebuah gambar peta kerumunan. Dengan melihat masing-masing kepala yang ada pada gambar, dideteksi dengan “kdtree” dan dihitung jarak rata-rata antar kepala dengan K Nearest Head. Selanjutnya dikonversi menggunakan 2D Gaussian Filter



**Gambar** Flowchart konversi ke *density map*



**Gambar 3.15** Gambar kerumunan RGB (kiri) dan Peta Kerapatan (kanan)

### 3.3.3.2 Modelling (CSRNet-Keras)

Dengan dataset Shanghai part A dan part B, dibuatlah sebuah model yang nantinya dapat memprediksi jumlah kerumunan. Dataset part A digunakan untuk sebuah model dengan kerumunan tingkat kepadatan tinggi dan part B untuk tingkat kepadatan tersebar.

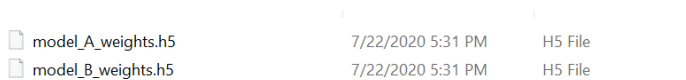
Menggunakan arsitektur CSRNet, yaitu pada bagian front-end menggunakan 13 pretrained layers dari model VGG-16 (10 convolution layer dan 3 max pooling layer) dan bagian back-end dengan dilated convolution layers.

Proses Modelling ini menggunakan pemrograman python, yaitu pada Jupyter Notebook. Library-library yang digunakan adalah sebagai berikut

Library	Versi
Python	3.7
Tensorflow	2.1.0
Keras	2.3.1
Scipy	1.18.5
Pillow	7.2.0
OpenCV	3.4.1

**Tabel 3.3** Libray yang digunakan saat modelling

Dengan proses modelling yaitu 4.5 jam dengan spesifikasi laptop Intel Core i7-855U (1.80Ghz – 2.00 GHz) dan ram 8.00 GB menghasilkan model A dan model B.

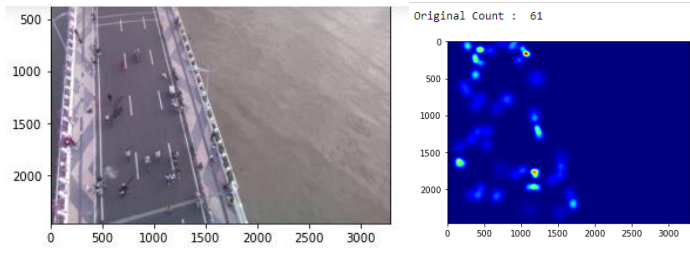


model_A_weights.h5	7/22/2020 5:31 PM	H5 File
model_B_weights.h5	7/22/2020 5:31 PM	H5 File

**Gambar 3.16** File Model A dan Model B hasil modelling

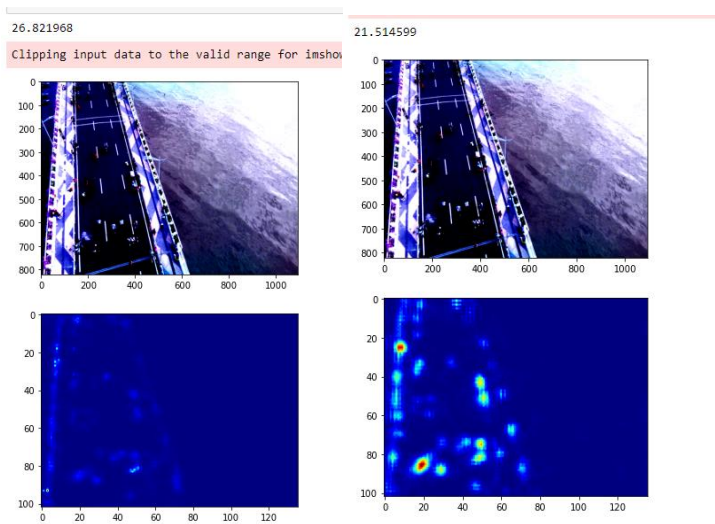
### 3.3.3.3 Predicting

Dengan model yang sudah didapatkan, dilakukan sebuah prediksi jumlah kerumunan pada sebuah gambar dengan menginputkan file gambarnya. Akan ditampilkan juga gambar RGB dan peta kepadatan saat preprocessing. Jumlah asli kerumunan (ground\_truth) akan ditampilkan di atas peta kepadatan.



**Gambar 3.17** Gambar RGB dan Peta Kepadatannya (61 orang)

Dengan sebuah permodelan A dan juga permodelan B, akan ditampilkan peta kepadatan yang baru sesuai hasil permodelan. Ditampilkan juga hasil prediksi jumlah kerumunan yang ada



**Gambar 3.18** Hasil prediksi jumlah kerumunan dengan Model A (kiri) 26 orang dan Model B (kanan) 21 orang

## BAB IV

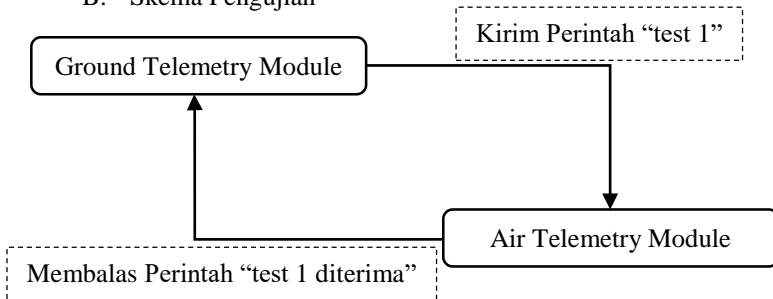
### PENGUJIAN DAN ANALISA

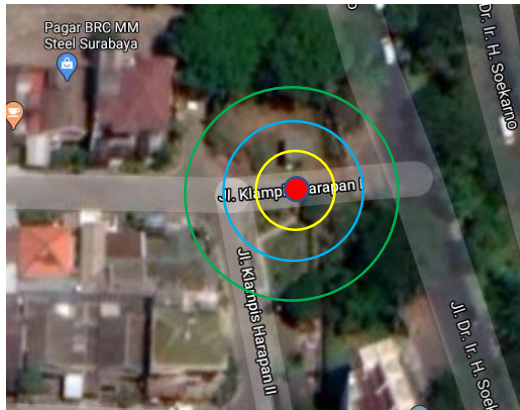
Dalam bab ini akan dijelaskan mengenai hasil uji coba perangkat sistem. Selain itu, dari hasil uji coba yang telah dilakukan akan dianalisa apakah rancangan ini dapat memenuhi tujuan yang akan dicapai seperti yang telah dipaparkan pada Bab I. Berikut ini adalah beberapa tahap pengujian yang telah dilakukan :

#### 4.1 Pengujian Range Komunikasi Telemetry

Pengujian ini bertujuan untuk melihat jarak maksimal komunikasi antar module telemetry. Dilakukan dengan membaca kembali perintah yang dikirim dari telemetry yang ada pada camera spectral ke telemetry yang ada di ground

- A. Alat dan Bahan
  - a. Camera Spectral (air module telemetry)
  - b. Laptop dengan GUI membaca data perintah telemetry (ground module telemetry)
- B. Skema Pengujian





**Gambar** Test jarak komunikasi telemetry (merah = ground, kuning = 10 m, biru = 20 m dan hijau = 30 m)

### C. Langkah Pengujian

- a. Terbangkan kamera spectral dengan radius yang ditentukan 10 meter (kuning)
- b. Atur ketinggian dari kamera spectral dengan tinggi 15 meter
- c. Kirimkan perintah test dengan menekan tombol “Test 1” pada GUI
- d. Tunggu beberapa saat sampai pada layar receiver bertuliskan “Test 1 diterima”, yang artinya data terkirim.
- e. Ulangi langkah “c” dengan angka yang otomatis berubah menjadi “test 2”, “test 3” dan seterusnya sampai 5x
- f. Ulangi langkah “b” dengan memberikan variasi ketinggian 20 meter, 25 meter dan 30 meter
- g. Ulangi langkah “a” dengan variasi radius 20 meter dan 30 meter

- D. Hasil dan Analisa
- Ketinggian 15 meter

10 Meter *Kuning	
No.	Ground
1	OK
2	OK
3	OK
4	OK
5	OK
20 Meter *Biru	
No.	Ground
1	OK
2	OK
3	OK
4	OK
5	OK

30 Meter *Hijau	
No.	Ground
1	OK
2	OK
3	OK
4	OK
5	OK

- Ketinggian 20 Meter

10 Meter *Kuning	
No.	Ground
1	OK
2	OK
3	OK
4	OK
5	OK
20 Meter *Biru	
No.	Ground
1	OK
2	OK
3	OK
4	OK
5	OK

30 Meter *Hijau	
No.	Ground
1	FAIL
2	FAIL
3	FAIL
4	FAIL
5	FAIL



- Ketinggian 25 Meter

10 Meter Kuning	
No.	Ground
1	OK
2	OK
3	OK
4	OK
5	OK
20 Meter *Biru	
No.	Ground
1	FAIL
2	OK
3	OK
4	OK
5	FAIL

30 Meter *Hijau	
No.	Ground
1	FAIL
2	FAIL
3	FAIL
4	FAIL
5	FAIL

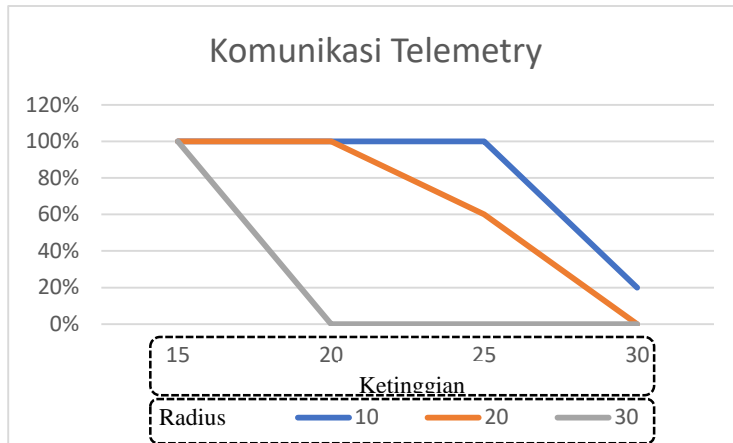
- Ketinggian 30 Meter

10 Meter Kuning	
No.	Ground
1	FAIL
2	FAIL
3	FAIL
4	FAIL
5	OK
20 Meter *Biru	
No.	Ground
1	FAIL
2	FAIL
3	FAIL
4	FAIL
5	FAIL

30 Meter *Hijau	
No.	Ground
1	FAIL
2	FAIL
3	FAIL
4	FAIL
5	FAIL

Data yang diterima oleh ground telemetry module lebih bagus pada ketinggian 15 meter dengan radius bisa sampai 30 meter. Untuk ketinggian 20 meter lebih baik digunakan pada radius maksimal 20 meter

saja, karena ketika radius sampai 30 meter, data kadang diterima kadang tidak. Untuk ketinggian 25 meter, jarak radius maksimal 10 meter saja, selebihnya kadang tersambung kadang tidak. Tidak disarankan dengan ketinggian 30 meter, karena pada radius 10 meter saja, dalam 5x percobaan, hanya 1x data diterima.



