# *PEOPLE COUNTING* DAN PENGUKURAN JARAK UNTUK *INDOOR MONITORING* BERBASIS PEMROSESAN CITRA

**SKRIPSI**

untuk memenuhi salah satu persyaratan
mencapai derajat Sarjana S1


**Disusun oleh:**

**Luthfi Radifan Ihtisyamuddin**

**16524134**

**Jurusan Teknik Elektro**

**Fakultas Teknologi Industri**

**Universitas Islam Indonesia**

**Yogyakarta**


**2020**

# LEMBAR PENGESAHAN

## *PEOPLE COUNTING* DAN PENGUKURAN JARAK UNTUK *INDOOR MONITORING* BERBASIS PEMROSESAN CITRA

**TUGAS AKHIR**

**Diajukan sebagai Salah Satu Syarat untuk Memperoleh**
**Gelar Sarjana Teknik**
**pada Program Studi Teknik Elektro**
**Fakultas Teknologi Industri**
**Universitas Islam Indonesia**

**Disusun oleh:**

**Luthfi Radifan Ihtisyamuddin**
**16524134**

**Yogyakarta, 27 Agustus 2020**

**Menyetujui,**

**Pembimbing**

**Elvira Sukma Wahyuni, S.Pd.T., M.Eng.**
**155231301**

# LEMBAR PENGESAHAN

## SKRIPSI

## *PEOPLE COUNTING* DAN PENGUKURAN JARAK UNTUK *INDOOR MONITORING* BERBASIS PEMROSESAN CITRA

Dipersiapkan dan disusun oleh:

**Luthfi Radifan Ihtisyamuddin**

**16524134**

**Telah dipertahankan di depan dewan penguji**

**Pada tanggal: 19 Agustus 2020**

**Susunan dewan penguji**

**Ketua Penguji : Elvira Sukma Wahyuni, S.Pd.T., M.Eng.,** _____

**Anggota Penguji 1: Dzata Farahiyah, S.T., M.Sc.,** _____

**Anggota Penguji 2: Dwi Ana Ratna Wati, S.T., M.Eng.,** _____

**Skripsi ini telah diterima sebagai salah satu persyaratan untuk memperoleh gelar Sarjana**

**Tanggal: 31 Agustus 2020**

**Ketua Program Studi Teknik Elektro**

**Yusuf Aziz Amrullah, S.T., M.Eng., Ph.D.**

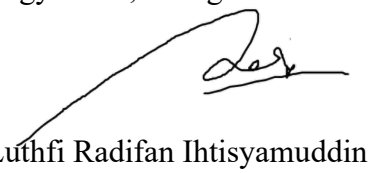
**045240101**

# PERNYATAAN

Dengan ini Saya menyatakan bahwa:

1. Skripsi ini tidak mengandung karya yang diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan Saya juga tidak mengandung karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.
2. Informasi dan materi Skripsi yang terkait hak milik, hak intelektual, dan paten merupakan milik bersama antara tiga pihak yaitu penulis, dosen pembimbing, dan Universitas Islam Indonesia. Dalam hal penggunaan informasi dan materi Skripsi terkait paten maka akan diskusikan lebih lanjut untuk mendapatkan persetujuan dari ketiga pihak tersebut diatas.

Yogyakarta, 27 Agustus 2020



Luthfi Radifan Ihtisyamuddin

# KATA PENGANTAR

Assalamu'alaykum Warahmatullahi Wabarakaatuh

*Alhamdulillahirabbil'alamin*, segala puji syukur pada Allah *subhanahu wa ta'ala* yang telah memberikan rahmat, hidayah, dan karunia-Nya sehingga Skripsi ini dapat terselesaikan dan semoga bermanfaat bagi orang banyak. Shalawat dan salam semoga tercurahkan kepada Nabi besar kita, Nabi Muhammad *shallallahu'alaihi wa sallam* beserta para keluarga, sahabat, dan pengikutnya hingga akhir zaman. Semoga kita selalu mendapatkan syafaat hingga yaumil akhir nanti.

*Alhamdulillahirabbil'alamin*, penulis ucapkan terima kasih karena dapat menyelesaikan Skripsi yang berjudul "*People Counting* dan Pengukuran Jarak untuk *Indoor Monitoring* Berbasis Pemrosesan Citra". Skripsi ini dibuat sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia.

Selama pengerjaan Skripsi ini, penulis telah banyak diberikan bantuan, bimbingan, dukungan, kerjasama, fasilitas, dan kemudahan dari berbagai pihak. Maka pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT atas segala rahmat-Nya yang senantiasa memberikan Kesehatan, kesempatan, serta kemudahan kepada hamba-Nya dalam segala urusan.
2. Nabi Muhammad SAW sebagai suri tauladan bagi seluruh umat muslim di seluruh dunia. Semoga suri tauladan beliau akan selalu jadi pedoman bagi penulis untuk terus memperbaiki diri dalam menjalani kehidupan sebagai seorang muslim.
3. Bapak Budi Cahyono dan Ibu Mei Munah Cahyani selaku orang tua penulis yang selalu memberikan doa, motivasi, dukungan, serta semangat sehingga penulis dapat menyelesaikan skripsi ini.
4. Ibu Elvira Sukma Wahyuni, S.Pd.T., M.Eng. selaku dosen pembimbing Skripsi yang telah meluangkan waktu dan membagi pengetahuan untuk memberikan arahan dan bimbingan hingga terselesaikannya Skripsi ini.
5. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D. selaku Rektor Universitas Islam Indonesia beserta seluruh jajaran pimpinan universitas.

6. Bapak Yusuf Aziz Amrullah, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Teknik Elektro FTI UII beserta seluruh jajaran pengajar dan pengurus Program Studi Teknik Elektro.
7. Ibu Dzata Farahiyah, S.T., M.Sc. selaku dosen penguji 1 yang telah meluangkan waktunya dalam pelaksanaan sidang Skripsi dan saran-saran yang diberikan.
8. Ibu Dwi Ana Ratna Wati, S.T., M.Eng. selaku dosen penguji 2 yang telah meluangkan waktunya dalam pelaksanaan sidang Skripsi dan saran-saran yang diberikan.
9. Sarach Respilia Sukma atas doa, motivasi, dukungan, serta semangat sehingga penulis dapat menyelesaikan skripsi ini.
10. Rekan-rekan KFC, Wawan, Ilham, Panatas, Farrosha atas semangat, dukungan, dan ilmu yang diberikan sehingga penulis dapat menyelesaikan skripsi ini.
11. Rekan-rekan KKN Unit 105 dan Unit 107 telah menjadi keluarga bagi penulis.
12. Rekan-rekan kelas bimbingan Ibu Elvira, Zulfika, Abidafi, Ryan, dan Galang.
13. Rekan-rekan kelas bimbingan Bapak Nandra, Thomas, Anas, Rafi, Aldan, Tias, dan Barry.
14. Seluruh rekan-rekan Teknik Elektro Angkatan 2016 atas pengalaman terbaiknya selama mengenyam pendidikan di kampus Universitas Islam Indonesia.
15. Anas, Andaru, Bagas, dan seluruh rekan-rekan Unisi Robotic atas ilmu dan pengalaman yang diberikan selama ini.
16. Fauzan, Rudyan, Rofiq, Bevrin atas dukungan dan doanya sehingga penulis dapat menyelesaikan skripsi ini.
17. Seluruh keluarga besar penulis yang selalu memberikan doa, motivasi, dukungan, dan semangat sehingga penulis dapat menyelesaikan skripsi ini.
18. Kepada semua pihak yang namanya tidak dapat disebutkan seluruhnya atas doa, motivasi, dukungan, dan semangat kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna, oleh karena itu saran dan kritik sangat diharapkan untuk penyempurnaan penelitian berikutnya. Semoga skripsi ini dapat bermanfaat bagi pembaca dan pihak yang berkepentingan.

Wassalamu’alaykum Warahmatullahi Wabarakaatuh

# ARTI LAMBANG DAN SINGKATAN

| Lambang/singkatan | Keterangan |
|---|---|
| Wi-Fi | *Wireless Fidelity* |
| AP | *Access Point* |
| RSS | *Received Signal Strength* |
| CCTV | *Closed Circuit Television* |
| KLT | Kanade-Lucas-Tomasi |
| OP | *Optical Origin* |
| *Hs* | Jarak *optical origin* dengan sisi depan kamera |
| *href1*, *href2*, *hm1*, *hm2*, *b* | Jarak referensi (cm) |
| *h1*, *h2*, *a* | Jarak terukur (cm) |
| $\Delta h$ | Jarak perpindahan antara dua jarak referensi |
| *D(h1)*, *D(h2)*, *Dm1*, *Dm2* | Lebar jangkauan kamera dalam nilai riil (cm) |
| *L* | Lebar objek dalam nilai riil (cm) |
| *Pa*, *Pb* | Titik tepi kanan dan kiri piksel daripada objek |
| *N(h1)*, *N(h2)* | Lebar objek dalam nilai piksel |
| *Nmax1*, *Nmax2* | Lebar jangkauan kamera dalam nilai piksel |
| *MR(h1)*, *MR(h2)* | Nilai titik tepi piksel kanan |
| *ML(h1)*, *ML(h2)* | Nilai titik tepi piksel kiri |
| RGB | *Red-Green-Blue* |
| ROI | *Region of Interest* |
| *D* | Hasil jarak referensi (cm) |
| *x1*, *x2* | Jarak *x* terukur (cm) |
| *y1*, *y2* | Jarak *y* terukur (cm) |

# ABSTRAK

Dewasa ini fungsi *Wi-Fi* tidak hanya digunakan untuk aktivitas dalam jaringan internet saja, melainkan *Wi-Fi* dapat digunakan untuk sarana lainnya. Salah satu sarana dari penggunaan *Wi-Fi* adalah sebagai pemantau dalam menghitung jumlah objek, terlebih di dalam ruangan (*indoor monitoring*) yang mana objek yang terhubung dengan *Wi-Fi* sebagai *access point* akan terhitung sebagai jumlah objek. Di sisi lain, *Wi-Fi* memiliki kelemahan, yaitu jumlah orang yang terhubung ke *access point* itu sendiri. Untuk itu diperlukan alternatif sistem *indoor monitoring* yang dapat menghitung jumlah orang serta posisi orang yang berada di dalam ruangan. Pada penelitian ini akan dilakukan pemantauan jumlah dan koordinat objek dalam ruangan (*indoor monitoring system*) menggunakan kamera CCTV. Terdapat dua metode utama yang digunakan pada penelitian ini, yaitu deteksi objek menggunakan *Haar-cascade* dan KLT *tracker*, serta Pengukuran Jarak Berbasis Piksel untuk deteksi koordinat objek. Hasil deteksi objek menunjukkan rata-rata *error* deteksi yang dihasilkan pada satu hingga tiga objek sebesar 0. Hasil pengukuran koordinat objek menunjukkan rata-rata *error* yang dihasilkan pada satu hingga tiga objek sebesar 0,19% hingga 1,79%.

Kata kunci: *Indoor monitoring*, *people counting*, *Haar-like*, KLT *tracker*, jarak berbasis piksel.

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Sistem *indoor monitoring* dewasa ini banyak digunakan diberbagai aplikasi. Salah satunya adalah untuk menghitung jumlah orang di dalam ruangan terpadu. Pada umumnya sistem *indoor monitoring* menggunakan jaringan nirkabel berupa *Wi-Fi* sebagai *Access Point* (AP). Dalam suatu ruangan biasanya terdapat *access point* dan terdapat sejumlah orang di dalamnya, di mana orang tersebut terhubung dengan *access point* sebagai perangkat yang akan menerima data berupa banyaknya orang serta posisi orang yang terhubung ke *access point*. Dari data banyaknya orang yang terhubung ke *access point* tersebut dapat diketahui jumlah banyaknya orang serta posisi orang yang terdapat di ruangan tersebut. Namun dilain sisi, sistem *indoor monitoring* menggunakan *Wi-Fi* masih terdapat kelemahan.

Kelemahan sistem *indoor monitoring* menggunakan *Wi-Fi* adalah jumlah orang yang terhubung ke *access point* itu sendiri. Kelemahan tersebut dikarenakan di dalam tubuh manusia sendiri mengandung 70% air. Kandungan air pada tubuh manusia menyerap lebih banyak *Received Signal Strength* (RSS) daripada udara. Selain kandungan air pada tubuh manusia, jumlah banyaknya orang yang terhubung dengan *access point* akan mempengaruhi nilai RSS yang akan diterima *access point* [1]. Dampak dari nilai RSS yang diterima *access point* adalah pembacaan data berupa perkiraan jumlah orang serta posisi orang yang berada di ruangan tersebut, apakah data jumlah orang serta posisi orang yang diterima *access point* sesuai atau tidak sesuai dengan kondisi riilnya.

Untuk mengurangi kendala tersebut, pada penelitian ini diusulkan untuk membuat dan mengembangkan sistem *indoor monitoring* menggunakan kamera CCTV (*Closed Circuit Television*). Dengan menggunakan kamera CCTV, dapat dilakukan pemantauan jumlah orang serta posisi orang yang berada di dalam ruangan terpadu. Sistem pemantuan jumlah orang (*people counting*) menggunakan kamera termasuk dalam bidang *computer vision* khususnya pemrosesan citra. Sistem *people counting* sendiri dapat berbasis citra gambar maupun citra video [2]. Pada penelitian sebelumnya yang dilakukan oleh [3], mereka menggunakan kamera tunggal untuk menghitung jumlah orang secara *real-time* dengan tujuan dapat mengetahui jumlah orang di dalam suatu bangunan atau ruangan (*indoor people counting*) dan mereka berpendapat bahwa dengan digunakannya *people counting* pada suatu bangunan atau ruangan adalah cara kritis untuk keamanan distrik bisnis dan operasi penyelamatan.