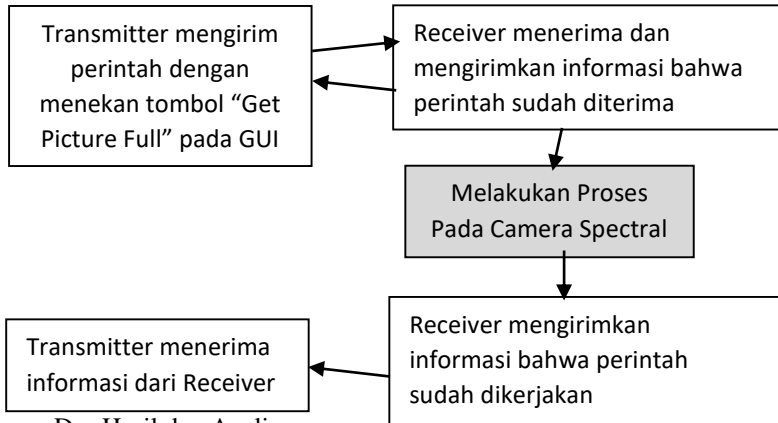
**Tabel 3.4** Perbandingan Komunikasi Telemetry antara Radius dan Ketinggi

Jarak jangkauan ini dipengaruhi oleh air speed yang tx power yang diseting pada telemetry module saat menggunakan aplikasi mission planer.

## 4.2 Pengambilan Gambar Kamera Spectral

Pengambilan gambar ini dilakukan agar mendapat 4 fitur berbeda dari objek yang ditangkap, mulai dari kamera RGB (600nm - 700nm), NoIR band1 (770nm - 790nm), NoIR band2 (855nm - 890nm) dan NoIR band3 (928nm - 955nm) dengan resolusi yang sama 1093 pixel x 821 pixel. Proses pengambilan gambar dilakukan dengan ketinggian yang tetap yaitu 15 meter dengan sudut pandang 30 derajat.

- A. Alat dan Bahan
- Camera Spectral
  - Laptop dengan GUI
- B. Skema Pengujian



- D. Hasil dan Analisa
- NoIR Band 1 (770nm – 790nm)



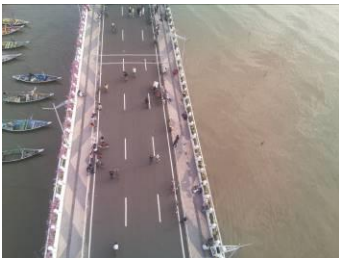
- NoIR Band 2 (855nm – 890nm)



- NoIR Band 3 (928nm – 955nm)



- RGB (600nm - 700nm)



Dari keempat gambar yang diambil, untuk saat ini gambar yang nantinya akan diolah menjadi dataset saat training adalah gambar RGB. Pada masa mendatang, akan ditambahkan gambar NoIR Band1, Band2 dan Band3 agar mendapat fitur yang lebih kuat lagi untuk mendeteksi sebuah kerumunan.

### 4.3 Pengujian Model CSRNet

Pengujian dilakukan dengan 2 model, model A yaitu model untuk tipe kepadatan tinggi dan model B yaitu model tipe kepadatan yang tersebar. Model A dihasilkan dari proses modelling menggunakan dataset Shanghai part A dan Model B menggunakan dataset Shanghai part B

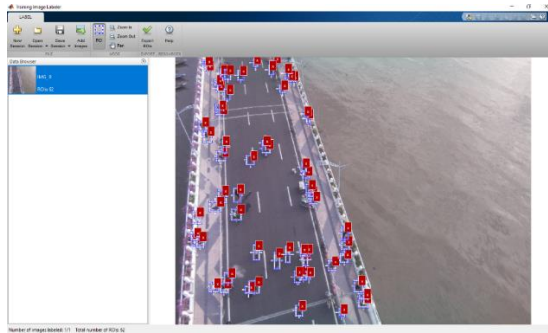
#### A. Langkah Pengujian

- **Preprocessing** (Membuat Peta Kerapatan)
- a. Mengambil sebuah gambar dataset sesuai dengan path yang telah ditentukan



```
img_paths = []
for path in path_sets:
    for img_path in glob.glob(os.path.join(path,
        '*.jpg')):
        img_paths.append(img_path)
print(len(img_paths))
```

- b. Mengambil data jumlah kerumunan beserta posisinya dengan format .mat



```
mat = io.loadmat(img_path.replace('.jpg', '.mat').replace('images', 'ground truth').replace('IMG ', 'GT IMG '))
```

- c. Membuat matrix nol sesuai dengan dimensi gambar yang diambil

```
k = np.zeros((img.shape[0],img.shape[1]))
```

- d. Mengambil data .mat 2 array yang berisikan [posisi posisi] tiap kerumunan

```
gt = mat["image_info"][0,0][0,0][0]
```

- e. Menjadikan tiap-tiap nilai pixel dengan posisi yang sama pada tiap kerumunan bernilai "1"

```
for i in range(0,len(gt)):
    if int(gt[i][1])<img.shape[0] and int(gt[i][0])<img.shape[1]:
        k[int(gt[i][1]),int(gt[i][0])]=1
```

- f. Masuk ke fungsi Gaussian Filter

```
k = gaussian_filter_density(k)
```

g. Menghitung tiap kerumunan pada dataset

```
gt_count = np.count_nonzero(gt)
```

h. Mencari K Nearest Neighbours menggunakan KDTree

```
pts =  
np.array(list(zip(np.nonzero(gt)[1].ravel(),  
np.nonzero(gt)[0].ravel())))  
  
leafsize = 2048
```

i. Membuat kdtree

```
tree = scipy.spatial.KDTree(pts.copy(),  
leafsize=leafsize)
```

j. Mendapatkan jarak antar 4 kerumunan beserta posisinya

```
distances, locations = tree.query(pts, k=4)
```

k. Menghitung sigma jarak

```
for i, pt in enumerate(pts):  
    pt2d = np.zeros(gt.shape,  
dtype=np.float32)  
    pt2d[pt[1],pt[0]] = 1.  
    if gt_count > 1:  
        sigma =  
(distances[i][1]+distances[i][2]+distances[i][3]  
) * 0.1  
    else:
```

```

sigma =
np.average(np.array(gt.shape))/2./2. #case: 1
point

```

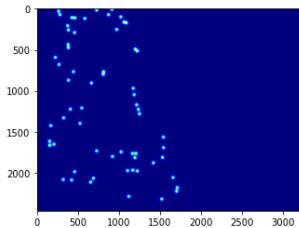
### 1. Convolve dengan Gaussian filter

```

density +=
scipy.ndimage.filters.gaussian_filter(pt2d,
sigma, mode='constant')

```

### m. Membuat peta kerapatan dengan format .h5



```

with
h5py.File(img_path.replace('.jpg','h5').replace
('images','ground_truth'), 'w') as hf:

    hf['density'] = k

```

### - **Modelling**

- a. Membuat model jaringan dengan jumlah filter yang ditentukan sesuai program, ukuran kernel 3x3, fungsi aktivasi menggunakan relu dan selalu dilakukan padding 1

```

rows = None
cols = None

#Batch Normalisation option

batch_norm = 0
kernel = (3, 3)

```



```

init = RandomNormal(stddev=0.01)
model = Sequential()

#custom VGG:

if(batch_norm):
    model.add(Conv2D(64, kernel_size = kernel,
input_shape = (rows,cols,3),activation = 'relu',
padding='same'))
        model.add(BatchNormalization())
    model.add(Conv2D(64, kernel_size =
kernel,activation = 'relu', padding='same'))
        model.add(BatchNormalization())

model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(128,kernel_size
= kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(128,kernel_size
= kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())

model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(256,kernel_size
= kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(256,kernel_size
= kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(256,kernel_size
= kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())

model.add(MaxPooling2D(strides=2))

```

```

        model.add(Conv2D(512,
kernel_size = kernel,activation = 'relu',
padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(512,
kernel_size = kernel,activation = 'relu',
padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(512,
kernel_size = kernel,activation = 'relu',
padding='same'))
        model.add(BatchNormalization())

    else:
        model.add(Conv2D(64, kernel_size
= kernel,activation = 'relu',
padding='same',input_shape = (rows, cols, 3),
kernel_initializer = init))
        model.add(Conv2D(64, kernel_size
= kernel,activation = 'relu', padding='same',
kernel_initializer = init))

model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(128,kernel_size
= kernel, activation = 'relu', padding='same',
kernel_initializer = init))
        model.add(Conv2D(128,kernel_size
= kernel, activation = 'relu', padding='same',
kernel_initializer = init))

model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(256,kernel_size
= kernel, activation = 'relu', padding='same',
kernel_initializer = init))