Berdasarkan permasalahan dan solusi yang sudah ada, penulis ingin membuat dan mengembangkan penelitian yang sudah ada terkait sistem *indoor monitoring* untuk memantau jumlah orang dan posisi orang yang berada di dalam ruangan menggunakan kamera CCTV.

## 1.2 Rumusan Masalah

1. Bagaimana cara membuat sistem *indoor monitoring* berbasis *video processing* menggunakan kamera CCTV?
2. Bagaimana unjuk kerja sistem *indoor monitoring* berbasis *video processing* menggunakan kamera CCTV?

## 1.3 Batasan Masalah

1. Penggunaan kamera CCTV sebanyak 1 buah terpasang di sudut ruangan.
2. Perekaman video CCTV yang digunakan adalah secara *online* atau *real-time*.
3. Pendeteksian objek untuk sistem *people counting* dirancang khusus untuk wajah yang menghadap kamera.
4. Penelitian melibatkan objek berjumlah tiga orang.
5. Pengukuran jarak dilakukan pada masing-masing objek apabila objek lebih dari satu (untuk mengetahui jarak antar objek).
6. Penelitian dilakukan pada *grid* berukuran 5x5 (25 titik) dengan jarak per *grid* 60 cm.

## 1.4 Tujuan Penelitian

1. Mengetahui cara membuat sistem *indoor monitoring* berbasis *video processing* menggunakan kamera CCTV.
2. Mengetahui unjuk kerja sistem *indoor monitoring* berbasis *video processing* menggunakan kamera CCTV.

## 1.5 Manfaat Penelitian

1. Untuk membantu dalam pemantauan jumlah, posisi, serta jarak objek dalam suatu ruangan.
2. Untuk memberi kontribusi terhadap pengembangan ilmu pengetahuan dan teknologi pada bidang pemrosesan citra.

# BAB 2
# TINJAUAN PUSTAKA

## 2.1 Studi Literatur

Penelitian sistem *people counting* yang dilakukan oleh [4] menggunakan metode deteksi wajah dengan klasifikasi *Haar-cascade*. Mereka melakukan pengujian terhadap jumlah orang yang melewati kamera dengan *counting line*. Dari hasil pengujian penggunaan metode deteksi wajah menggunakan klasifikasi *Haar-cascade* dihasilkan akurasi 94%, dan *tracking* menggunakan *extended CAMSHIFT* yang dapat menanggulangi tabrakan *tracking window* dengan warna *background* yang sama dengan akurasi 88%. Namun pada penelitian ini sistem pelacakan yang digunakan adalah *extended CAMSHIFT* di mana proses komputasi dan performa pelacakan yang dihasilkan kurang baik. Maka pada penelitian yang akan dilakukan yaitu dengan mengganti metode pelacakan menggunakan KLT *tracker* dan metode deteksi wajah tetap menggunakan *Haar-cascade*. KLT *tracker* dipilih karena proses komputasi dan performa pelacakan yang dihasilkan lebih baik daripada *extended CAMSHIFT* [5].

Penelitian yang dilakukan oleh [6], mereka mengembangkan sistem *people counting* otomatis untuk pemantauan dalam ruangan (*indoor monitoring*). Mereka menggunakan *Histogram of Oriented Gradients* (HOG) sebagai deskriptor fitur untuk deteksi orang dan *Support Vector Machine* (SVM) sebagai klasifikasinya. Dari sistem yang dikembangkan, ketidak akuratan dipengaruhi oleh jarak kamera dengan objek, intensitas cahaya, dan objek yang bertepatan. Hal tersebut dikarenakan HOG hanya mendeteksi objek yang berada di depan dan tidak dapat mendeteksi objek kedua yang berada di belakang (tertutup oleh objek pertama). Pada penelitian ini sistem *people counting* diaplikasikan pada pemantauan dalam ruangan (*indoor monitoring*) dengan menghitung jumlah objek yang terdeteksi di dalam ruangan serta uji coba performa deteksi objek dengan jarak yang ditentukan. Jarak tersebut meliputi jarak objek dengan kamera dan jarak antara objek satu dengan objek dua. Namun deteksi objek yang digunakan pada penelitian ini yaitu mendeteksi seluruh bagian tubuh, sehingga ketika kedua objek berada di posisi yang berhimpitan, sistem hanya dapat mendeteksi objek sebanyak satu objek saja. Maka pada penelitian yang akan dilakukan yaitu dengan mengubah sistem deteksi objek dari deteksi seluruh tubuh menjadi deteksi wajah, dan menambah variasi jarak pengukuran dengan membuat *grid* 5x5 (25 titik koordinat/posisi).

Penelitian [4] dan [6] merupakan landasan bagi penulis untuk mengerjakan penelitian yang akan dilakukan. Kedua penelitian tersebut dapat membantu penulis dalam menganalisis hasil deteksi objek berbasis deteksi wajah menggunakan *Haar-cascade*, pelacakan wajah menggunakan

KLT, dan pengukuran koordinat objek. Deteksi wajah dipilih karena dalam pendeteksiannya deteksi wajah cukup akurat dalam mendeteksi objek yang berhimpitan karena lebar wajah objek tidak lebih lebar dari tubuh objek, sehingga saat objek berhimpitan sistem masih dapat mendeteksi objek tersebut. Untuk mendeteksi wajah digunakan fitur *Haar-like* khususnya *Haar-cascade*. Dari hasil wajah objek yang terdeteksi nantinya akan dihitung sebagai jumlah objek yang terdeteksi.

Penelitian ini dilakukan di dalam ruangan (*indoor monitoring*) dengan penambahan variasi pengukuran jarak dengan membuat *grid* 5x5 atau sebanyak 25 titik koordinat/posisi. Penambahan variasi pengukuran tersebut bertujuan untuk mengetahui jarak objek yang dihasilkan dengan objek berdiri pada titik koordinat yang dibuat. Jarak tersebut meliputi jarak objek dengan kamera dan jarak objek satu dengan objek lainnya.

## 2.2 Tinjauan Teori

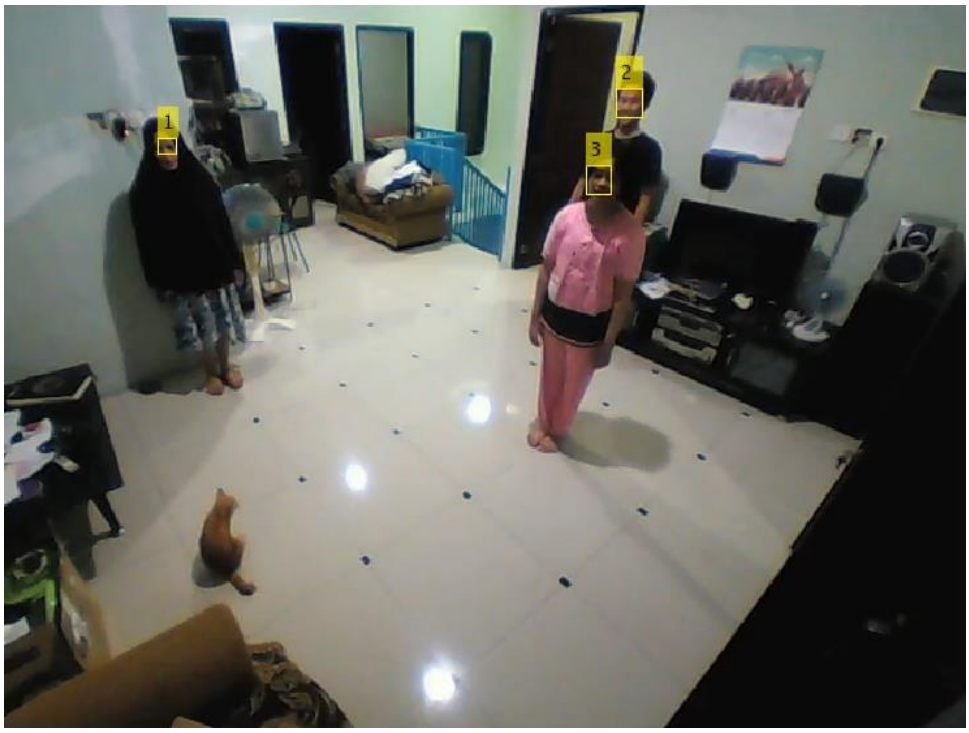
### 2.2.1 Kamera CCTV

*Closed Circuit Television* (CCTV) merupakan perangkat keras berbentuk kamera yang digunakan untuk memantau kondisi dan situasi di tempat tertentu. Penggunaan CCTV pada dasarnya tak lepas dari sebuah pengawasan atau pemantauan terhadap suatu yang diawasi atau dipantau. Umumnya CCTV digunakan untuk pemantauan tempat umum, baik itu luar ruangan (*outdoor*) maupun dalam ruangan (*indoor*). Data berupa gambar atau video dari CCTV kemudian diterima oleh kendali pemantauan untuk mengetahui kondisi tempat yang dipantau secara langsung. Gambar 2.1 menunjukkan kamera CCTV.


Gambar 2.1 Kamera CCTV

### 2.2.2 People Counting

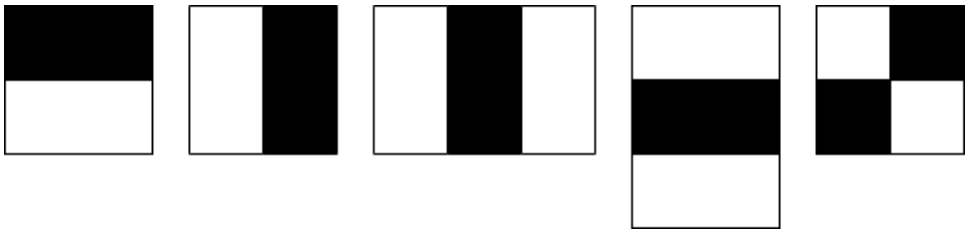
*People counting* merupakan perhitungan terhadap objek berupa manusia baik bergerak ataupun diam. Umumnya objek berada di dalam atau melintasi bagian tertentu sebuah ruangan. *People counting* banyak membantu masyarakat terkait pemantauan jumlah orang baik dalam sektor bisnis, keamanan, pendidikan, dan lain-lain. Dalam aplikasinya di lapangan, *people counting* umumnya menggunakan sensor seperti sensor ultrasonik, *Wi-Fi* (*access point*), dan kamera. Gambar 2.2 menunjukkan aplikasi kamera dalam penggunaan *people counting*.




Gambar 2.2 *People counting* menggunakan kamera

### 2.2.3 Haar-like *Features*

*Haar-like features* pertama kali diperkenalkan oleh Paul Viola dan Michael Jones. *Haar-like features* sendiri merupakan metode klasifikasi yang digunakan untuk mendeteksi objek. Selain itu, *Haar-like features* juga dapat mengenali objek berdasarkan nilai sederhana dari sebuah fitur. Fitur ini didasarkan pada *wavelet haar* yang dikenal dengan daerah terang dan gelap dengan kombinasi-kombinasi persegi yang digunakan untuk pendeteksian objek atau disebut *rectangular feature* [7].



Gambar 2.3 Contoh *Haar-like Features*

Gambar 2.3 menunjukkan setiap fitur terdiri dari gabungan persegi-persegi hitam dan putih. Beberapa fitur *Haar-like* terdiri dari *two-rectangular feature* (vertikal/horisontal), *three-rectangular feature*, dan *four-rectangular feature*. *Haar feature* ditentukan dengan mengurangi rata-rata piksel pada daerah gelap dan rata-rata piksel pada daerah terang. Nilai *Haar-like features* adalah perbedaan antara nilai piksel *gray level* yang terdapat dalam daerah persegi hitam dan persegi putih [8].

Kemudian terdapat beberapa konsep penting untuk mengetahui alur kerja fitur *Haar-like* ini, antara lain sebagai berikut.

1. Citra integral untuk deteksi fitur *Haar-like* untuk perhitungan yang lebih cepat.
2. *Classifier* wajah yang dibangun menggunakan algoritma *learning Adaboost*.
3. Kombinasi dari beberapa *classifier* dalam bentuk *cascade classifier*.

### 2.2.4 Kanade-Lucas-Tomasi *Tracker*

Kanade-Lucas-Tomasi (KLT) merupakan metode pelacakan sebuah gerakan yang terdeteksi berdasarkan urutan cahaya pada citra secara berurutan. Pada bagian citra tertentu, perubahan intensitas cahaya dapat disebabkan oleh pergerakan objek, sumber cahaya, maupun sisi sudut pandang kamera. Selain itu, dalam penggunannya juga menggunakan beberapa asumsi yang umum digunakan, diantaranya sebagai berikut.

1. *Brightness I* (*x, y, t*) bergantung pada koordinat *x* dan *y* yang terdapat dalam bagian yang lebih besar dari gambar.
2. *Brightness* tiap titik pada objek bergerak yang tidak berubah terhadap waktu.

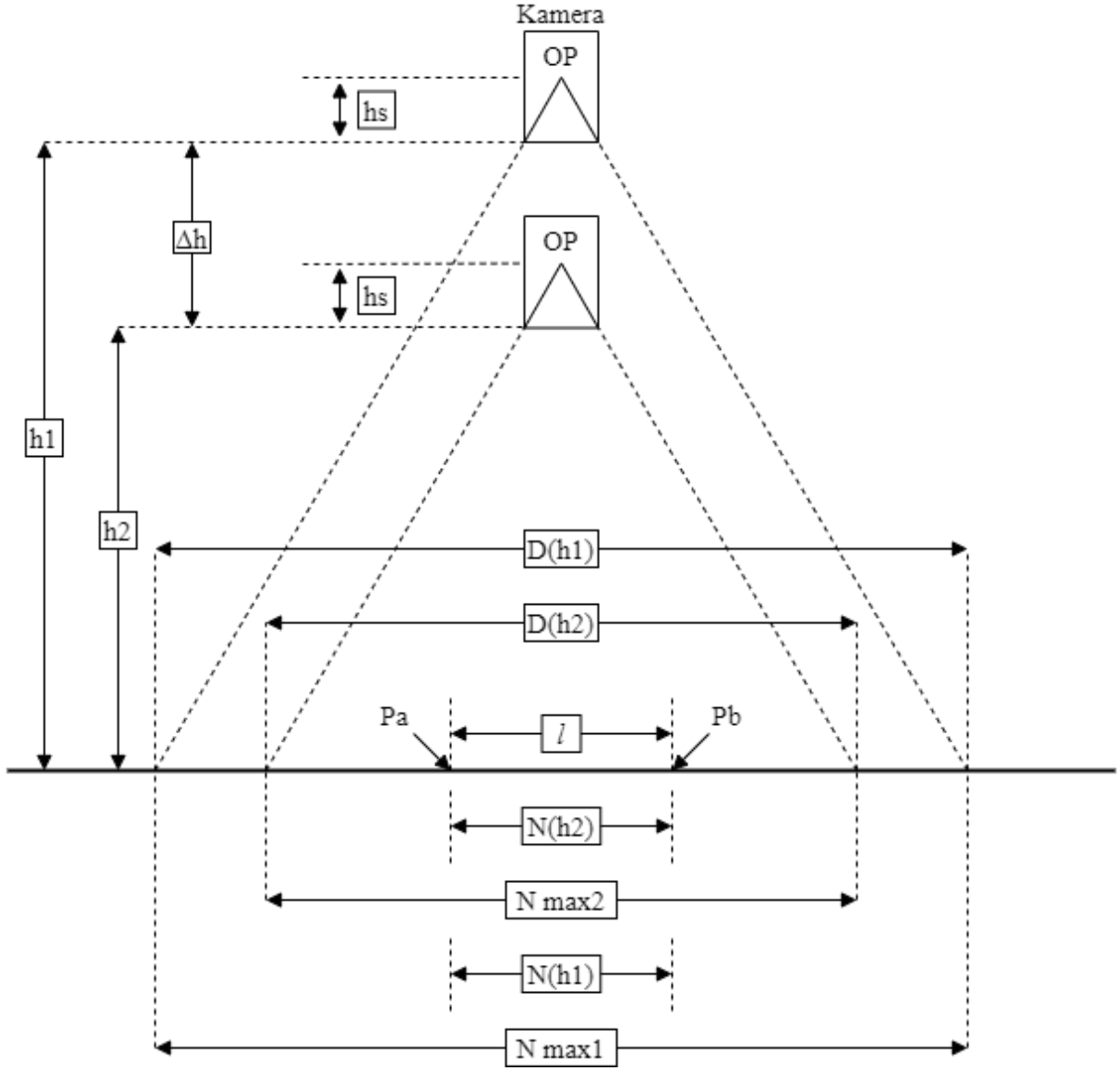
Kemudian terdapat tiga tahap dalam prinsip kerja Kanade-Lucas-Tomasi dalam melacak titik fitur wajah, diantaranya sebagai berikut.

1. Mengekstraksi titik fitur wajah (*feature extraction*).
2. Memilih fitur wajah (*feature selection*).
3. Melacak titik fitur wajah (*feature tracking*).

### 2.2.5 Pengukuran Jarak Berbasis Piksel

Pengukuran jarak terhadap suatu objek dengan menggunakan citra digital dapat dilakukan dengan variasi nilai piksel. Dalam citra digital nilai yang digunakan berupa nilai piksel. Umumnya objek memiliki spesifikasi seperti tinggi dan lebar, dari tinggi dan lebar objek tersebut secara riil yang kemudian objek tersebut dilakukan pengambilan gambar maka secara otomatis di dalam citra digital, tinggi dan lebar dari objek tersebut akan terukur dengan nilai piksel. Dari nilai piksel yang

dihasilkan dari citra digital tersebut kemudian dihitung dengan nilai jarak, panjang, tinggi, dan lebar secara riilnya, dan dihasilkan nilai jarak hasil perhitungan menggunakan variasi piksel.




Gambar 2.4 Diagram skematik pengukuran jarak berbasis piksel

Gambar 2.4 Menunjukkan diagram skematik dari pengukuran jarak berbasis variasi nilai piksel [9]. Pada waktu pengambilan data menggunakan metode ini, posisi kamera berada di depan objek dikarenakan orientasi sudut pandang dari kamera adalah langsung berhadapan dengan objek. Parameter yang ditunjukkan dari Gambar 2.4 diantaranya; *hs* adalah jarak antara *optical origin* (OP) dengan sisi depan kamera, *h1* dan *h2* adalah jarak antara kamera dengan objek, Δ*h* adalah jarak perpindahan posisi kamera *h1* dengan posisi kamera *h2*, *D(h1)* dan *D(h2)* adalah jarak riil terlebar yang dijangkau oleh kamera, *l* adalah lebar riil dari objek, *Pa* dan *Pb* adalah titik tepi piksel dari objek, *N(h1)* dan *N(h2)* adalah lebar piksel dari objek, *Nmax1* dan *Nmax2* adalah nilai piksel terlebar yang dijangkau oleh kamera yang umumnya mengikuti lebar piksel dari citra yang digunakan.

Untuk mengetahui jarak terukur yang dihasilkan dari *h1* dan *h2*, maka jarak tersebut dapat dihitung menggunakan persamaan sebagai berikut.

$$h_1 = \frac{N(h_2)}{N(h_2) - N(h_1)} \times \Delta h - h_s \quad (2.1)$$

$$h_2 = \frac{N(h_1)}{N(h_2) - N(h_1)} \times \Delta h - h_s \quad (2.2)$$

Nilai *N(h1)* dan *N(h2)* pada persamaan (2.1) dan (2.2) adalah jumlah piksel antara titik tepi *Pa* dan *Pb* yang dapat dihitung menggunakan persamaan sebagai berikut.

$$N(h_1) = M_R(h_1) - M_L(h_1) \quad (2.3)$$

$$N(h_2) = M_R(h_2) - M_L(h_2) \quad (2.4)$$

Nilai $\Delta h$ pada persamaan (2.1) dan (2.2) adalah jarak perpindahan posisi kamera sepanjang *h* yang dapat dihitung menggunakan persamaan sebagai berikut.

$$\Delta h = h_1 - h_2 \quad (2.5)$$

Nilai *h1* dan *h2* pada persamaan (2.5) merupakan jarak riil yang harus didefinisikan di awal penghitungan. Sehingga, akan terdapat *h1* dan *h2* untuk definisi jarak awal (jarak referensi) dan *h1* dan *h2* hasil keluaran pengukuran jarak yang telah dihitung dengan persamaan (2.1) dan (2.2). Kemudian nilai *hs* pada persamaan (2.1) dan (2.2) adalah jarak antara *optical origin* (OP) dengan sisi depan kamera yang dapat dihitung menggunakan persamaan sebagai berikut.

$$h_s = \frac{h_{m1}D_{m2} - h_{m2}D_{m1}}{D_{m1} - D_{m2}} \quad (2.6)$$

Dengan:

$h_{m1} = h_1$ = Jarak kamera dengan objek (cm).

$h_{m2} = h_2$ = Jarak kamera dengan objek (cm).

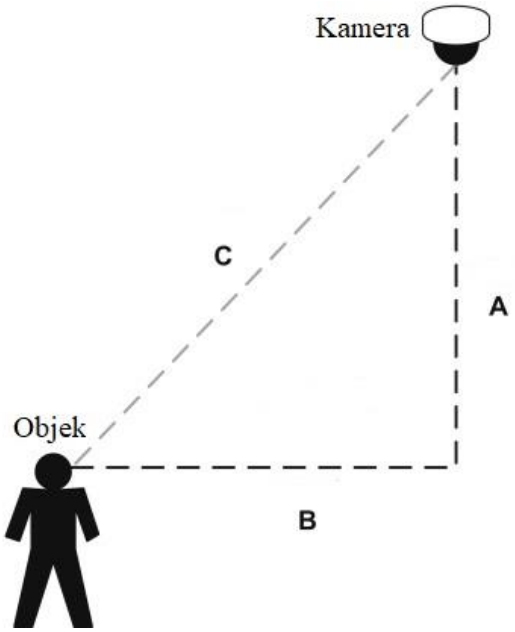
$D_{m1} = D(h_1)$ = Jarak riil terlebar yang dijangkau oleh kamera (cm).

$D_{m2} = D(h_2)$ = Jarak riil terlebar yang dijangkau oleh kamera (cm).

### 2.2.6 Pengukuran Koordinat Objek

Posisi kamera yang dilakukan dalam pengukuran jarak objek dengan metode variasi nilai piksel berhadapan langsung di depan objek. Sedangkan pada penelitian yang dilakukan, posisi kamera berada di sudut atas ruangan. Sehingga diperlukan metode tambahan agar dapat langsung mengukur jarak antara kamera yang berada di sudut atas ruangan dengan objek yang berdiri pada

titik koordinat. Metode tambahan yang digunakan yaitu metode *Pythagoras* dan *Euclidean Distance*.




Gambar 2.5 Metode pengukuran jarak kamera dengan objek

Gambar 2.5 menunjukkan metode pengukuran jarak objek. Pengukuran jarak kamera dengan objek pada penelitian ini menggunakan metode *Pythagoras* dengan persamaan sebagai berikut.

$$a^2 + b^2 = c^2 \tag{2.7}$$

Dari persamaan (2.7) kemudian dicari nilai *c* sebagai nilai jarak terukur antara kamera dengan objek, maka persamaan menjadi sebagai berikut.

$$c = \sqrt{a^2 + b^2} \tag{2.8}$$

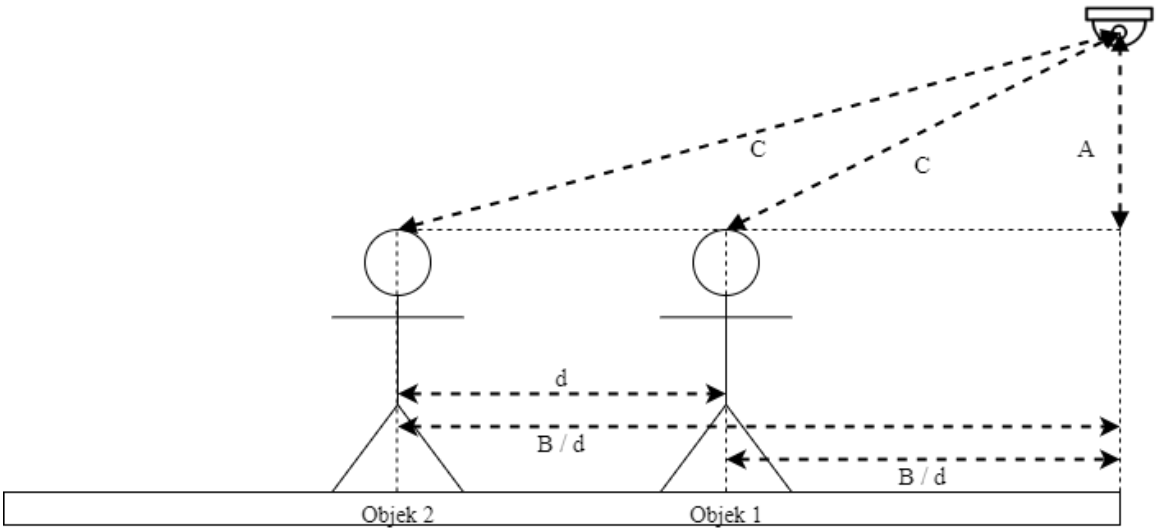
Dengan:

*a* = Selisih ketinggian kamera dengan tinggi objek (cm).

*b* = Jarak referensi (cm).