```

```

        model.add(Conv2D(256, kernel_size
= kernel, activation = 'relu', padding='same',
kernel_initializer = init))
        model.add(Conv2D(256, kernel_size
= kernel, activation = 'relu', padding='same',
kernel_initializer = init))

model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(512,
kernel_size = kernel, activation = 'relu',
padding='same', kernel_initializer = init))
        model.add(Conv2D(512,
kernel_size = kernel, activation = 'relu',
padding='same', kernel_initializer = init))
        model.add(Conv2D(512,
kernel_size = kernel, activation = 'relu',
padding='same', kernel_initializer = init))

#Conv2D
        model.add(Conv2D(512, (3, 3),
activation='relu', dilation_rate = 2,
kernel_initializer = init, padding = 'same'))
        model.add(Conv2D(512, (3, 3),
activation='relu', dilation_rate = 2,
kernel_initializer = init, padding = 'same'))
        model.add(Conv2D(512, (3, 3),
activation='relu', dilation_rate = 2,
kernel_initializer = init, padding = 'same'))
        model.add(Conv2D(256, (3, 3),
activation='relu', dilation_rate = 2,
kernel_initializer = init, padding = 'same'))

```

```

        model.add(Conv2D(128, (3, 3),
activation='relu', dilation_rate = 2,
kernel_initializer = init, padding = 'same'))
        model.add(Conv2D(64, (3, 3),
activation='relu', dilation_rate = 2,
kernel_initializer = init, padding = 'same'))
        model.add(Conv2D(1, (1, 1),
activation='relu', dilation_rate = 1,
kernel_initializer = init, padding = 'same'))

```

#### b. Menghitung Euclidean distance loss

```

def euclidean_distance_loss(y_true, y_pred):
    # Euclidean distance as a measure of loss
    (Loss function)
    return K.sqrt(K.sum(K.square(y_pred -
y_true), axis=-1))

```

#### c. Inisialisasi berap epoch

```

model.fit_generator(train_gen, epochs=1, steps_per
_epoch= 700 , verbose=1) #700

```

#### d. Save model dengan format .h5

```

save_mod(model, "weights/model_A_weights.h5", "mod
els/Model.json")

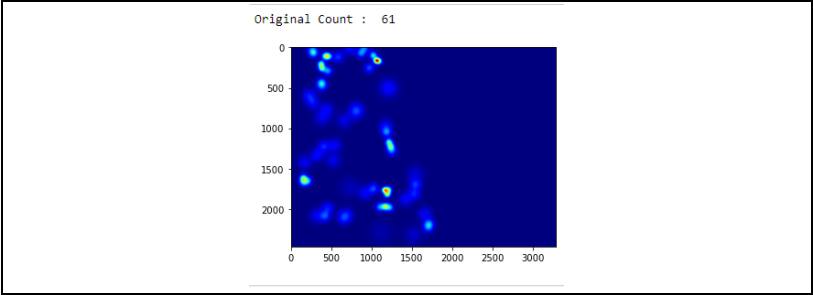
```

### B. Data dan Analisa

#### - Test 1 (Jembatan Suroboyo)

Ground\_Truth : 61 People

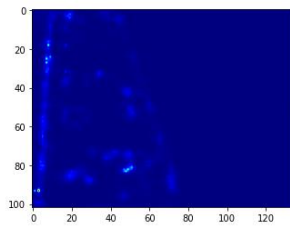
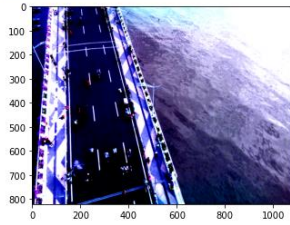




### Model A (High Density) : 26 People (Error = 57%)

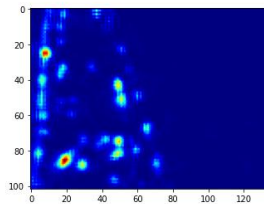
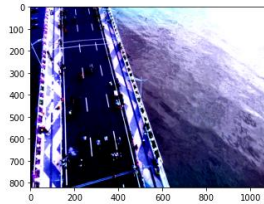
26.821968

Clipping input data to the valid range for imshow



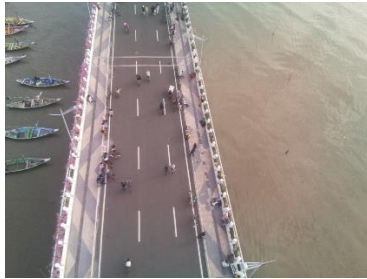
### Model B (Sparse Density) : 21 People (Error = 65%)

21.514599



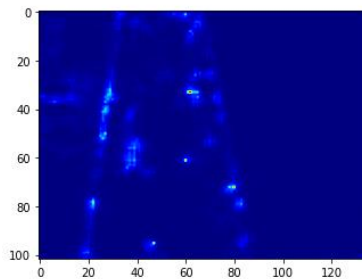
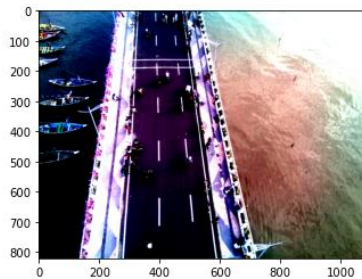
- Test 2 (Jembatan Suroboyo 2)

Ground Truth: 59 People



Model A (High Density) : 23 People (Error = 61%)

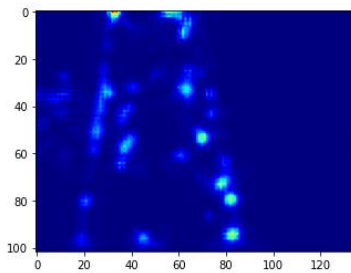
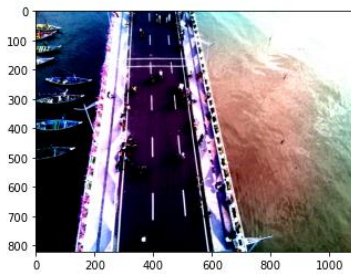
23.971191



Model B (Sparse Density) : 18 People (Error = 69%)

18.500908

Clipping input data to the valid range for in



- Test 3 (Pasar Pacar Keling)

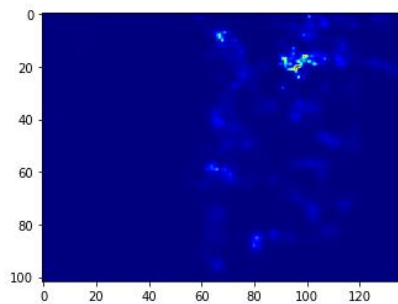
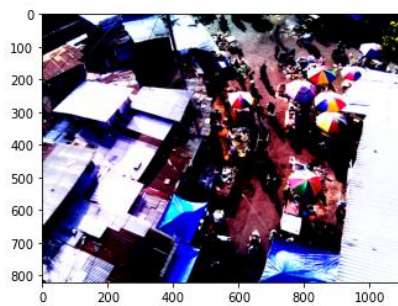
Ground Truth: 31 People



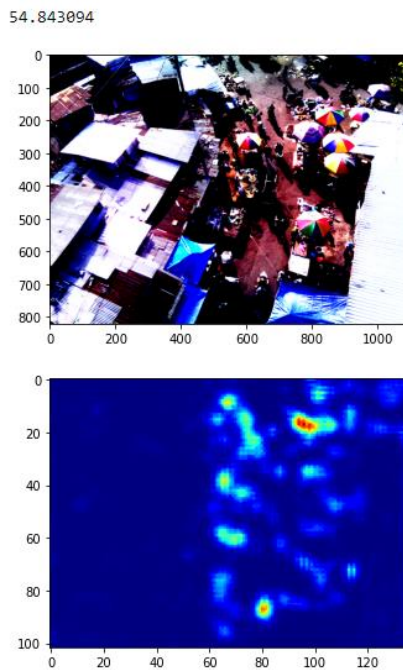


Model A (High Density) : 84 People (Error = 170%)

84.28146



Model B (Sparse Density) : 54 People (Error = 74%)

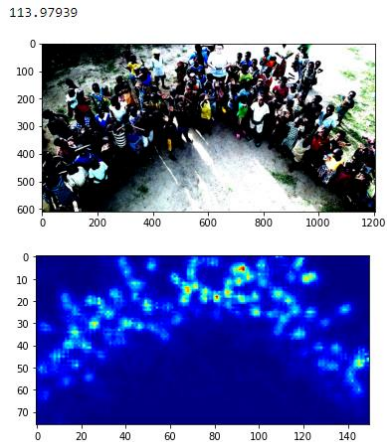


- Test 4 (From Internet)

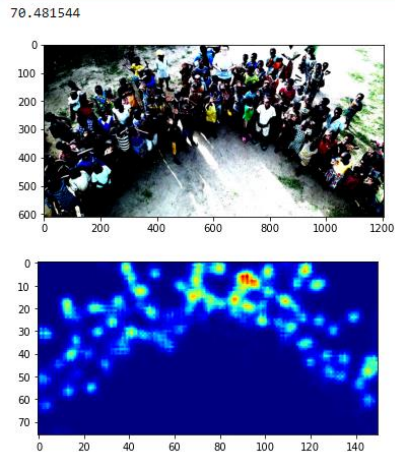
*Ground Truth: 88 People*



*Model A (High Density) : 113 People (Error = 28%)*



*Model B (Sparse Density) : 70 People (Error = 20%)*



Prediksi jumlah kerumunan dari gambar yang diambil menggunakan kamera spectral, yaitu menggunakan kamera RGB saja

(tanpa NoIR) dengan model A dan model B menghasilkan error yang sangat besar. Paling kecil dengan error 57% dan plaing besar 170%.

Prediksi dari gambar Jembatan Suroboyo 1 dan 2 menghasilkan error yang cukup besar dikarenakan pengambilan gambar dengan camera spectral terlalu tinggi sehingga program sulit membedakan kerumunan dan bukan. Gambar yang didapat dari kamera spectral juga tidak memperlihatkan sebuah kerumunan.

Prediksi dari gambar Pasar Pacar Keling menghasilkan error yang sangat besar di model A maupun model B. Ini dikarenakan terdapat beberapa objek yang dikira kerumunan orang, ternyata bukan, terlihat pada peta kerapatan yang dihasilkan.

Akan tetapi, prediksi gambar yang didapat dari internet, yang berisikan kerumunan orang yang diambil dari jarak dekat menghasilkan error yang lebih rendah, yaitu 20% pada model B dan 28% pada model A. Ini dikarenakan pengambilan gambar dengan jarak yang tidak tinggi, yaitu  $\pm 5$  meter. Gambar yang diambil juga terlihat jelas berkerumun, oleh karena itu prediksinya tidak jauh berbeda dengan nilai ground truthnya.