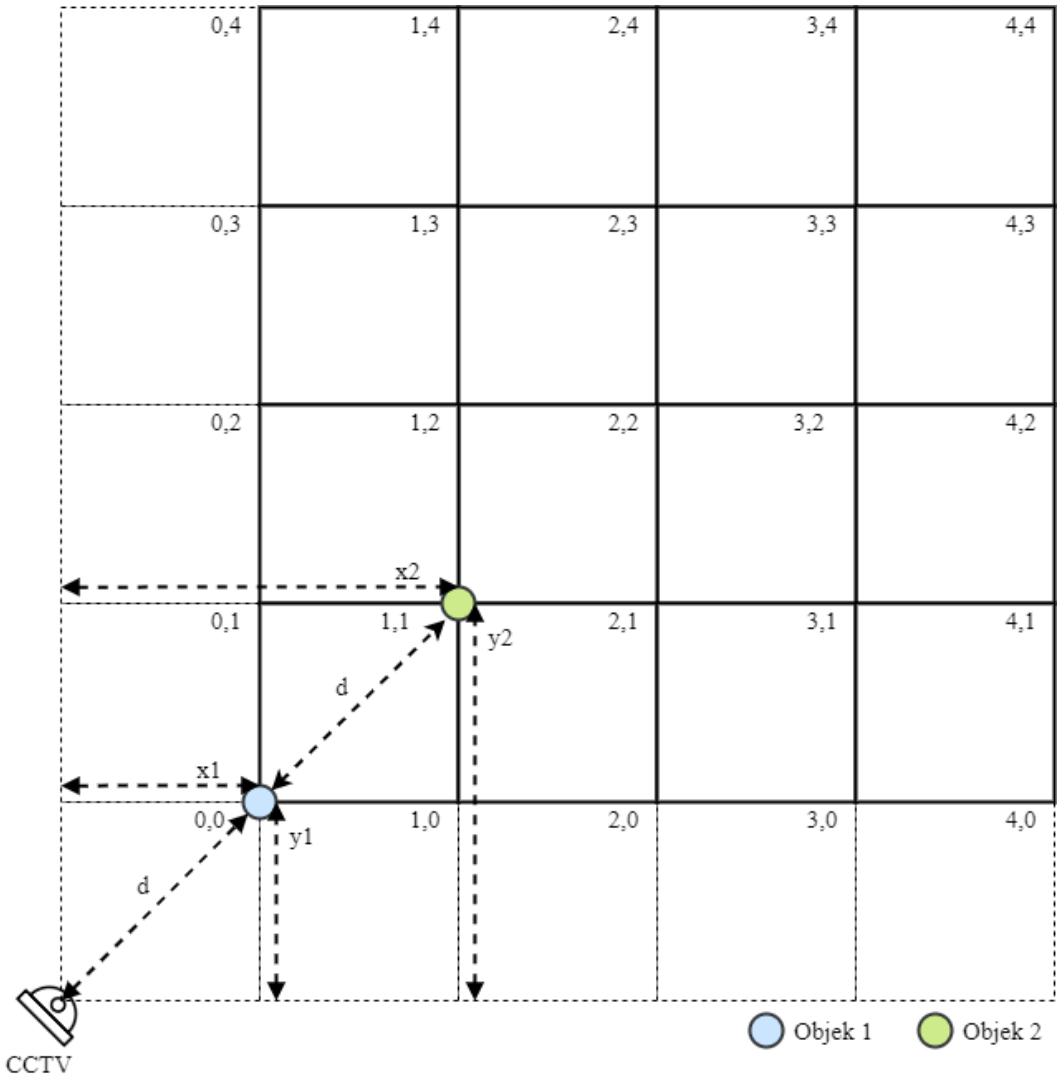
*c* = Jarak terukur (cm).

Untuk mengetahui nilai *b* yang merupakan jarak referensi, yaitu jarak mendatar antara objek yang berdiri sejajar dengan kamera. Maka diperlukan perhitungan jarak menggunakan metode *Euclidean Distance*. Metode ini digunakan untuk mengukur jarak antara objek satu dengan objek lainnya. Gambar 2.6 dan Gambar 2.7 menunjukkan skema tampilan *layout grid* dari tampak samping dan tampak atas yang dibuat dengan ukuran 5x5 titik.




A = Selisih tinggi objek dengan tinggi kamera;
B = Jarak referensi/jarak datar lantai;
C = Jarak terukur objek;
d = Jarak antar objek.

Gambar 2.6 Skema *layout grid* 5x5 titik tampak samping




Gambar 2.7 Skema *layout grid* 5x5 titik tampak atas

Untuk dapat mengetahui nilai *d* yang hasilnya akan digunakan ke dalam persamaan (2.8) yaitu nilai *b*, maka diperlukan persamaan *Euclidean Distance* sebagai berikut.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.9}$$

Dengan:

*d* = Hasil jarak referensi atau jarak mendatar (cm).

$x_1, x_2$ = Jarak *x* terukur dari pengukuran jarak berbasis variasi nilai piksel (cm).

$y_1, y_2$ = Jarak *y* terukur dari pengukuran jarak berbasis variasi nilai piksel (cm).

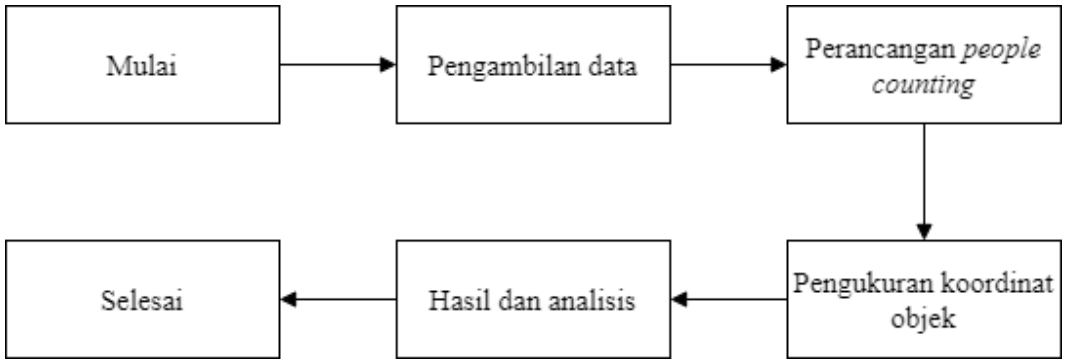
# BAB 3
# METODOLOGI

## 3.1 Alat dan Bahan

1. Laptop Asus VivoBook A412DA dengan spesifikasi prosesor AMD Ryzen 3 3200U dan kartu grafis AMD Radeon Vega 3.
2. Kamera Genius F100.
3. MATLAB R2014a.

## 3.2 Alur Penelitian

Gambar 3.1 menunjukkan diagram blok alur penelitian yang dilakukan pada proses penelitian ini.



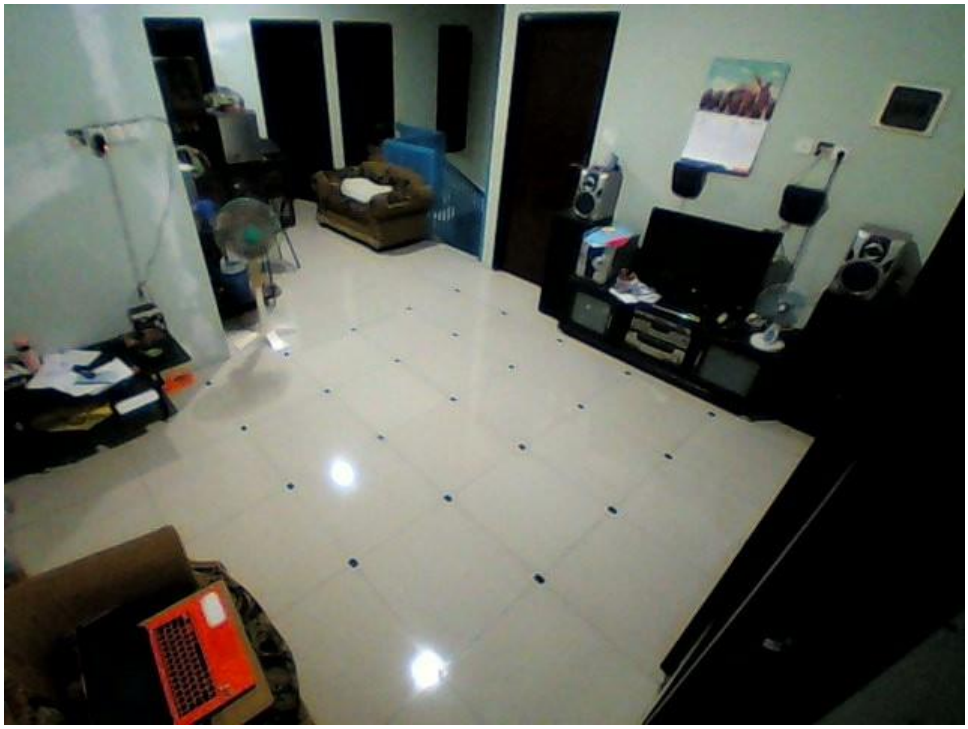

Gambar 3.1 Alur penelitian

### 3.2.1 Pengambilan Data

Tahap pertama yang dilakukan dalam penelitian ini adalah pengambilan data video sebagai data *input* yang selanjutnya akan diproses dan dideteksi. Data video diambil menggunakan kamera CCTV yang dipasang di sudut ruang tengah pada rumah penulis. Ruang tengah dimodifikasi lantainya dengan membuat *grid* 5x5, yaitu pembuatan titik koordinat posisi yang dibuat pada lantai yang diberi tanda hitam sebagai titik posisi sebagai 25 titik. Untuk ukuran lantai yang digunakan yaitu 60 cm x 60 cm, sehingga jarak setiap satu lantai adalah 60 cm. Resolusi video yang digunakan adalah 640x480 piksel. Data video yang diambil adalah data video secara *real-time*, sehingga objek yang masuk ke dalam pantauan kamera akan langsung terhitung sebagai objek terdeteksi.

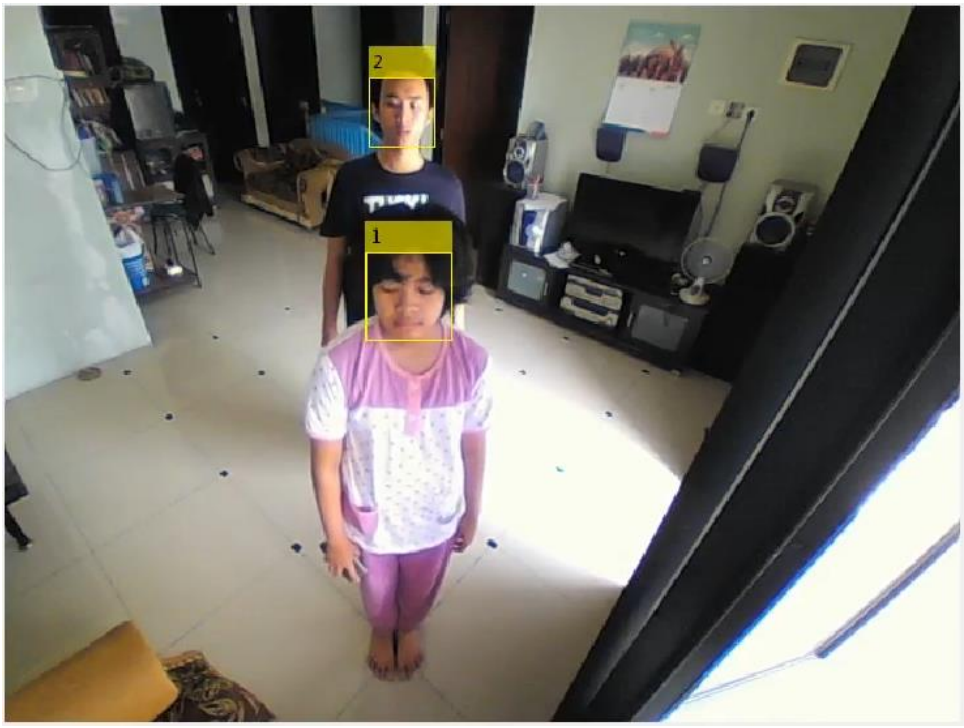
Pengambilan data melibatkan orang sebagai objek berjumlah tiga orang. Pengambilan data video dilakukan sebanyak 40 kali, yaitu untuk objek satu orang dengan 25 titik posisi, objek dua orang dengan 10 titik posisi dengan variasi titik posisi yang berbeda pada masing-masing

objeknya, dan objek tiga orang dengan 5 titik posisi dengan variasi titik posisi yang berbeda pada masing-masing objeknya. Gambar 3.2 menunjukkan sudut pandang kamera terhadap area *grid* untuk pengambilan data. Gambar 3.3, Gambar 3.4, dan Gambar 3.5 menunjukkan sampel pengambilan data dengan jumlah objek satu orang, dua orang, dan tiga orang.



Gambar 3.2 Sudut pandang kamera terhadap area *grid*


Gambar 3.3 Sampel pengambilan data dengan objek satu orang



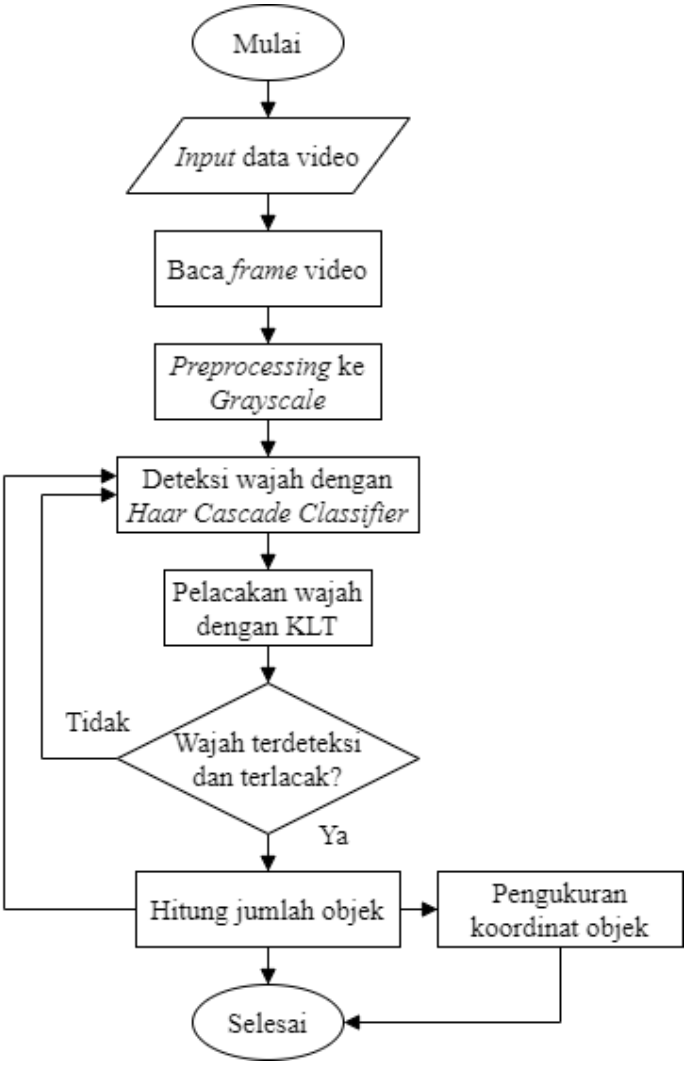

Gambar 3.4 Sampel pengambilan data dengan objek dua orang


Gambar 3.5 Sampel pengambilan data dengan objek tiga orang

### 3.2.2 Perancangan *People Counting*

Data video yang digunakan sebagai masukan ke komputer dan terproses secara *real-time* akan dideteksi menggunakan *Haar Cascade Classifier* untuk mendeteksi wajah orang yang terpantau dalam kamera. Kemudian setelah wajah objek terdeteksi, langkah berikutnya adalah

melakukan *tracking* terhadap wajah yang telah terdeteksi menggunakan KLT *tracker*. Setelah deteksi wajah dan *tracking* wajah dilakukan, langkah selanjutnya adalah mengukur jarak objek yang terdiri jarak objek terhadap kamera dan jarak objek satu dengan objek yang lainnya menggunakan variasi piksel, *Pythagoras*, dan *Euclidean Distance*.




Gambar 3.6 Diagram alir perancangan *people counting* dan pengukuran jarak

Gambar 3.6 menunjukkan diagram alir untuk perancangan sistem *people counting* dan pengukuran jarak yang digunakan dalam penelitian ini. Penjelasan lebih lanjut mengenai masing-masing langkah yang dilakukan adalah sebagai berikut.

1. *Input* Data Video

   *Input* yang digunakan untuk mendeteksi wajah untuk menghitung jumlah orang adalah data video yang direkam secara langsung (*real-time*) melalui kamera CCTV untuk dideteksi menggunakan MATLAB. Data video yang digunakan berupa deteksi jumlah objek yang terpantau oleh kamera CCTV dengan jumlah objek mulai dari satu orang, dua orang, dan tiga orang. Kemudian titik posisi objek dibuat bermacam-macam sehingga didapat variasi posisi

dan jarak yang dihasilkan antar objek. Gambar 3.7 menunjukkan bagian program yang digunakan untuk *input* data video dari kamera CCTV.

```
vidObj = webcam();
```

Gambar 3.7 *Listing* program *input* data video

2. Baca *Frame* Video

Setelah data video diterima oleh MATLAB secara *real-time*, data video tersebut dibaca setiap *frame*-nya selama data video berjalan hingga data video tersebut diberhentikan. Gambar 3.8 menunjukkan bagian program yang digunakan untuk membaca *frame* video.

```
frameNumber = 0;
frameNumber = frameNumber + 1;
```

Gambar 3.8 *Listing* program baca *frame* video

3. *Preprocessing* ke *Grayscale*

Data video yang dijalankan masih dalam format warna *Red-Green-Blue* (RGB) kemudian akan dilakukan *preprocessing* dengan mengubahnya ke dalam format keabuan (*grayscale*). Hal tersebut dikarenakan fitur *Haar-like* dan KLT *tracker* hanya dapat membaca nilai yang dihasilkan dari perubahan intensitas cahaya pada derajat keabuan *grayscale*. Gambar 3.9 menunjukkan bagian program yang digunakan untuk *preprocessing* video dari RGB ke *grayscale*.

```
framergb = snapshot(vidObj);
frame = rgb2gray(framergb);
```

Gambar 3.9 *Listing* program *preprocessing* ke *grayscale*

4. Deteksi Wajah dengan *Haar Cascade Classifier*

Pada perancangan ini digunakan fitur *Haar-like* dengan beberapa kombinasi *classifier* atau disebut sebagai *Haar Cascade Classifier*. Penggunaan fitur ini diperlukan data wajah yang akan dilatih dan hasilnya berupa *classifier* yang akan digunakan untuk mengenal daerah wajah dan non-wajah. Data pelatihan disimpan dalam file XML dan data tersebut akan digunakan sebagai *classifier* untuk *people counter*. *Classifier* yang telah didapat sebelumnya akan dipakai dalam pendeteksian wajah selanjutnya. *Classifier* akan menghasilkan nilai positif dan nilai negatif untuk wajah. Data *classifier* XML hasil pelatihan wajah bisa didapat dari beberapa sumber seperti hasil pelatihan dari beberapa *dataset* atau dari OpenCV [4].

Untuk pengujian deteksi wajah akan dicari parameter terbaik untuk digunakan pada *people counting*. Deteksi wajah ini penting dilakukan karena jika tidak ada wajah yang

terdeteksi maka tidak ada wajah yang dapat di-*tracking* dan dihitung. Jika ada wajah yang terdeteksi, maka *Region of Interest* (ROI) disimpan kemudian wajah dapat di-*tracking* dan dihitung. Menunjukkan diagram alir perancangan deteksi wajah untuk sistem *people counting*.


Gambar 3.10 Diagram alir deteksi wajah dengan *Haar Cascade Classifier*

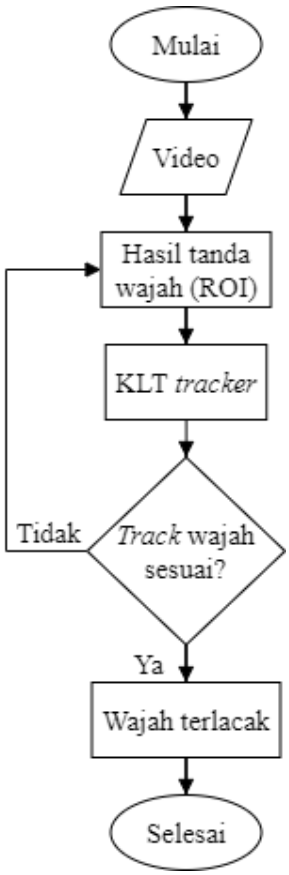
Gambar 3.10 menunjukkan diagram alir yang merepresentasikan alur kerja dari sistem deteksi wajah menggunakan *Haar Cascade Classifier*. Fitur-fitur dari *Haar-like* yang digunakan untuk mendeteksi bagian-bagian wajah, kemudian dari beberapa hasil deteksi fitur tersebut digabungkan menjadi *integral image* [7]. Gambar 3.11 menunjukkan bagian program yang digunakan untuk deteksi wajah objek menggunakan fitur *Haar Cascade*.

```
faceDetector = vision.CascadeObjectDetector('Haar.xml','MergeThreshold',5);
% detektor wajah dengan Haar
```

Gambar 3.11 *Listing* program deteksi wajah menggunakan fitur *Haar Cascade*

5. Pelacakan Wajah dengan Kanade-Lucas-Tomasi

Sistem *tracking* untuk deteksi wajah yang digunakan yaitu Kanade-Lucas-Tomasi (KLT). Prinsip kerja pada *tracker* KLT ini terdiri dari tiga tahapan dasar dalam melacak titik fitur wajah, diantaranya *feature extraction*, *feature selection*, dan *feature tracking*. Menunjukkan diagram alir perancangan *tracking* KLT.




Gambar 3.12 Diagram alir pelacakan wajah dengan KLT

Gambar 3.12 menunjukkan diagram alir yang merepresentasikan alur kerja dari sistem pelacakan wajah menggunakan KLT. Gambar 3.13 menunjukkan bagian program yang digunakan untuk memanggil fungsi dari KLT *tracker*, yaitu program fungsi *MultiObjectTrackerKLT* yang digunakan untuk melacak objek yang terdeteksi sebelumnya (program fungsi tersedia pada lampiran) [10].

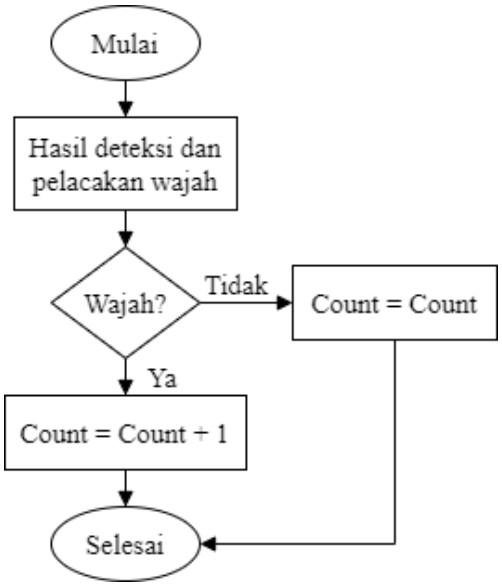
```
tracker = MultiObjectTrackerKLT; % fungsi tracking wajah dengan KLT
```

Gambar 3.13 *Listing* program pelacakan wajah menggunakan KLT

6. Hitung Jumlah Objek

Penghitungan objek dilakukan apabila wajah objek telah terdeteksi dan terlacak, kemudian akan dihitung sebagai objek pertama. Apabila objek lebih dari satu dan semua wajah telah terdeteksi dan terlacak, kemudian akan dihitung sebagai objek berikutnya (kedua, ketiga, dan seterusnya). Jika wajah objek gagal terdeteksi dan terlacak, maka tidak dapat dihitung sebagai jumlah objek. Kemudian sistem akan mengulang aksi kembali untuk mendeteksi wajah, apabila berhasil dilanjutkan dengan pelacakan wajah. Setelah semua aksi untuk mendeteksi dan melacak wajah berhasil, objek akan terhitung sebagai jumlah objek. Apabila tidak berhasil, sistem akan mengulang-ulang aksinya hingga berhasil terdeteksi dan

menghitung semua jumlah objek yang terdeteksi. Gambar 3.14 menunjukkan diagram alir yang merepresentasikan cara kerja sistem menghitung objek dan Gambar 3.15 menunjukkan bagian program yang digunakan untuk menghitung jumlah objek yang terdeteksi berdasarkan objek yang telah terdeteksi wajahnya.



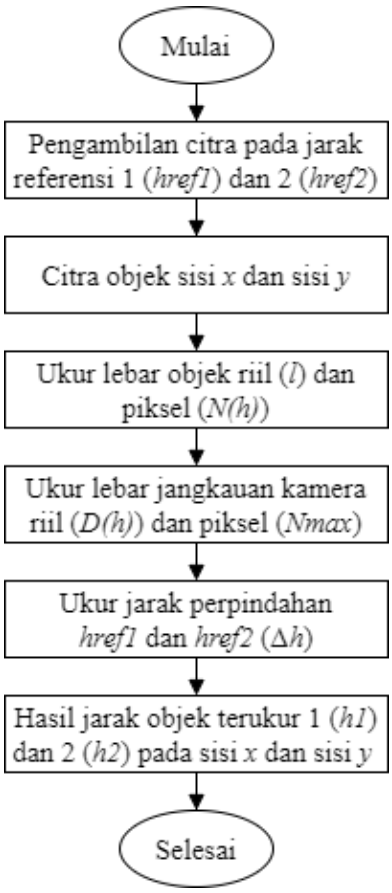

Gambar 3.14 Diagram alir penghitung jumlah objek (*people counting*)

```
if mod(frameNumber, 5) == 0
    bboxes = 1*faceDetector.step(imresize(frame, 1));
    if ~isempty(bboxes)
        tracker.addDetections(frame, bboxes);
    end
else
    % Track faces
    tracker.track(frame);
end
% menampilkan wajah terdeteksi dan terhitung dengan bounding box dan tracking wajah
set(handles.text2,'String',num2str(tracker.NextId-1));
displayFrame = insertObjectAnnotation(framergb, 'rectangle',...
    tracker.Bboxes, tracker.BoxIds);
imshow(displayFrame);
```

Gambar 3.15 *Listing* program menghitung jumlah objek

### 3.2.3 Pengukuran Koordinat Objek

Pengukuran koordinat objek yang dilakukan meliputi pengukuran jarak objek terhadap kamera dan jarak objek satu dengan objek lainnya. Pengukuran jarak dilakukan pada grid 5x5, yaitu pembuatan titik koordinat posisi yang dibuat pada lantai yang diberi tanda hitam sebagai titik posisi sebagai 25 titik. Untuk ukuran lantai yang digunakan yaitu 60 cm x 60 cm, sehingga jarak setiap satu lantai adalah 60 cm.




Gambar 3.16 Diagram alir pengukuran koordinat objek

Gambar 3.16 menunjukkan diagram alir untuk pengukuran koordinat objek yang digunakan untuk mengukur jarak objek di dalam penelitian ini. Penjelasan lebih lanjut mengenai masing-masing langkah yang dilakukan adalah sebagai berikut.

1. Pengambilan Citra pada Jarak Referensi 1 (*href1*) dan 2 (*href2*)

   Pada tahap ini, pengambilan citra dilakukan dengan jarak awal yang ditentukan atau jarak referensi yang nantinya jarak referensi akan dibandingkan dengan jarak terukur untuk mengetahui nilai jarak yang dihasilkan apakah melebihi jarak referensi atau bahkan kurang dari jarak referensi. Contoh untuk tahap ini adalah jarak referensi pertama dengan jarak 120 cm dan jarak referensi kedua dengan jarak 60 cm, jarak referensi yang dimaksud adalah jarak antara kamera dengan objek, di mana posisi kamera berada tepat di depan objek.