Dalam percobaan memprediksi jumlah sebuah kerumunan, terlihat jika gambar yang diambil dengan jarak 15 meter dengan resolusi  $1093 \times 821$ , hasil prediksinya jauh dengan nilai ground truthnya. Ini dikarenakan kerumunan yang didapatkan pada gambar tidak terlalu jelas. Pada peta kerapatannya, terlihat benda-benda yang bukan kerumunan seperti pagar, seng atap rumah dideteksi juga sebagai suatu kerumunan.

Faktor lainnya adalah saat mentraining model A dan Model B menggunakan CSRNet, data inputnya tidak ada gambar kerumunan yang diambil menggunakan drone. Model yang terbentuk kurang sesuai dengan gambar testingnya, yaitu gambar kerumunan yang diambil menggunakan drone.

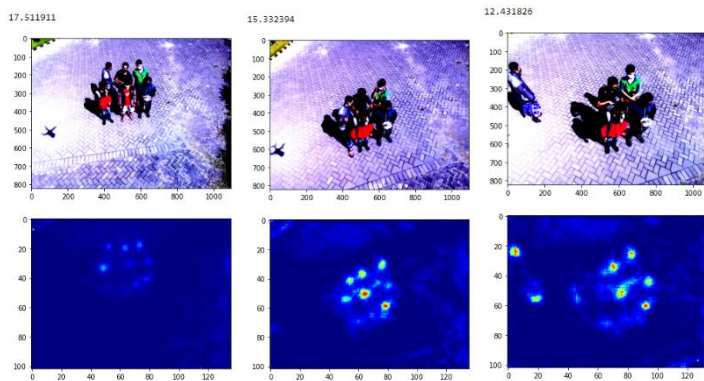
#### 4.4 Pengujian Model (Ketinggian 5 meter)

Pengujian dilakukan dengan jumlah 5 orang sampai 6 orang dengan ketinggian pengambilan object gambar tetap (5 meter). Pengujian dilakukan untuk menghitung kira-kira berapa jumlah minimal yang pas untuk sebuah kerumunan

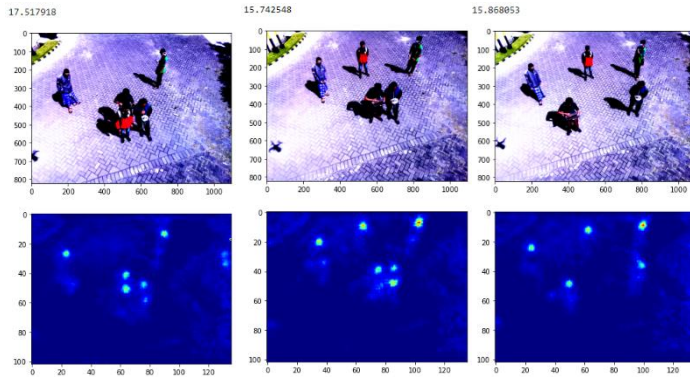
Objek 5 orang ini nantinya akan berpisah menjadi 4 orang, 3 orang, 2 orang dan 1 orang.

##### A. Data dan Analisa

##### - Model A

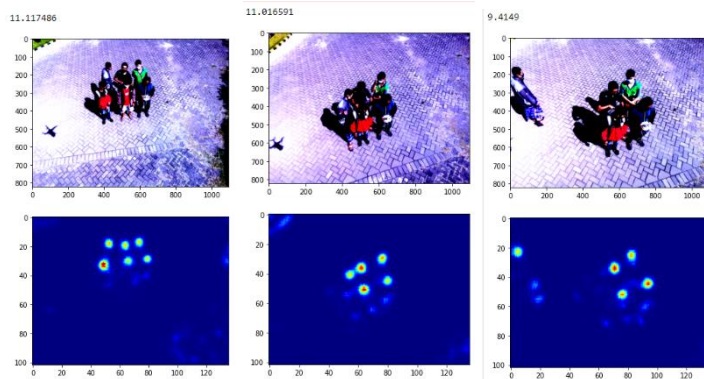


ground truth = 6	= 5	= 4   1
Predict = 17	= 15	= 12
Error = 183%	= 200%	= 140%

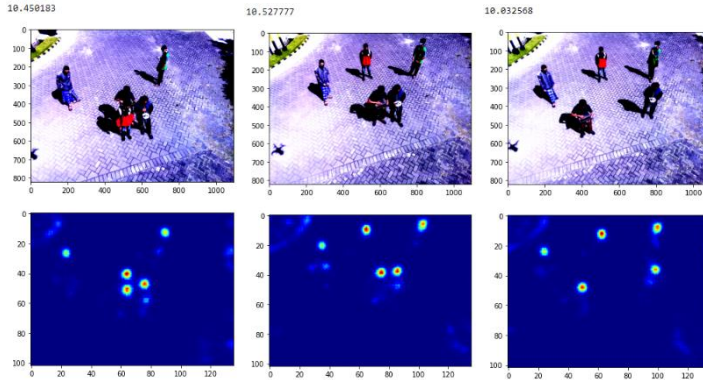


ground truth = 3 1 1	= 2 1 1 1	= 1 1 1 1 1
Predict = 17	= 15	= 15
Error = 183%	= 200%	= 200%

- **Model B**



ground truth = 6	= 5	= 4 1
Predict = 11	= 11	= 9
Error = 83%	= 120%	= 80%



ground truth =	3 1 1	= 2 1 1 1	= 1 1 1 1 1
Predict	= 10	= 10	= 10
Error	= 100%	= 100%	= 100%

Prediksi yang dihasilkan oleh model A dan model B pada gambar sangatlah besar. Hampir semua diatas 100%. Ini dikarenakan gambar yang diambil masih belum terlihat berkerumun. Oleh karnanya da bagian diluar kerumunan yang tetap dihitung sebagai kerumunan. Dapat terlihat pada peta kerapatannya

Meskipun kerumunan dalam keadaan terpisah, seperti yang semula 5 orang menjadi 4 orang & 1 orang, sampai masing-masing 1 orang tetap memprediksi nilai yang hamper sama dengan prediksi 5 orang.

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

1. Jangkauan komunikasi antar modul telemetry adalah pada ketinggian maksimal 15 meter dengan radius maksimal 30 meter
2. Pada kamera spectral, untuk mengambil gambar object kerumunan sementara digunakan 1 kamera saja, yaitu kamera RGB
3. Gambar yang diambil untuk memprediksi sebuah kerumunan sebaiknya maksimal ketinggian 5 meter, dengan kepadatan dalam 1 gambar tinggi.
4. Untuk dataset yang digunakan saat membuat permodelan digunakan gambar dengan kepadatan lebih tinggi.

#### **5.2 Saran**

Oleh karena itu, perlu adanya saran yang dibuat agar kedepannya penelitian ini dapat lebih baik. Saran yang dapat diberikan sebagai berikut :

1. Pengambilan data perlu diperhatikan sebelum menjadikan data tersebut sebagai dataset untuk sebuah permodelan
2. Kamera NoIR akan digunakan di penelitian selanjutnya untuk mendapatkan model deteksi kerumunan yang lebih baik sehingga error akan mengecil



## DAFTAR PUSTAKA

- CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes Yuhong Li<sup>1,2</sup>, Xiaofan Zhang<sup>1</sup>, Deming Chen<sup>1</sup> <sup>1</sup>University of Illinois at Urbana-Champaign <sup>2</sup>Beijing University of Posts and Telecommunications
- Single-Image Crowd Counting via Multi-Column Convolutional Neural Network (Yingying Zhang Desen Zhou Siqin Chen Shenghua Gao Yi Ma) Shanghaitech University
- Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. CoRR, abs/1706.05587, 2017.
- Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu Switching convolutional neural network for crowd counting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 1, page 6, 2017.
- <https://www.analyticsvidhya.com/blog/2019/02/building-crowd-counting-model-python/>

## LAMPIRAN A

### A. Program GUI Transmitter

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace ComPort
{
    public partial class Form1 : Form
    {
        string dataOUT;
        string dataIN;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender,
EventArgs e)
        {
            string[] ports =
SerialPort.GetPortNames();
            cBocCOMPORT.Items.AddRange(ports);

            //

            chBoxAddToOldData.Checked = true;
        }
    }
}
```

```

        chBoxAlwaysUpdate.Checked = false;
    }

    private void btnOpen_Click(object
sender, EventArgs e)
    {
        try
        {
            serialPort1.PortName =
cBocCOMPORT.Text;
            serialPort1.BaudRate =
Convert.ToInt32(cBoxBaudRate.Text);
            serialPort1.DataBits =
Convert.ToInt32(cBoxDataBits.Text);
            serialPort1.StopBits =
(StopBits)Enum.Parse(typeof(StopBits),
cBoxStopBits.Text);
            serialPort1.Parity =
(Parity)Enum.Parse(typeof(Parity),
cBoxParityBits.Text);

            serialPort1.Open();
            progressBar1.Value = 100;
        }

        catch (Exception err)
        {
            MessageBox.Show(err.Message, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void btnClose_Click(object
sender, EventArgs e)

```

```

        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Close();
                progressBar1.Value = 0;
            }
        }

        private void btnSendData_Click(object
sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                dataOUT = tBoxDataOut.Text;

//serialPort1.WriteLine(dataOUT);          // Ada
Enternya
                serialPort1.Write(dataOUT);
// Gk Ada Enternya
            }
        }

        private void
btnClearDataOUT_Click(object sender, EventArgs
e)
        {
            if(tBoxDataOut.Text != "")
            {
                tBoxDataOut.Text = "";
            }
        }

        private void
serialPort1_DataReceived(object sender,
SerialDataReceivedEventArgs e)

```