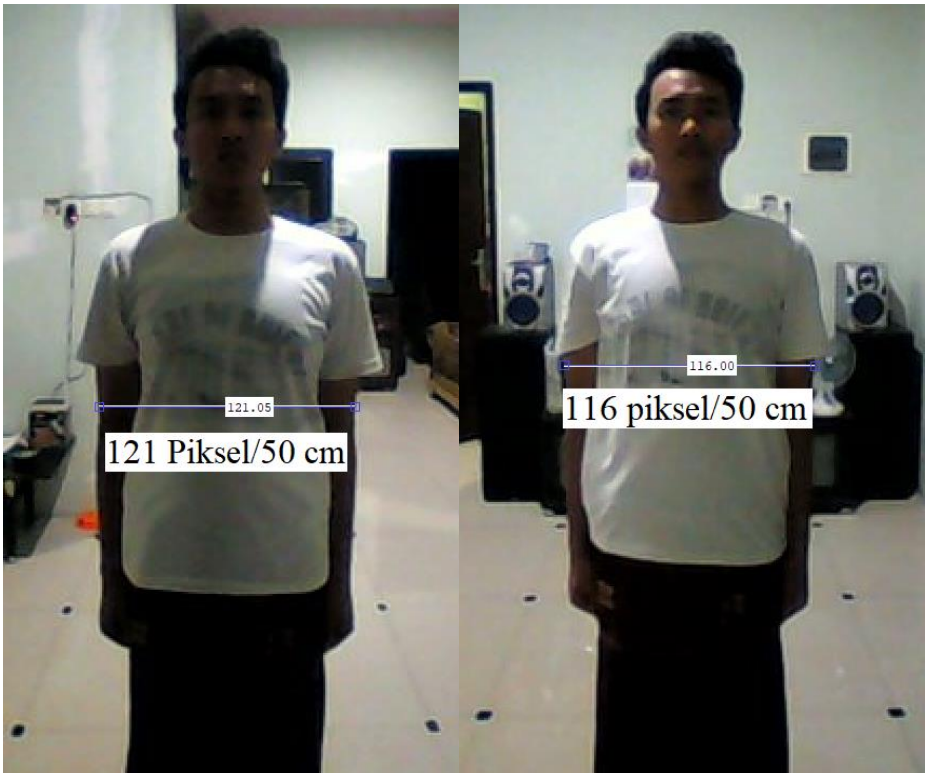
2. Citra Objek Sisi *x* dan Sisi *y*

   Pengambilan citra yang dilakukan terhadap objek adalah pengambilan citra saat objek menghadap koordinat *x* dan saat objek menghadap koordinat *y*, seperti yang ditunjukkan pada Gambar 3.17.

Gambar 3.17 Citra objek menghadap sisi *y* (kiri) dan sisi *x* (kanan)

3. Ukur Lebar Objek Riil (*l*) dan Piksel (*N(h)*)

Lebar objek diukur secara riil dan secara nilai piksel. Lebar objek jika diukur secara riil menggunakan penggaris menghasilkan lebar 50 cm, sedangkan untuk lebar objek jika diukur dalam nilai piksel dihasilkan lebar *n* piksel, seperti ditunjukkan pada Gambar 3.18.




Gambar 3.18 Lebar objek dalam piksel dan riil untuk sisi *x* (kiri) dan sisi *y* (kanan)

4. Ukur Lebar Jangkauan Kamera Riil (*D(h)*) dan Piksel (*Nmax*)

Kamera umumnya memiliki sudut lensa yang berpengaruh terhadap jangkauan terlebar yang dapat ditangkap oleh kamera. Jangkauan terlebar dalam piksel umumnya nilai pikselnya mengikuti resolusi piksel dari citra, seperti contoh citra dengan resolusi 640x480, jangkauan terlebar dari citra tersebut dalam piksel adalah 640 piksel. Untuk jangkauan terlebar secara

riilnya diukur secara manual. Cara pengukurannya yaitu objek yang berdiri pada titik koordinat diharuskan untuk berjalan menuju tepi kanan dan kiri hingga objek berada dititik tak terlihat oleh kamera. Kemudian objek memberi tanda pada lapangan sebagai lebar jangkauan secara riilnya. Gambar 3.19 menunjukkan lebar jangkauan kamera dalam piksel dan riil dengan contoh objek berjarak 180 cm dari kamera.


Gambar 3.19 Lebar jangkauan kamera dalam piksel dan riil

5. Ukur Jarak Perpindahan *href1* dan *href2* ($\Delta h$)

   Jarak referensi atau jarak awal pengambilan citra didefinisikan sebagai *href1* dan *href2*. Hasil selisih dari jarak referensi *href1* dan *href2* adalah jarak perpindahan pengambilan citra atau $\Delta h$. Di mana *href1* adalah jarak referensi terjauh dan *href2* adalah jarak referensi terdekat.

6. Hasil Jarak Objek Terukur (*h1* dan *h2*)

   Hasil jarak objek terukur *h1* dan *h2* didapatkan daripada masing-masing koordinat *x* maupun koordinat *y*. Hasil jarak objek terukur kemudian dibandingkan dengan jarak referensi untuk mengetahui nilai *error* yang dihasilkan dari pengukuran koordinat objek berbasis variasi nilai piksel.

### 3.2.4 Hasil dan Analisis

Hasil dan analisis terhadap deteksi wajah untuk penghitungan orang dan *error* yang dihasilkan dari *people counting*, yaitu perbandingan jumlah objek riil dengan jumlah objek yang

terdeteksi dari penggunaan fitur *Haar-like* (*Haar Cascade Classifier*) dan KLT *tracker*. Selain itu, sudut pandang kamera dan intensitas cahaya juga berpengaruh terhadap performa deteksi wajah dan *error* yang dihasilkan.

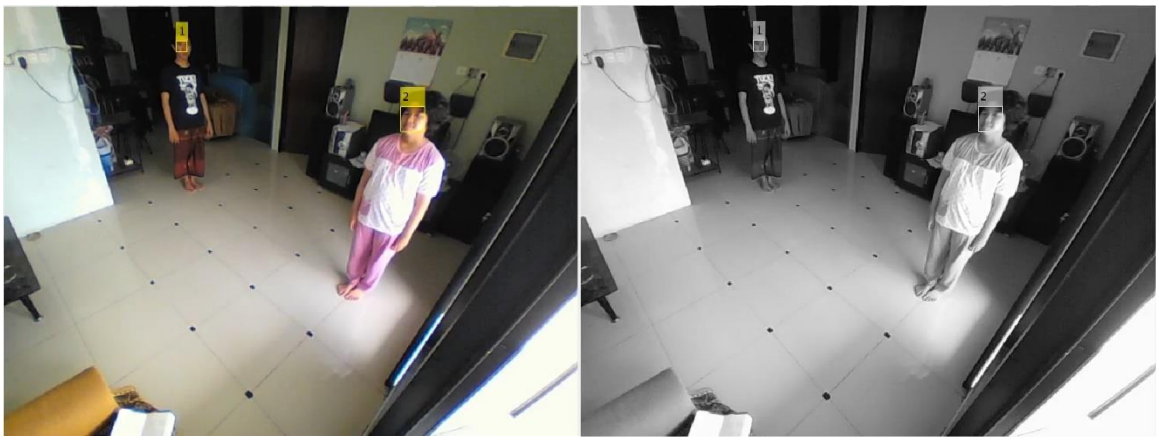
Hasil dan analisis terhadap jarak dan *error* yang dihasilkan dari pengukuran koordinat objek. Pengukuran koordinat objek meliputi jarak kamera dengan objek, jarak objek satu dengan objek lainnya baik jarak riil maupun jarak terukur. Perlu diketahui untuk objek berjumlah tiga orang dengan tinggi yang berbeda, sehingga dengan tinggi yang berbeda tersebut juga berpengaruh terhadap jarak dan *error* yang dihasilkan.

# BAB 4
# HASIL DAN PEMBAHASAN

## 4.1 Hasil Perancangan Sistem *Indoor Monitoring* Menggunakan Kamera CCTV

Pada bagian ini ditampilkan hasil perancangan sistem *indoor monitoring* menggunakan kamera CCTV. Sistem ini dibuat melalui beberapa proses, diantaranya masukan video dari kamera CCTV yang terhubung langsung dengan komputer, *preprocessing* ke *grayscale*, deteksi wajah menggunakan *Haar Cascade Classifier*, *tracking* wajah menggunakan Kanade-Lucas-Tomasi, dan yang terakhir adalah penghitungan jumlah objek yang didapatkan dari wajah objek yang telah terdeteksi dan terlacak.

Pada langkah pertama setelah video dari kamera CCTV diterima komputer, kemudian video dilakukan langkah *preprocessing* ke *grayscale* seperti ditunjukkan pada Gambar 4.1. Langkah ini dilakukan agar mengubah warna video aslinya (RGB) menjadi warna keabuan (*grayscale*). Karena dalam proses selanjutnya, fitur deteksi wajah dan fitur pelacakan wajah yang digunakan dapat bekerja dengan baik untuk membandingkan nilai hitam dan putih dalam menseleksi daerah ROI wajah yang dihasilkan dari video yang sudah di-*preprocces* menjadi *grayscale*. Sehingga langkah *preprocessing* ini sangat penting dilakukan untuk menghasilkan performa sistem yang baik ketika akan digunakan fitur deteksi wajah dan fitur pelacakan wajah.



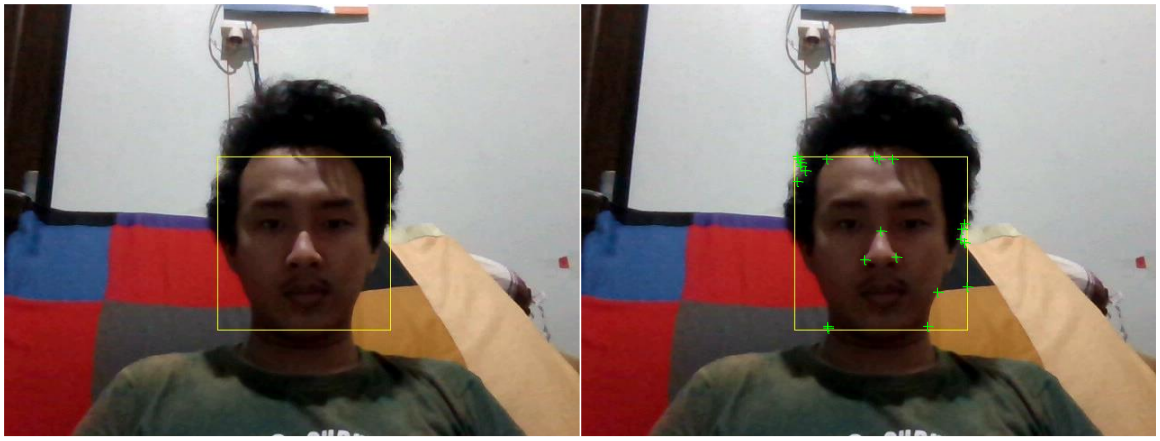
Gambar 4.1 *Preprocessing* video dari RGB ke *grayscale*

Pada langkah selanjutnya setelah dilakukannya *preprocessing* adalah penggunaan fitur *Haar Cascade Classifier* yang berfungsi sebagai detektor wajah objek. Fitur ini nantinya akan difungsikan sebagai penghitung jumlah objek yang terdeteksi. Ketika wajah objek terdeteksi dan muncul *bounding box*, kemudian objek tersebut akan dihitung sebagai jumlah objek yang terdeteksi. Fitur *Haar Cascade* yang digunakan untuk deteksi wajah dan hasil wajah terdeteksi seperti ditunjukkan pada Gambar 4.2 dan Gambar 4.3.

Gambar 4.2 Fitur *Haar Cascade* yang digunakan

Gambar 4.3 Hasil wajah terdeteksi menggunakan fitur *Haar Cascade*

Langkah berikutnya setelah wajah objek terdeteksi hasil penggunaan fitur *Haar Cascade* adalah melacak wajah objek (*face tracking*) menggunakan Kanade-Lucas-Tomasi (KLT). Pelacakan wajah dilakukan agar objek yang telah terdeteksi sebelumnya dan nomor yang muncul pada label *bounding box* pada objek tersebut agar *bounding box* beserta nomor labelnya dapat tetap berada pada objeknya ketika objek tersebut bergerak. Hasil wajah objek yang telah terlacak seperti ditunjukkan pada.



Gambar 4.4 Hasil *tracking* KLT pada wajah objek yang terdeteksi

Dari langkah pertama hingga langkah ketiga tersebut kemudian dihasilkan deteksi wajah objek beserta *tracking* menggunakan KLT yang difungsikan sebagai *people counting*, di mana objek yang terhitung sebagai jumlah objek didapat dari wajah objek yang telah terdeteksi dan

terlacak sebelumnya dengan klasifikasi jika yang terdeteksi tersebut adalah wajah maka sistem akan menghitung objek tersebut sesuai jumlah objek yang terpantau oleh kamera CCTV. Di lain sisi jika yang terdeteksi adalah bukan wajah maka sistem tidak akan menghitung objek tersebut atau jumlah objek sesuai jumlah objek terakhir yang terdeteksi. Hasil akhir perancangan sistem *indoor monitoring* menggunakan kamera CCTV seperti ditunjukkan pada Gambar 4.5, Gambar 4.6, dan Gambar 4.7.