```

        {
            dataIN = serialPort1.ReadExisting();
            this.Invoke(new
EventHandler(ShowData));
        }

        private void ShowData(object sender,
EventArgs e)
        {
            if (chBoxAlwaysUpdate.Checked)
            {
                tBoxDataIN.Text = dataIN;
            }
            else if (chBoxAddToOldData.Checked)
            {
                tBoxDataIN.Text += dataIN;
            }
        }

        private void
chBoxAlwaysUpdate_CheckedChanged(object sender,
EventArgs e)
        {
            if (chBoxAlwaysUpdate.Checked)
            {
                chBoxAlwaysUpdate.Checked =
true;
                chBoxAddToOldData.Checked =
false;
            }
            else { chBoxAddToOldData.Checked =
true; }
        }

```

```

        private void
chBoxAddToOldData_CheckedChanged(object sender,
EventArgs e)
    {
        if (chBoxAddToOldData.Checked)
        {
            chBoxAlwaysUpdate.Checked =
false;
            chBoxAddToOldData.Checked =
true;
        }
        else { chBoxAlwaysUpdate.Checked =
true; }
    }

    private void btnClearDataIN_Click(object
sender, EventArgs e)
    {
        if (tBoxDataIN.Text != "")
        {
            tBoxDataIN.Text = "";
        }
    }

    private void button1_Click(object
sender, EventArgs e)
    {
        string test = "test";
        serialPort1.Write(test);
    }

    private void button2_Click(object
sender, EventArgs e)
    {
        string CameraComplete = "go";
        serialPort1.Write(CameraComplete);
    }

```

```

    }

    private void btnRGB_NoIR_Click(object
sender, EventArgs e)
    {
        string RGB_NoIR = "2cam";
        serialPort1.Write(RGB_NoIR);
    }

    private void btnNoir_Click(object
sender, EventArgs e)
    {
        string NoIR = "noir";
        serialPort1.Write(NoIR);
    }

    private void btnNoIR_P_Click(object
sender, EventArgs e)
    {
        string NoIR_Preview = "noir-
preview";
        serialPort1.Write(NoIR_Preview);
    }

    private void btnRGB_Click(object sender,
EventArgs e)
    {
        string RGB = "rgb";
        serialPort1.Write(RGB);
    }

    private void btnRB_P_Click(object
sender, EventArgs e)
    {
        string RGB_Preview = "rgb-preview";
        serialPort1.Write(RGB_Preview);
    }

```

```

    }

    private void
btnSimulationMotor_Click(object sender,
EventArgs e)
    {
        string MotorComplete = "motor";
        serialPort1.Write(MotorComplete);
    }

    private void btnRight_Click(object
sender, EventArgs e)
    {
        string Right = "r";
        serialPort1.Write(Right);
    }

    private void btnLeft_Click(object
sender, EventArgs e)
    {
        string Left = "l";
        serialPort1.Write(Left);
    }

    private void
btnRightCalibration_Click(object sender,
EventArgs e)
    {
        string RightC = "kalibrasiR";
        serialPort1.Write(RightC);
    }

    private void
btnLeftCalibration_Click(object sender,
EventArgs e)
    {

```



```

        string LeftC = "kalibrasiL";
        serialPort1.Write(LeftC);
    }

    private void btnShutDown_Click(object
sender, EventArgs e)
    {
        string Exit = "exit";
        serialPort1.Write(Exit);
    }

    private void btnRestart_Click(object
sender, EventArgs e)
    {
        string Restart = "restart";
        serialPort1.Write(Restart);
    }

    private void
btnGetPictureCompleteV2_Click(object sender,
EventArgs e)
    {
        string CameraCompleteV2 = "gov2";
        serialPort1.Write(CameraCompleteV2);
    }
}
}

```

## **B. Program UART Communication Camera Spectral**

```

#!/usr/bin/env python
import time
import serial
import os

ser = serial.Serial(
    port='/dev/ttyUSB0',      # Di Atas Kanan
    #port='/dev/ttyUSB1',    #Di Bawah Kanan

```

```

baudrate = 57600,
parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE,
bytesize=serial.EIGHTBITS,
timeout=1
)

def main():
    cd = "/home/pi/TA/[PA_2]"
    os.chdir(cd)
    sudocamera = "python3 [PROGRAM_RGB-
NoIR_MotorStepper4x].py"
    os.system(sudocamera)
def mainKanan():
    cd = "/home/pi/TA/UART Communication"
    os.chdir(cd)
    sudocameraKanan = "python3 [IF-
FULL_MotorKanan].py"
    os.system(sudocameraKanan)
def mainKiri():
    cd = "/home/pi/TA/UART Communication"
    os.chdir(cd)
    sudocameraKiri = "python3 [IF-
FULL_MotorKiri].py"
    os.system(sudocameraKiri)
def motorFinal():
    cd = "/home/pi/TA/[PA_2]"
    os.chdir(cd)
    sudomotorFinal = "python3
Motor_Stepper_Final.py"
    os.system(sudomotorFinal)
def camera2():
    cd = "/home/pi/TA/[PA_2]"
    os.chdir(cd)
    sudocamera2 = "python3 Camera_RGB-NoIR.py"
    os.system(sudocamera2)

counter = 1
test = 1

```

```

camera = 1

ser.write(b'Camera Ready! ')
while 1:
    x=ser.readline().decode('ascii')
    if x == 'go':
        ser.write(b'-Received %d-'%(counter))
        main()
        ser.write(b'-FINISH %d- '%(counter))
        counter += 1
    elif x == 'cam':
        ser.write(b'-dapat cam %d-'%(camera))
        camera2()
        ser.write(b'-selesai cam %d- '%(camera))
        camera += 1
    elif x == 'test':
        ser.write(b'-test diterima %d-' %(test))
        test += 1
    elif x == 'motor':
        motorFinal()
        ser.write(b'-motor ok %d-' %(test))
        test += 1
    elif x == 'kanan':
        ser.write(b'-Kanan Received%d-'%(counter))
        mainKanan()
        ser.write(b'-Kanan FINISH%d- '%(counter))
        counter += 1
    elif x == 'kiri':
        ser.write(b'-Kiri Received%d-'%(counter))
        mainKiri()
        ser.write(b'-Kiri FINISH%d- '%(counter))
        counter += 1
    elif x == 'exit':
        ser.write(b' -EXIT- ')
        os.system("sudo shutdown -h now")

print (x)

```

### **C. Program Get Image Camera Spectral**

```
import RPi.GPIO as gp
```

```

from datetime import datetime as dtm
from picamera import PiCamera
import time
import os

gp.setwarnings(False)
gp.setmode(gp.BOARD)

gp.setup(7, gp.OUT)
gp.setup(11, gp.OUT)
gp.setup(12, gp.OUT)

control_pins = [13,15,16,22]
for pin in control_pins:
    gp.setup(pin, gp.OUT)
    gp.output(pin, 0)
halfstep_seq = [
    [1,0,0,0],
    [1,1,0,0],
    [0,1,0,0],
    [0,1,1,0],
    [0,0,1,0],
    [0,0,1,1],
    [0,0,0,1],
    [1,0,0,1]
]
halfstep_seq_REVERSE = [
    [0,0,0,1],
    [0,0,1,1],
    [0,0,1,0],
    [0,1,1,0],
    [0,1,0,0],
    [1,1,0,0],
    [1,0,0,0],
    [1,0,0,1]
]

gp.output(11, True)
gp.output(12, True)

```

```

saat_ini = dtm.now() #tgl dan jam saat ini
now = dtm.strftime(saat_ini, '%d-%b-
%Y_%H:%M:%S') # tpye = string

camera = PiCamera(resolution = (3280, 2464))

#camera = PiCamera(resolution = (3280, 2464))
#camera.rotation = 180

def main():

    print("Start testing the camera A RGB ",
now)
    i2c = "i2cset -y 1 0x70 0x00 0x04"
    os.system(i2c)
    gp.output(7, False)
    gp.output(11, False)
    gp.output(12, True)
    captureRGB(1)          # RGB

    print("Start testing the camera C NoIR
Band2")
    i2c = "i2cset -y 1 0x70 0x00 0x06"
    os.system(i2c)
    gp.output(7, False)
    gp.output(11, True)
    gp.output(12, False)
    captureNoIR(2)
    for i in range(128):          # 90 Derajat
Kanan
        for halfstep in range(8):
            for pin in range(4):
                gp.output(control_pins[pin],
halfstep_seq_REVERSE[halfstep][pin])
                time.sleep(0.001)
            print('Putaran 1')

```

```

    print("Start testing the camera C NoIR
Band3", now)
    captureNoIR(3)      # BAND 3
    for i in range(256):      # 180 Derajat
Kiri
        for halfstep in range(8):
            for pin in range(4):
                gp.output(control_pins[pin],
halfstep_seq[halfstep][pin])
                time.sleep(0.001)
            print('Putaran 2')

    print("Start testing the camera C NoIR
Band1", now)
    captureNoIR(1)      # BAND 1
    for i in range(128):      # 90 Derajat
Kanan
        for halfstep in range(8):
            for pin in range(4):
                gp.output(control_pins[pin],
halfstep_seq_REVERSE[halfstep][pin])
                time.sleep(0.001)
            print('Putaran 3')

def captureNoIR(cam):
    camera.rotation = 180
    camera.start_preview(resolution=(1920,1080))
    time.sleep(3.5)      # Waktu
Pengambilan Gambar
    if cam == 1:

camera.capture("/home/pi/TA/[PA_2]/[Hasil_Gambar
]/NoIR_Band1_%.jpg" % now)
        elif cam == 2:

camera.capture("/home/pi/TA/[PA_2]/[Hasil_Gambar
]/NoIR_Band2_%.jpg" % now)
        else:

```

```

camera.capture("/home/pi/TA/[PA_2]/[Hasil_Gambar
]/NoIR_Band3_%s.jpg" % now)
    camera.stop_preview()

def captureRGB(cam):
    camera.rotation = 180
    camera.start_preview(resolution=(1920,1080))
    time.sleep(3.5)                # Waktu
    Pengambilan Gambar

camera.capture("/home/pi/TA/[PA_2]/[Hasil_Gambar
]/RGB_%s.jpg" % now)
    camera.stop_preview()

if __name__ == "__main__":
    main()

    gp.output(7, False)
    gp.output(11, False)
    gp.output(12, True)
    gp.cleanup()
    os._exit(0)

```

#### **D. Program Rotate Motor Stepper**