Gambar 4.5 Hasil perancangan sistem dengan objek satu orang


Gambar 4.6 Hasil perancangan sistem dengan objek dua orang


Gambar 4.7 Hasil perancangan sistem dengan objek tiga orang

## 4.2 Hasil Perancangan *People Counting* Untuk Sistem *Indoor Monitoring*

Pada bagian ini ditampilkan hasil deteksi wajah untuk sistem *people counting*. Hasil deteksi wajah objek didapat melalui rekaman video *online* menggunakan MATLAB yang terhubung dengan kamera CCTV. Sistem kemudian dilakukan uji coba secara *online* atau *real-time*. Pengujian yang dilakukan yaitu mendeteksi adanya wajah yang terdeteksi oleh kamera kemudian dari hasil wajah terdeteksi akan dihitung sebagai objek. Pengujian dilakukan sebanyak tiga variasi, variasi pertama dengan jumlah objek satu orang pada 25 titik posisi, variasi kedua dengan jumlah objek dua orang pada 10 titik posisi, dan variasi ketiga dengan jumlah objek tiga orang pada 5 titik posisi.

Dari pengujian tersebut didapatkan informasi berupa jumlah orang yang ditandai dengan *bounding box* pada wajah objek yang terdeteksi beserta label angka di atas *bounding box* sebagai hasil dari jumlah objek yang terdeteksi. Hasil dari *people counting* berbasis deteksi wajah akan dibandingkan dengan *error* yang dihasilkan seperti jumlah objek riil dan jumlah objek terdeteksi. Perbandingan hasil *people counting* dengan *error* yang dihasilkan yaitu untuk mengetahui seberapa baik akurasi dari fitur yang digunakan. Hasil deteksi wajah untuk sistem *people counting* ditunjukkan pada Tabel 4.1, Tabel 4.2, dan Tabel 4.3.

### 4.2.1 Pengujian Deteksi Objek Satu Orang

Tabel 4.1 Hasil pengujian deteksi objek satu orang

| No | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi | *Error* |
|---|---|---|---|---|
| 1 | 0,0 | 1 | 1 | 0 |

| No | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi | *Error* |
|---|---|---|---|---|
| 2 | 0,1 | 1 | 1 | 0 |
| 3 | 0,2 | 1 | 1 | 0 |
| 4 | 0,3 | 1 | 1 | 0 |
| 5 | 0,4 | 1 | 1 | 0 |
| 6 | 1,0 | 1 | 1 | 0 |
| 7 | 2,0 | 1 | 1 | 0 |
| 8 | 3,0 | 1 | 1 | 0 |
| 9 | 4,0 | 1 | 1 | 0 |
| 10 | 1,1 | 1 | 1 | 0 |
| 11 | 1,2 | 1 | 1 | 0 |
| 12 | 1,3 | 1 | 1 | 0 |
| 13 | 1,4 | 1 | 1 | 0 |
| 14 | 2,1 | 1 | 1 | 0 |
| 15 | 3,1 | 1 | 1 | 0 |
| 16 | 4,1 | 1 | 1 | 0 |
| 17 | 2,2 | 1 | 1 | 0 |
| 18 | 2,3 | 1 | 1 | 0 |
| 19 | 2,4 | 1 | 1 | 0 |
| 20 | 3,2 | 1 | 1 | 0 |
| 21 | 4,2 | 1 | 1 | 0 |
| 22 | 3,3 | 1 | 1 | 0 |
| 23 | 3,4 | 1 | 1 | 0 |
| 24 | 4,3 | 1 | 1 | 0 |
| 25 | 4,4 | 1 | 1 | 0 |

Data yang ditunjukkan pada Tabel 4.1 adalah data hasil pendeteksian *people counting* untuk objek berjumlah satu orang. Hasil deteksi menunjukkan objek dapat terdeteksi dengan baik mulai dari objek berdiri pada titik 0,0 (titik terdekat dengan kamera) hingga pada titik terjauh, yaitu titik 4,4. Objek terdeteksi dengan baik dikarenakan adanya *bounding box* yang muncul di kepala objek saat objek terpantau oleh kamera dan jumlah *bounding box* yang muncul dihitung sebagai jumlah objek yang terdeteksi. *Error* yang dihasilkan dari jumlah objek secara riil dan jumlah objek yang terdeteksi adalah nol. Hal tersebut dikarenakan jumlah objek secara riil dan jumlah objek yang terdeteksi relatif sama, yaitu sama-sama berjumlah satu orang.

Dari perbandingan antara jumlah objek secara riil dan jumlah objek terdeteksi terhadap *error* yang dihasilkan seperti ditunjukkan pada Tabel 4.1, dapat dianalisis bahwa objek dapat terdeteksi seluruhnya dengan baik karena objek berada di dalam jarak pantau kamera yang masih mampu mendeteksi objek hingga titik terjauh. Selain itu intensitas cahaya yang terang dikarenakan pengujian objek satu orang dilakukan pada siang hari membuat wajah objek sangat mudah dikenali dan dideteksi oleh sistem, sehingga intensitas cahaya sangat berpengaruh terhadap performa sistem dalam mendeteksi objek.

### 4.2.2 Pengujian Deteksi Objek Dua Orang

Tabel 4.2 Hasil pengujian deteksi objek dua orang

| No | Posisi (x,y) | | Jumlah Objek Riil | Jumlah Objek Terdeteksi | *Error* |
|---|---|---|---|---|---|
| | Objek 1 | Objek 2 | | | |
| 1 | 0,0 | 1,1 | 2 | 2 | 0 |
| 2 | 0,0 | 2,1 | 2 | 2 | 0 |
| 3 | 1,2 | 4,0 | 2 | 2 | 0 |
| 4 | 4,1 | 0,4 | 2 | 2 | 0 |
| 5 | 4,4 | 0,3 | 2 | 2 | 0 |
| 6 | 2,4 | 2,0 | 2 | 2 | 0 |
| 7 | 3,3 | 3,1 | 2 | 2 | 0 |
| 8 | 1,0 | 1,3 | 2 | 2 | 0 |
| 9 | 0,2 | 0,3 | 2 | 2 | 0 |
| 10 | 0,0 | 4,4 | 2 | 2 | 0 |

Data yang ditunjukkan pada Tabel 4.2 adalah data hasil pendeteksian *people counting* untuk objek berjumlah dua orang. Hasil deteksi menunjukkan objek dapat terdeteksi dengan baik mulai dari objek pertama dan objek kedua berdiri pada titik yang ditentukan seperti yang ditunjukkan pada Tabel 4.2. Objek terdeteksi dengan baik dikarenakan adanya *bounding box* yang muncul di kepala objek saat objek terpantau oleh kamera dan jumlah *bounding box* yang muncul dihitung sebagai jumlah objek yang terdeteksi. *Error* yang dihasilkan dari jumlah objek secara riil dan jumlah objek yang terdeteksi adalah nol. Hal tersebut dikarenakan jumlah objek secara riil dan jumlah objek yang terdeteksi relatif sama, yaitu sama-sama berjumlah dua orang.

Dari perbandingan antara jumlah objek secara riil dan jumlah objek terdeteksi terhadap *error* yang dihasilkan seperti ditunjukkan pada Tabel 4.2, dapat dianalisis bahwa objek dapat terdeteksi seluruhnya dengan baik karena objek berada di dalam jarak pantau kamera yang masih mampu mendeteksi objek hingga titik terjauh. Selain itu intensitas cahaya yang terang dikarenakan

pengujian objek dua orang dilakukan pada siang hari membuat wajah objek sangat mudah dikenali dan dideteksi oleh sistem, sehingga intensitas cahaya sangat berpengaruh terhadap performa sistem dalam mendeteksi objek.

#### 4.2.3 Pengujian Deteksi Objek Tiga Orang

Tabel 4.3 Hasil pengujian deteksi objek tiga orang

| No | Posisi (x,y) | | | Jumlah Objek Riil | Jumlah Objek Terdeteksi | *Error* |
|---|---|---|---|---|---|---|
| | Objek 1 | Objek 2 | Objek 3 | | | |
| 1 | 0,1 | 0,2 | 0,0 | 3 | 3 | 0 |
| 2 | 3,3 | 2,2 | 1,1 | 3 | 3 | 0 |
| 3 | 0,0 | 2,2 | 4,4 | 3 | 5 | 2 |
| 4 | 2,1 | 4,2 | 0,4 | 3 | 3 | 0 |
| 5 | 3,1 | 1,2 | 2,4 | 3 | 3 | 0 |

Data yang ditunjukkan pada Tabel 4.3 adalah data hasil pendeteksian *people counting* untuk objek berjumlah tiga orang. Hasil deteksi menunjukkan terdapat *error* pada percobaan nomor tiga dengan objek pertama di posisi (0,0), objek kedua di posisi (2,2), dan objek ketiga di posisi (4,4). Jumlah objek yang terdeteksi oleh sistem yaitu berjumlah lima objek terdeteksi, sedangkan jumlah objek riilnya yaitu tiga objek, sehingga didapatkan *error* sebanyak dua.

Dari percobaan pada nomor tiga seperti ditunjukkan pada Tabel 4.3, dapat dianalisis bahwa posisi objek ketiga (4,4) dianggap terlalu jauh dari pantauan kamera, sehingga pengaturan nilai *threshold* harus disesuaikan agar objek ketiga dapat terdeteksi oleh sistem. Jika nilai *threshold* diatur pada nilai di atas 4-5, sistem hanya mendeteksi objek sebanyak dua objek. Dan jika nilai *threshold* diatur pada nilai di bawah 4, sistem mendeteksi objek sebanyak lebih dari tiga objek. Sehingga percobaan nomor tiga didapati *error* objek riil dan objek terdeteksi. Selain itu intensitas cahaya yang kurang terang dikarenakan pengujian objek tiga orang dilakukan pada malam hari membuat wajah objek sedikit sulit dikenali dan dideteksi oleh sistem, sehingga intensitas cahaya sangat berpengaruh terhadap performa sistem dalam mendeteksi objek.

### 4.3 Hasil Pengukuran Koordinat Objek

#### 4.3.1 Pengukuran Koordinat Objek Satu Orang

Pengukuran koordinat objek berjumlah satu orang ditunjukkan pada Tabel 4.4. Hasil pengukuran yang dihasilkan didapat dari pengukuran jarak objek dengan kamera. Jarak objek dengan kamera meliputi jarak objek riil (jarak referensi atau B) dan jarak objek terukur (jarak

terukur atau C), sehingga dari perbandingan kedua jarak tersebut dihasilkan nilai *error* pengukuran koordinat.

Tabel 4.4 Hasil pengukuran koordinat untuk objek satu orang

| No | Posisi (x,y) | Jarak Objek Riil (cm) | Jarak Objek Terukur (cm) | *Error* (%) |
|---|---|---|---|---|
| 1 | 0,0 | 84.7083 | 85.7000 | 1.1708 |
| 2 | 0,1 | 134.6911 | 135.3170 | 0.4647 |
| 3 | 0,2 | 188.8788 | 189.3256 | 0.2366 |
| 4 | 0,3 | 247.8220 | 248.1627 | 0.1375 |
| 5 | 0,4 | 306.3915 | 306.6671 | 0.0900 |
| 6 | 1,0 | 133.9290 | 134.5584 | 0.4700 |
| 7 | 2,0 | 190.1547 | 190.5986 | 0.2334 |
| 8 | 3,0 | 247.1799 | 247.5215 | 0.1382 |
| 9 | 4,0 | 305.7325 | 306.0088 | 0.0904 |
| 10 | 1,1 | 169.8570 | 170.3538 | 0.2925 |
| 11 | 1,2 | 216.8047 | 217.1941 | 0.1796 |
| 12 | 1,3 | 268.8053 | 269.1195 | 0.1169 |
| 13 | 1,4 | 323.5661 | 323.8272 | 0.0807 |
| 14 | 2,1 | 215.8422 | 216.2333 | 0.1812 |
| 15 | 3,1 | 269.2390 | 269.5526 | 0.1165 |
| 16 | 4,1 | 324.0744 | 324.3350 | 0.0804 |
| 17 | 2,2 | 254.7303 | 255.0618 | 0.1301 |
| 18 | 2,3 | 301.3126 | 301.5929 | 0.0930 |
| 19 | 2,4 | 351.1750 | 351.4155 | 0.0685 |
| 20 | 3,2 | 297.4735 | 297.7575 | 0.0954 |
| 21 | 4,2 | 347.2055 | 347.4488 | 0.0701 |
| 22 | 3,3 | 335.4182 | 335.6701 | 0.0751 |
| 23 | 3,4 | 380.2191 | 380.4413 | 0.0584 |
| 24 | 4,3 | 380.2191 | 380.4413 | 0.0584 |
| 25 | 4,4 | 420.2710 | 420.4720 | 0.0478 |

Data yang ditunjukkan pada Tabel 4.4 adalah data hasil pengukuran jarak untuk objek satu orang. Parameter yang digunakan dalam pengukuran jarak dengan objek satu orang adalah jarak antara kamera dengan objek, sehingga nilai *error* yang didapat adalah hasil perbangingan dari jarak riil (B) dengan jarak terukur (C). Tinggi badan objek adalah 177 cm dan tinggi kamera CCTV

yang dipasang adalah 190 cm, sehingga didapat selisih tinggi antara objek dengan kamera CCTV sebesar 13 cm (A). Nilai *error* terkecil didapat pada objek dengan posisi 4,4 dengan jarak terukur 420,4720 cm atau 420,2710 cm pada jarak riil dengan nilai *error* sebesar 0,0478%. Nilai *error* terbesar didapat pada objek dengan posisi 0,0 dengan jarak terukur 85,7 cm atau 84,7083 cm pada jarak riil dengan nilai *error* sebesar 1,1708%. Objek pada posisi 0,0 menghasilkan nilai *error* terbesar sedangkan objek pada posisi 4,4 menghasilkan nilai *error* terkecil.