```

import RPi.GPIO as gp
import time
import os

gp.setwarnings(False)
gp.setmode(gp.BOARD)

control_pins = [13,15,16,22]
for pin in control_pins:
    gp.setup(pin, gp.OUT)

```

```

    gp.output(pin, 0)
halfstep_seq = [
    [1,0,0,0],
    [1,1,0,0],
    [0,1,0,0],
    [0,1,1,0],
    [0,0,1,0],
    [0,0,1,1],
    [0,0,0,1],
    [1,0,0,1]
]
halfstep_seq_REVERSE = [
    [0,0,0,1],
    [0,0,1,1],
    [0,0,1,0],
    [0,1,1,0],
    [0,1,0,0],
    [1,1,0,0],
    [1,0,0,0],
    [1,0,0,1]
]

for i in range(128):                # 90 Derajat
Kanan
    for halfstep in range(8):
        for pin in range(4):
            gp.output(control_pins[pin],
halfstep_seq_REVERSE[halfstep][pin])
            time.sleep(0.001)
print('Putaran 1')
time.sleep(0.5)

for i in range(256):                # 180 Derajat
Kiri
    for halfstep in range(8):
        for pin in range(4):
            gp.output(control_pins[pin],
halfstep_seq[halfstep][pin])
            time.sleep(0.001)

```



```
print('Putaran 2')
time.sleep(0.5)

for i in range(128):          # 90 Derajat
    Kanan
    for halfstep in range(8):
        for pin in range(4):
            gp.output(control_pins[pin],
halfstep_seq_REVERSE[halfstep][pin])
            time.sleep(0.001)
print('Putaran 3')
time.sleep(0.5)

gp.cleanup()
os._exit(0)
```

## E. Program Preprocessing Image Processing CSRNet (Jupyter Notebook)

```
In [ ]: import h5py
import scipy.io as io
import scipy.spatial
import PIL.Image as Image
import numpy as np
import os
import glob
from matplotlib import pyplot as plt
from scipy.ndimage.filters import gaussian_filter
import scipy
import json
from matplotlib import cm as CM
from image import *
from model import CSRNet
import torch
%matplotlib inline
```

```
In [ ]: #this is borrowed from https://github.com/davideverona/deep-crowd-counting_crowdnet
def gaussian_filter_density(gt):
    print (gt.shape)
    density = np.zeros(gt.shape, dtype=np.float32)
    gt_count = np.count_nonzero(gt)
    if gt_count == 0:
        return density

    pts = np.array(list(zip(np.nonzero(gt)[1], np.nonzero(gt)[0])))
    leafsize = 2048
    # build kdtree
    tree = scipy.spatial.KDTree(pts.copy(), leafsize=leafsize)
    # query kdtree
    distances, locations = tree.query(pts, k=4)

    print ('generate density...')
    for i, pt in enumerate(pts):
        pt2d = np.zeros(gt.shape, dtype=np.float32)
        pt2d[pt[1],pt[0]] = 1.
        if gt_count > 1:
            sigma = (distances[1][1]+distances[1][2]+distances[1][3])*0.1
        else:
            sigma = np.average(np.array(gt.shape))/2./2. #case: 1 point
        density += scipy.ndimage.filters.gaussian_filter(pt2d, sigma, mode='constant')
    print ('done.')
    return density
```

```

In [ ]: #set the root to the Shanghai dataset you download
# root = r'C:\Users\owner\CSRNet_Crowd_Detection_PA2\Dataset_Shanghai'
root = r'C:\Users\owner\CSRNet_Crowd_Detection_PA2\Dataset_Hanif'

In [ ]: #now generate the ShanghaiA's ground truth

# part_A_train = os.path.join(root, 'part_A_final/train_data', 'images')
part_A_train = os.path.join(root, 'part_A_final/train_data', 'images_test')

part_A_test = os.path.join(root, 'part_A_final/test_data', 'images')
# part_A_test = os.path.join(root, 'part_A_final/test_data', 'images_test')

part_B_train = os.path.join(root, 'part_B_final/train_data', 'images')
part_B_test = os.path.join(root, 'part_B_final/test_data', 'images')

path_sets = [part_A_train, part_A_test]

In [ ]: img_paths = []
for path in path_sets:
    for img_path in glob.glob(os.path.join(path, '*.jpg')):
        img_paths.append(img_path)

In [ ]: for img_path in img_paths:
    print (img_path)

    mat = io.loadmat(img_path.replace('.jpg', '.mat').replace('images_test', 'ground_truth_test').replace('IMG_', 'GT_IMG_'))
    # mat = io.loadmat(img_path.replace('.jpg', '.mat').replace('images', 'ground_truth').replace('IMG_', 'GT_IMG_'))

    img= plt.imread(img_path)
    k = np.zeros((img.shape[0],img.shape[1]))
    print(mat)

    # gt = mat["image_info"][0,0][0,0][0]
    gt = mat["image_info"][0,0][0]

    print()
    for i in range(0,len(gt)):
        if int(gt[i][1])<img.shape[0] and int(gt[i][0])<img.shape[1]:
            k[int(gt[i][1]),int(gt[i][0])]=1
    k = gaussian_filter_density(k)
    # with h5py.File(img_path.replace('.jpg', '.h5').replace('images', 'ground_truth_test'), 'w') as hf:
    with h5py.File(img_path.replace('.jpg', '.h5').replace('images', 'ground_truth'), 'w') as hf:
        hf['density'] = k

In [ ]: #now see a sample from ShanghaiA
plt.imshow(Image.open(img_paths[3]))

In [ ]: gt_file = h5py.File(img_paths[0].replace('.jpg', '.h5').replace('images', 'ground_truth'), 'r')
groundtruth = np.asarray(gt_file['density'])
plt.imshow(groundtruth, cmap=CM.jet)

In [ ]: np.sum(groundtruth)# don't mind this slight variation

```

```

In [ ]: np.sum(groundtruth)# don't mind this slight variation

In [ ]: #now generate the ShanghaiB's ground truth
path_sets = [part_0_train,part_0_test]

In [ ]: img_paths = []
for path in path_sets:
    for img_path in glob.glob(os.path.join(path, '*.jpg')):
        img_paths.append(img_path)

In [ ]: for img_path in img_paths:
    print (img_path)
    mat = io.loadmat(img_path.replace('.jpg','.mat').replace('images','ground_truth').replace('IMG_', 'GT_IMG_'))
    img= plt.imread(img_path)
    k = np.zeros((img.shape[0],img.shape[1]))
    gt = mat["image_info"][0,0][0]
    for i in range(0,len(gt)):
        if int(gt[i][1])<img.shape[0] and int(gt[i][0])<img.shape[1]:
            k[int(gt[i][1]),int(gt[i][0])]-1
    k = gaussian_filter(k,15)
    with h5py.File(img_path.replace('.jpg','.h5').replace('images','ground_truth'), 'w') as hf:
        hf['density'] = k

In [ ]: #now see a sample from ShanghaiB
plt.imshow(Image.open(img_paths[0]))

In [ ]: gt_file = h5py.File(img_paths[0].replace('.jpg','.h5').replace('images','ground_truth'),'r')
groundtruth = np.asarray(gt_file['density'])
plt.imshow(groundtruth,cmap=CM.jet)

In [ ]: np.sum(groundtruth)# don't mind this slight variation

```

## F. Program Modelling Image Processing CSRNet (Jupyter Notebook)

```
In [ ]: from keras.layers.normalization import BatchNormalization
        from keras.preprocessing.image import load_img, img_to_array
        from sklearn.metrics import mean_squared_error
        from keras.initializers import RandomNormal
        from keras.applications.vgg16 import VGG16
        from keras.optimizers import SGD
        from keras.models import Model, Sequential
        from keras.layers import *
        from keras import backend as K
        from keras.models import model_from_json
        from matplotlib import cm as CM
        import matplotlib.pyplot as plt
        import tensorflow as tf
        from tqdm import tqdm
        import scipy.io as io
        from PIL import Image
        import PIL
        import h5py
        import os
        import glob
        import cv2
        import random
        import math
        import sys

        import numpy as np
```

```
In [ ]: K.clear_session()
```

```
root = r'C:\Users\owner\CSRNet_Crowd_Detection_PA2\Dataset_Shanghai'
# root = r'C:\Users\owner\CSRNet_Crowd_Detection_PA2\Dataset_Hanif'
```

```

In [ ]: part_A_train = os.path.join(root, 'part_A_final/train_data', 'images')
        part_A_test = os.path.join(root, 'part_A_final/test_data', 'images')

        part_B_train = os.path.join(root, 'part_B_final/train_data', 'images')
        part_B_test = os.path.join(root, 'part_B_final/test_data', 'images')

        temp = ''
        #temp = 'test_images'
        path_sets = [part_A_train]

In [ ]: img_paths = []

        for path in path_sets:

            for img_path in glob.glob(os.path.join(path, '*.jpg')):

                img_paths.append(str(img_path))

        print("Total images : ", len(img_paths))    # Images of train_data (part A)

In [ ]: def create_img(path):
        #Function to load, normalize and return image
        im = Image.open(path).convert('RGB')

        im = np.array(im)

        im = im/255.0

        im[:, :, 0] = (im[:, :, 0] - 0.485) / 0.229
        im[:, :, 1] = (im[:, :, 1] - 0.456) / 0.224
        im[:, :, 2] = (im[:, :, 2] - 0.406) / 0.225

        #print(im.shape)
        #im = np.expand_dims(im, axis = 0)
        return im

```

```

def get_input(path):
    path = path[0]
    img = create_img(path)
    return(img)

def get_output(path):
    #import target
    #resize target

    gt_file = h5py.File(path, 'r')

    target = np.asarray(gt_file['density'])

    img = cv2.resize(target, (int(target.shape[1]/8), int(target.shape[0]/8)), interpolation = cv2.INTER_CUBIC)*64

    img = np.expand_dims(img, axis = 2)

    #print(img.shape)

    return img

def preprocess_input(image, target):
    #crop image
    #crop target
    #resize target
    crop_size = (int(image.shape[0]/2), int(image.shape[1]/2))

    if random.randint(0,9) <= -1:
        dx = int(random.randint(0,1)*image.shape[0]*1./2)
        dy = int(random.randint(0,1)*image.shape[1]*1./2)

```

```

def preprocess_input(image,target):
    #crop image
    #crop target
    #resize target
    crop_size = (int(image.shape[0]/2),int(image.shape[1]/2))

    if random.randint(0,9)<= -1:
        dx = int(random.randint(0,1)*image.shape[0]*1./2)
        dy = int(random.randint(0,1)*image.shape[1]*1./2)
    else:
        dx = int(random.random()*image.shape[0]*1./2)
        dy = int(random.random()*image.shape[1]*1./2)

    #print(crop_size , dx , dy)
    img = image[dx : crop_size[0]+dx , dy:crop_size[1]+dy]

    target_aug = target[dx:crop_size[0]+dx,dy:crop_size[1]+dy]
    #print(img.shape)

    return(img,target_aug)

```

```

: #Image data generator
def image_generator(files, batch_size = 64):

    while True:

        input_path = np.random.choice(a = files, size = batch_size)

        batch_input = []
        batch_output = []

        #for input_path in batch_paths:

```



```

        #for input_path in batch_paths:

        inputt = get_input(input_path )
        output = get_output(input_path[0].replace('.jpg','.h5').replace('images','ground_truth'))

        batch_input += [inputt]
        batch_output += [output]

        batch_x = np.array( batch_input )
        batch_y = np.array( batch_output )

        yield( batch_x, batch_y )

]: def save_mod(model , str1 , str2):
    model.save_weights(str1)

    model_json = model.to_json()

    with open(str2, "w") as json_file:
        json_file.write(model_json)

]: def init_weights_vgg(model):
    #vgg = VGG16(weights='imagenet', include_top=False)

    json_file = open('models/VGG_16.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
    loaded_model.load_weights("weights/VGG_16.h5")

    vgg = loaded_model

```

```

vgg = loaded_model

vgg_weights=[]
for layer in vgg.layers:
    if('conv' in layer.name):
        vgg_weights.append(layer.get_weights())

offset=0
i=0
while(i<10):
    if('conv' in model.layers[i+offset].name):
        model.layers[i+offset].set_weights(vgg_weights[i])
        i=i+1
        #print('h')
    else:
        offset=offset+1

return (model)

```

```

]: def euclidean_distance_loss(y_true, y_pred):
    # Euclidean distance as a measure of loss (Loss function)
    return K.sqrt(K.sum(K.square(y_pred - y_true), axis=-1))

```

```

]: # Neural network model : VGG + Conv
def CrowdNet():
    #Variable Input Size
    rows = None
    cols = None

    #Batch Normalisation option

```

```

def CrowdNet():
    #Variable Input Size
    rows = None
    cols = None

    #Batch Normalisation option

    batch_norm = 0
    kernel = (3, 3)
    init = RandomNormal(stddev=0.01)
    model = Sequential()

    #custom VGG:

    if (batch_norm):
        model.add(Conv2D(64, kernel_size = kernel, input_shape = (rows,cols,3),activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(64, kernel_size = kernel,activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(128, kernel_size = kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(128, kernel_size = kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(256, kernel_size = kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(256, kernel_size = kernel, activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(512, kernel_size = kernel,activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(512, kernel_size = kernel,activation = 'relu', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(512, kernel_size = kernel,activation = 'relu', padding='same'))
        model.add(BatchNormalization())

    else:
        model.add(Conv2D(64, kernel_size = kernel,activation = 'relu', padding='same',input_shape = (rows, cols, 3), kernel_initializer = init))
        model.add(Conv2D(64, kernel_size = kernel,activation = 'relu', padding='same', kernel_initializer = init))
        model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(128, kernel_size = kernel, activation = 'relu', padding='same', kernel_initializer = init))
        model.add(Conv2D(128, kernel_size = kernel, activation = 'relu', padding='same', kernel_initializer = init))
        model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(256, kernel_size = kernel, activation = 'relu', padding='same', kernel_initializer = init))
        model.add(Conv2D(256, kernel_size = kernel, activation = 'relu', padding='same', kernel_initializer = init))
        model.add(MaxPooling2D(strides=2))
        model.add(Conv2D(512, kernel_size = kernel,activation = 'relu', padding='same', kernel_initializer = init))
        model.add(Conv2D(512, kernel_size = kernel,activation = 'relu', padding='same', kernel_initializer = init))
        model.add(Conv2D(512, kernel_size = kernel,activation = 'relu', padding='same', kernel_initializer = init))

    #Conv2D
    model.add(Conv2D(512, (3, 3), activation='relu', dilation_rate = 2, kernel_initializer = init, padding = 'same'))
    model.add(Conv2D(512, (3, 3), activation='relu', dilation_rate = 2, kernel_initializer = init, padding = 'same'))
    model.add(Conv2D(512, (3, 3), activation='relu', dilation_rate = 2, kernel_initializer = init, padding = 'same'))
    model.add(Conv2D(256, (3, 3), activation='relu', dilation_rate = 2, kernel_initializer = init, padding = 'same'))
    model.add(Conv2D(128, (3, 3), activation='relu', dilation_rate = 2, kernel_initializer = init, padding = 'same'))
    model.add(Conv2D(64, (3, 3), activation='relu', dilation_rate = 2, kernel_initializer = init, padding = 'same'))
    model.add(Conv2D(1, (1, 1), activation='relu', dilation_rate = 1, kernel_initializer = init, padding = 'same'))

    sgd = SGD(lr = 1e-7, decay = (5*1e-4), momentum = 0.95)
    model.compile(optimizer=sgd, loss=euclidean_distance_loss, metrics=['mse'])

    model = init_weights_vgg(model)

    return model

```

```
] : model = CrowdNet()

] : model.summary()

] : train_gen = image_generator(img_paths,1)

] : sgd = SGD(lr = 1e-7, decay = (5*1e-4), momentum = 0.95)
    model.compile(optimizer=sgd, loss=euclidean_distance_loss, metrics=['mse'])

] : model.fit_generator(train_gen,epochs=1,steps_per_epoch= 700 , verbose=1) #700

] : #save_mod(model,"weights/model_A_weightsV2.h5","models/ModelV2.json")
    save_mod(model,"weights/model_A_weights.h5","models/Model.json")
```

## G. Program Inference Image Processing CSRNet (Jupyter Notebook)

```
In [ ]: import cv2
import h5py
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm as d
from keras.models import model_from_json

In [ ]: def load_model():
    # Function to Load and return neural network model

    # json_file = open('models/ModelDiaoXY.json', 'r')
    json_file = open('models/Model.json', 'r')

    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)

    loaded_model.load_weights("weights/model_A_weights.h5")
    # Loaded_model.Load_weights("weights/model_A_weightsDiaoXY.h5")

    return loaded_model

def create_img(path):
    #Function to Load,normalize and return image
    print(path)
    im = Image.open(path).convert('RGB')

    im = np.array(im)

    im = im/255.0

    im[:, :, 0] = (im[:, :, 0] - 0.485) / 0.229
    im[:, :, 1] = (im[:, :, 1] - 0.456) / 0.224
    im[:, :, 2] = (im[:, :, 2] - 0.406) / 0.225

    im = np.expand_dims(im, axis = 0)
```

```

In [ ]: def predict(path):
        #Function to load image,predict heat map, generate count and return (count , image , heat map)
        model = load_model()
        image = create_img(path)
        ans = model.predict(image)
        count = np.sum(ans)
        return count,image,ans

In [ ]: # ans,img,hmap = predict(r"C:\Users\owner\CSRNNet_Crowd_Detection_PA2\Dataset_Hanif\part_A_final\train_data\images\IMG_33.jpg")
ans,img,hmap = predict(r"C:\Users\owner\CSRNNet_Crowd_Detection_PA2\Dataset_Hanif\part_A_final\test_data\images\4\IMG_2.jpg")

# # ans,img,hmap = predict(r"C:\Users\owner\CSRNNet_Crowd_Detection_PA2\Dataset_Shanghai\part_A_final\test_data\images\IMG_5.jpg")
#

In [ ]: np.sum(ans)
#print(predict(r"C:\Users\owner\Dataset_Shanghai_Keras\part_A_final\test_data\images\IMG_170.jpg"))
#print count, image, heat map

print(ans)
plt.imshow(img.reshape(img.shape[1],img.shape[2],img.shape[3]))
plt.show()
plt.imshow(hmap.reshape(hmap.shape[1],hmap.shape[2]), cmap = c.jet )
plt.show()

In [ ]: # temp = predict(r"C:\Users\owner\CSRNNet_Crowd_Detection_PA2\Dataset_Hanif\part_A_final\test_data\images\IMG_5.jpg")
temp = predict(r"C:\Users\owner\CSRNNet_Crowd_Detection_PA2\Dataset_Hanif\part_A_final\test_data\images\4\IMG_5.jpg")

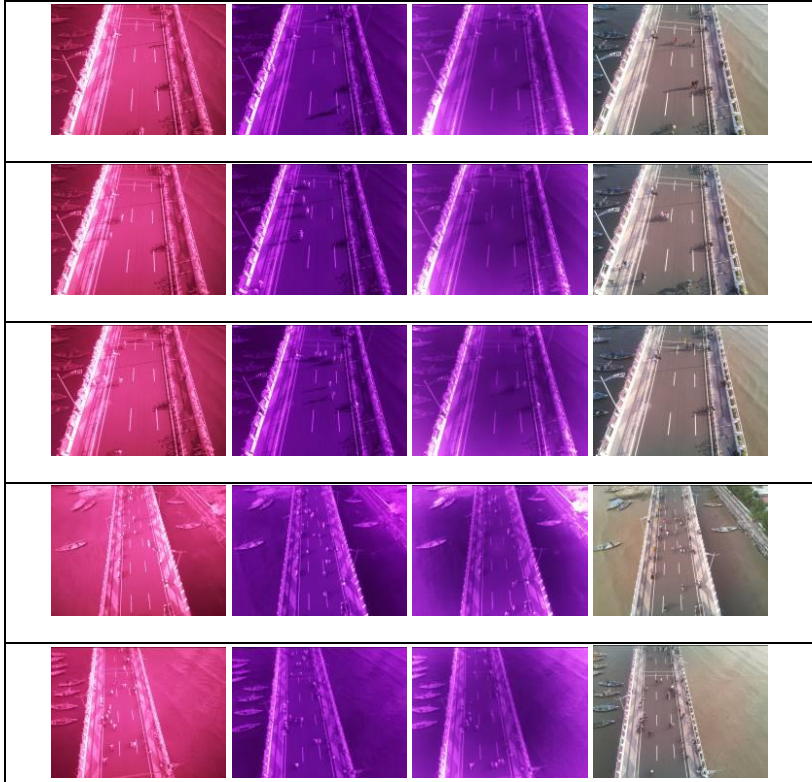
# temp = h5py.File(r"C:\Users\owner\CSRNNet_Crowd_Detection_PA2\Dataset_Shanghai\part_A_final\test_data\ground_truth\IMG_5.h5")
temp_1 = np.asarray(temp['density'])
plt.imshow(temp_1,cmap = c.jet)
print("Original Count : ",int(np.sum(temp_1)) + 1)

```

## LAMPIRAN B

### Hasil Gambar Camera Spectral

#### Jembatan Suroboyo



**Gambar** Spectral NoIR Band1 – Band2 – Band3 – RGB

## Pasar Pacar Keling



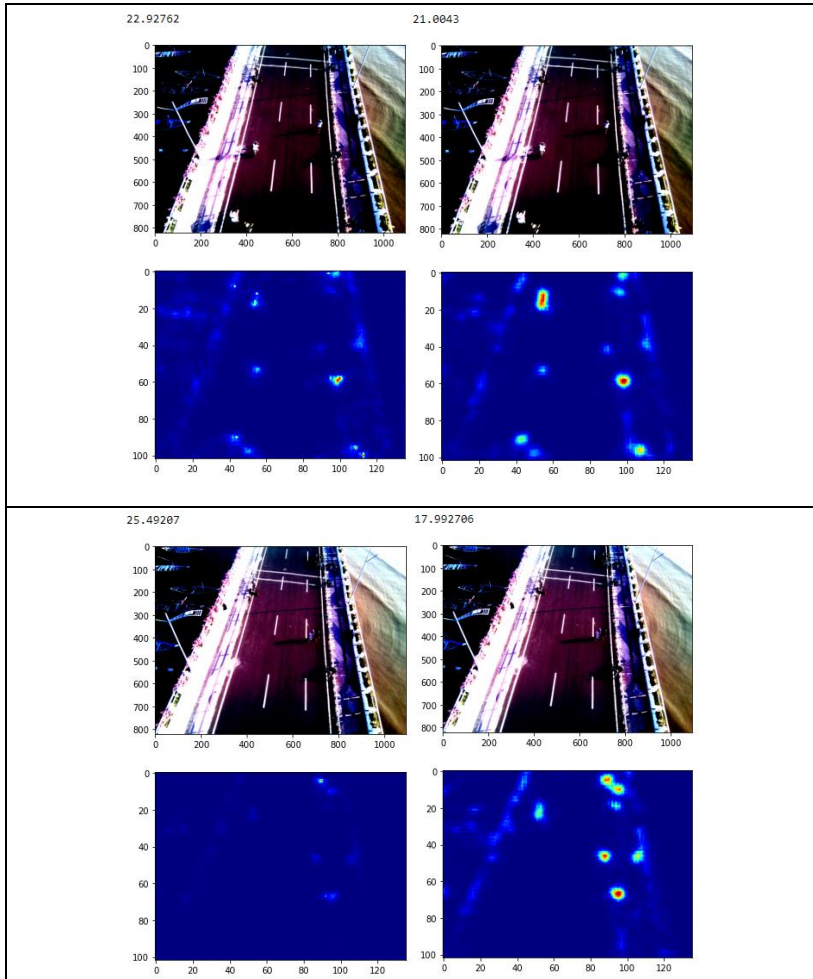
**Gambar** Spectral NoIR Band2 – RGB



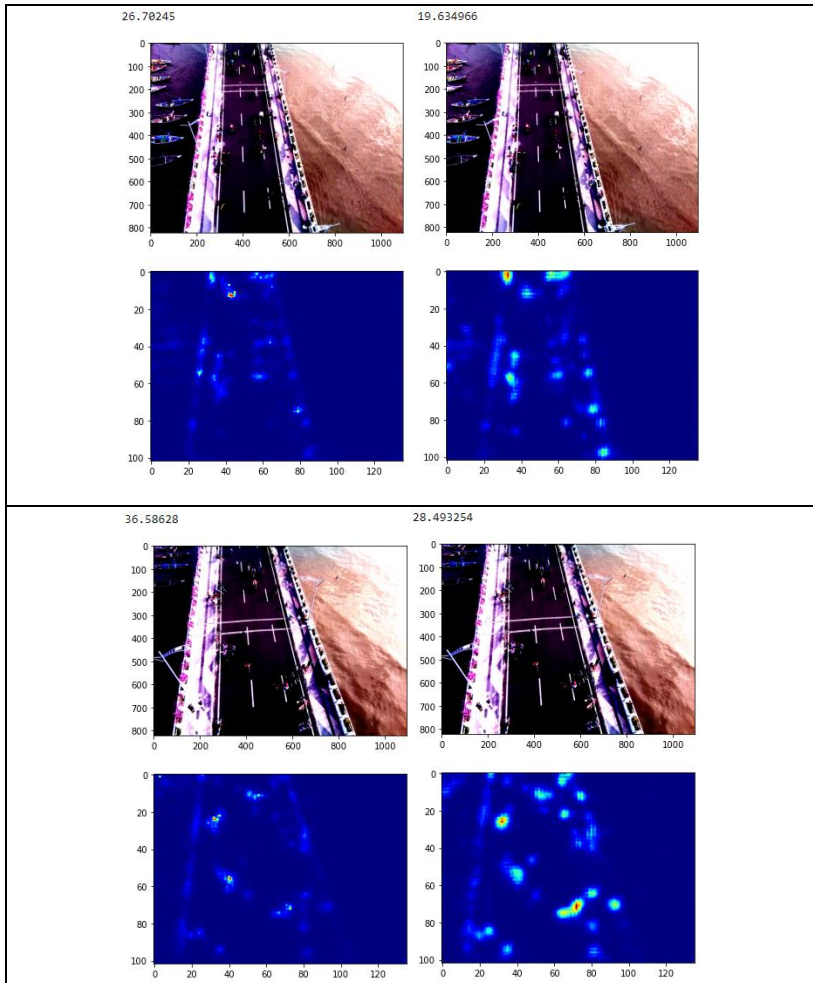
# Lampiran C

## Hasil Gambar Peta Kerapatan

### Jembatan Suroboyo

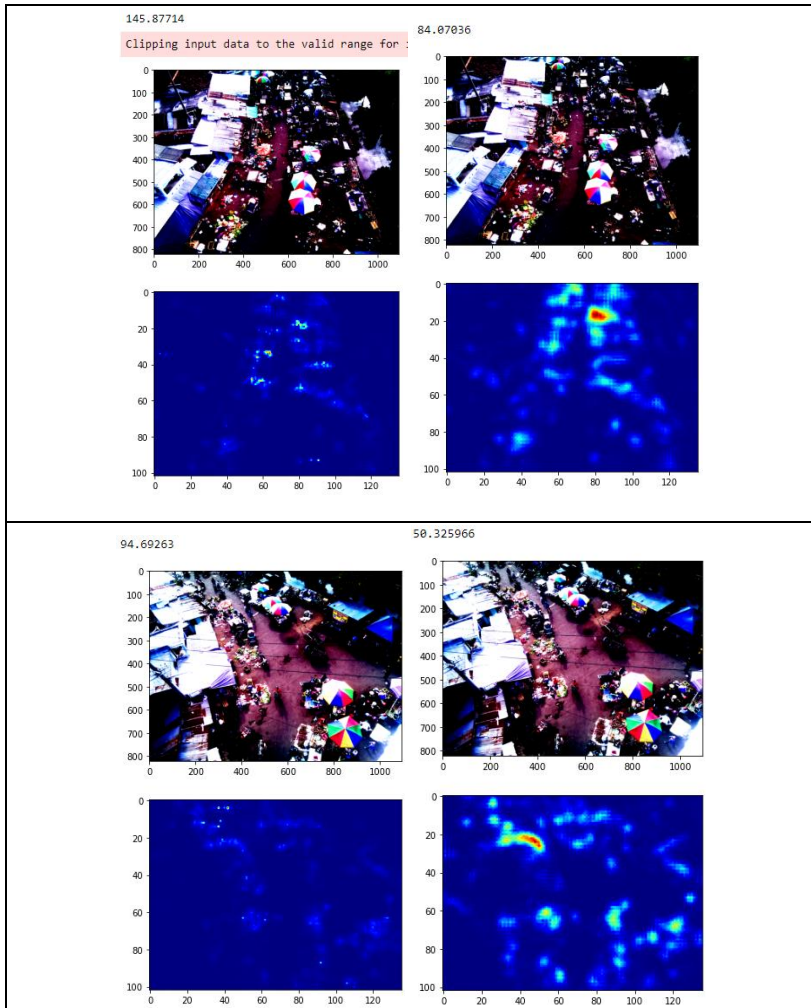


**Gambar** Peta kerapatan Model A (kiri) dan Model B (kanan)



**Gambar** Peta kerapatan Model A (kiri) dan Model B (kanan)

## Pasar Pacar Keling



**Gambar** Peta kerapatan Model A (kiri) dan Model B (kanan)

## BIODATA PENULIS



Nama : Hanif Izzudin Rahman  
Tempat/Tanggal Lahir : Malang / 15 Mei 1998  
Alamat : Jl. Klampis Harapan 3 No. 2A  
Telepon/Hp : 085720176894  
Hobi : Olahraga, nonton film, dengerin musik  
Motto : Keep progress and working habit !!!

### **Riwayat Pendidikan :**

- SD Islam Terpadu Fitrah Insani Tahun 2005 – 2010
- SMP Negeri 1 Cimahi Tahun 2010 – 2013
- SMA Negeri 2 Cimahi Tahun 2013 – 2016
- Politeknik Elektronika Negeri Surabaya Tahun 2016 – 2020