Dari perbandingan antara jarak objek secara riil dan jarak objek terukur terhadap *error* yang dihasilkan seperti ditunjukkan pada Tabel 4.4, dapat dianalisis bahwa posisi objek berpengaruh terhadap nilai jarak yang dihasilkan. Semakin dekat objek dengan kamera maka jarak terukur yang dihasilkan semakin besar (nilai jarak terukur menjauhi jarak riil) dan semakin jauh objek dengan kamera maka jarak terukur yang dihasilkan semakin kecil (nilai jarak terukur mendekati jarak riil). Hal tersebut dikarenakan pengukuran pada sisi jarak terukur (C) yang dilakukan secara diagonal. Sehingga semakin tinggi atau semakin jauh sisi jarak terukur (C) dari sisi jarak riil (B) maka sisi jarak terukur (C) akan menghasilkan nilai jarak yang semakin besar yang menyebabkan nilai *error* yang dihasilkan semakin besar. Rata-rata *error* yang dihasilkan dari pengukuran jarak objek satu orang yaitu sebesar 0,19%.

### 4.3.2 Pengukuran Koordinat Objek Dua Orang

Pengukuran koordinat objek berjumlah dua orang ditunjukkan pada Tabel 4.5. Hasil pengukuran yang dihasilkan didapat dari pengukuran jarak objek satu dengan kamera, jarak objek dua dengan kamera, dan jarak objek satu dengan objek dua. Jarak objek satu dengan kamera dan jarak objek dua dengan kamera masing-masing meliputi jarak objek riil (jarak referensi atau B) dan jarak objek terukur (jarak terukur atau C), sehingga dari perbandingan kedua jarak tersebut dihasilkan nilai *error* pengukuran koordinat dari masing-masing objek. Pengukuran jarak objek satu dengan objek dua dihasilkan dari pengukuran jarak berbasis piksel masing-masing objek (jarak *x* dan *y* masing-masing objek).

Tabel 4.5 Hasil pengukuran koordinat untuk objek dua orang

| No | Posisi (x,y) | | Jarak Objek Riil (cm) | | Jarak Objek Terukur (cm) | | *Error* (%) | | Jarak Objek 1-Objek 2 (cm) |
|---|---|---|---|---|---|---|---|---|---|
| | Objek 1 | Objek 2 | Objek 1 | Objek 2 | Objek 1 | Objek 2 | Objek 1 | Objek 2 | |
| 1 | 0,0 | 1,1 | 84.7083 | 169.8570 | 88.6086 | 170.3538 | 4.6045 | 0.2925 | 85.1504 |
| 2 | 0,0 | 2,1 | 84.7083 | 215.8422 | 88.6086 | 216.2333 | 4.6045 | 0.1812 | 133.8505 |
| 3 | 1,2 | 4,0 | 216.8047 | 305.7325 | 218.3582 | 306.0088 | 0.7165 | 0.0904 | 215.9645 |
| 4 | 4,1 | 0,4 | 324.0744 | 306.3915 | 325.1157 | 306.6671 | 0.3213 | 0.0900 | 301.4859 |
| 5 | 4,4 | 0,3 | 420.2710 | 247.8220 | 420.4720 | 249.1821 | 0.0478 | 0.5488 | 244.0533 |

| No | Posisi (x,y) | | Jarak Objek Riil (cm) | | Jarak Objek Terukur (cm) | | *Error* (%) | | Jarak Objek 1-Objek 2 (cm) |
|---|---|---|---|---|---|---|---|---|---|
| | Objek 1 | Objek 2 | Objek 1 | Objek 2 | Objek 1 | Objek 2 | Objek 1 | Objek 2 | |
| 6 | 2,4 | 2,0 | 351.1750 | 190.1547 | 351.4155 | 191.9240 | 0.0685 | 0.9304 | 241.1216 |
| 7 | 3,3 | 3,1 | 335.4182 | 269.2390 | 335.6701 | 270.4915 | 0.0751 | 0.4652 | 117.4511 |
| 8 | 1,0 | 1,3 | 133.9290 | 268.8053 | 136.4294 | 269.1195 | 1.8669 | 0.1169 | 180.3239 |
| 9 | 0,2 | 0,3 | 188.8788 | 247.8220 | 190.6599 | 248.1627 | 0.9430 | 0.1375 | 61.3361 |
| 10 | 0,0 | 4,4 | 84.7083 | 420.2710 | 88.6086 | 420.4720 | 4.6045 | 0.0478 | 335.5629 |

Data yang ditunjukkan pada Tabel 4.5 adalah data hasil pengukuran jarak untuk objek dua orang. Parameter yang digunakan dalam pengukuran jarak dengan objek dua orang adalah jarak antara kamera dengan objek, sehingga nilai *error* yang didapat adalah hasil perbangingan dari jarak riil (B) dengan jarak terukur (C). Tinggi badan objek wanita adalah 164 cm, tinggi badan objek pria adalah 177 cm, dan tinggi kamera CCTV yang dipasang adalah 190 cm, sehingga didapat selisih tinggi antara objek wanita dengan kamera CCTV sebesar 26 cm (A1) dan 13 cm (A2) untuk selisih tinggi antara objek pria dengan kamera CCTV. Nilai *error* terkecil objek pertama didapat pada objek dengan posisi 4,4 dengan jarak terukur 420,4720 cm atau 420,2710 cm pada jarak riil dengan nilai *error* sebesar 0,0478%. Nilai *error* terbesar objek pertama didapat pada objek dengan posisi 0,0 dengan jarak terukur 88,6086 cm atau 84,7083 cm pada jarak riil dengan nilai *error* sebesar 4,6045%. Nilai *error* terkecil objek kedua didapat pada objek dengan posisi 4,4 dengan jarak terukur 420,4720 cm atau 420,2710 cm pada jarak riil dengan nilai *error* sebesar 0,0478%. Nilai *error* terbesar objek kedua didapat pada objek dengan posisi 2,0 dengan jarak terukur 191,9240 cm atau 190,1547 cm pada jarak riil dengan nilai *error* sebesar 0,9304%.

Dari perbandingan antara jarak objek secara riil dan jarak objek terukur terhadap *error* yang dihasilkan seperti ditunjukkan pada Tabel 4.5, dapat dianalisis bahwa posisi objek berpengaruh terhadap nilai jarak yang dihasilkan. Semakin dekat objek dengan kamera maka jarak terukur yang dihasilkan semakin besar (nilai jarak terukur menjauhi jarak riil) dan semakin jauh objek dengan kamera maka jarak terukur yang dihasilkan semakin kecil (nilai jarak terukur mendekati jarak riil). Hal tersebut dikarenakan pengukuran pada sisi jarak terukur (C) yang dilakukan secara diagonal. Sehingga semakin tinggi atau semakin jauh sisi jarak terukur (C) dari sisi jarak riil (B) maka sisi jarak terukur (C) akan menghasilkan nilai jarak yang semakin besar yang menyebabkan nilai *error* yang dihasilkan semakin besar. Rata-rata *error* yang dihasilkan dari pengukuran jarak objek dua orang yaitu sebesar 1,79% untuk objek pertama dan 0,29% untuk objek kedua.

### 4.3.3 Pengukuran Koordinat Objek Tiga Orang

Pengukuran koordinat objek berjumlah tiga orang ditunjukkan pada Tabel 4.6. Hasil pengukuran yang dihasilkan didapat dari pengukuran jarak objek satu dengan kamera, jarak objek dua dengan kamera, jarak objek tiga dengan kamera, jarak objek satu dengan objek dua, jarak objek dua dengan objek tiga, dan jarak objek satu dengan objek tiga. Jarak objek satu dengan kamera, jarak objek dua dengan kamera, dan jarak objek tiga dengan kamera masing-masing meliputi jarak objek riil (jarak referensi atau B) dan jarak objek terukur (jarak terukur atau C), sehingga dari perbandingan kedua jarak tersebut dihasilkan nilai *error* pengukuran koordinat dari masing-masing objek. Pengukuran jarak objek satau dengan objek dua, jarak objek dua dengan objek tiga, dan jarak objek satu dengan objek tiga dihasilkan dari pengukuran jarak berbasis piksel masing-masing objek (jarak *x* dan *y* masing-masing objek).

Tabel 4.6 Hasil pengukuran koordinat untuk objek tiga orang

| **No** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Posisi (x,y)** | | | | | |
| **Objek 1** | 0,1 | 3,3 | 0,0 | 2,1 | 3,1 |
| **Objek 2** | 0,2 | 2,2 | 2,2 | 4,2 | 1,2 |
| **Objek 3** | 0,0 | 1,1 | 4,4 | 0,4 | 2,4 |
| **Jarak Objek Riil (cm)** | | | | | |
| **Objek 1** | 134,6911 | 335,4182 | 84,7083 | 215,8422 | 269,2390 |
| **Objek 2** | 188,3256 | 254,7303 | 254,7303 | 347,2055 | 216,8047 |
| **Objek 3** | 84,7083 | 169,8570 | 420,2710 | 306,3915 | 351,1750 |
| **Jarak Objek Terukur (cm)** | | | | | |
| **Objek 1** | 137,1776 | 335,6701 | 88,6086 | 217,4025 | 270,4915 |
| **Objek 2** | 189,3256 | 256,0537 | 256,0537 | 347,4488 | 217,1941 |
| **Objek 3** | 88,6086 | 171,8354 | 420,4720 | 307,4927 | 352,1361 |
| ***Error* (%)** | | | | | |
| **Objek 1** | 1,8461 | 0,0751 | 4,6045 | 0,7229 | 0,4625 |
| **Objek 2** | 0,2366 | 0,5196 | 0,5196 | 0,3594 | 0,1796 |
| **Objek 3** | 4,6045 | 1,1647 | 0,0478 | 0,3594 | 0,2737 |
| **Jarak Antar Objek (cm)** | | | | | |
| **Objek 1-Objek 2 (cm)** | 58.4734 | 80.6933 | 170.0235 | 131.9363 | 135.0111 |
| **Objek 2-Objek 3 (cm)** | 119.1639 | 84.8733 | 165.5440 | 266.4188 | 135.0328 |
| **Objek 1-Objek 3 (cm)** | 60.6905 | 165.5624 | 335.5629 | 216.7865 | 191.1339 |

Data yang ditunjukkan pada Tabel 4.6 adalah data hasil pengukuran jarak untuk objek tiga orang. Parameter yang digunakan dalam pengukuran jarak dengan objek tiga orang adalah jarak antara kamera dengan objek, sehingga nilai *error* yang didapat adalah hasil perbangingan dari jarak riil (B) dengan jarak terukur (C). Tinggi badan objek wanita adalah 164 cm, tinggi badan objek pria adalah 177 cm, dan tinggi kamera CCTV yang dipasang adalah 190 cm, sehingga didapat selisih tinggi antara objek wanita dengan kamera CCTV sebesar 26 cm (A1 dan A2) dan 13 cm (A3) untuk selisih tinggi antara objek pria dengan kamera CCTV. Nilai *error* terkecil objek pertama didapat pada objek dengan posisi 3,3 dengan jarak terukur 335,6701 cm atau 335,4182 cm pada jarak riil dengan nilai *error* sebesar 0,0751%. Nilai *error* terbesar objek pertama didapat pada objek dengan posisi 0,0 dengan jarak terukur 88,6086 cm atau 84,7083 cm pada jarak riil dengan nilai *error* sebesar 4,6045%. Nilai *error* terkecil objek kedua didapat pada objek dengan posisi 1,2 dengan jarak terukur 217,1941 cm atau 216,8047 cm pada jarak riil dengan nilai *error* sebesar 0,1796%. Nilai *error* terbesar objek kedua didapat pada objek dengan posisi 2,2 dengan jarak terukur 256,0537 cm atau 254,7303 cm pada jarak riil dengan nilai *error* sebesar 0,5196%. Nilai *error* terkecil objek ketiga didapat pada objek dengan posisi 4,4 dengan jarak terukur 420,4720 cm atau 420,2710 cm pada jarak riil dengan nilai *error* sebesar 0,0478%. Nilai *error* terbesar objek ketiga didapat pada objek dengan posisi 0,0 dengan jarak terukur 88,6086 cm atau 84,7083 cm pada jarak riil dengan nilai *error* sebesar 4,6045%.

Dari perbandingan antara jarak objek secara riil dan jarak objek terukur terhadap *error* yang dihasilkan seperti ditunjukkan pada Tabel 4.6, dapat dianalisis bahwa posisi objek berpengaruh terhadap nilai jarak yang dihasilkan. Semakin dekat objek dengan kamera maka jarak terukur yang dihasilkan semakin besar (nilai jarak terukur menjauhi jarak riil) dan semakin jauh objek dengan kamera maka jarak terukur yang dihasilkan semakin kecil (nilai jarak terukur mendekati jarak riil). Hal tersebut dikarenakan pengukuran pada sisi jarak terukur (C) yang dilakukan secara diagonal. Sehingga semakin tinggi atau semakin jauh sisi jarak terukur (C) dari sisi jarak riil (B) maka sisi jarak terukur (C) akan menghasilkan nilai jarak yang semakin besar yang menyebabkan nilai *error* yang dihasilkan semakin besar. Rata-rata *error* yang dihasilkan dari pengukuran jarak objek tiga orang yaitu sebesar 1,54% untuk objek pertama, 0,36% untuk objek kedua, dan 1,29% untuk objek ketiga.

# BAB 5
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

1. Sistem *people counting* menggunakan deteksi wajah dengan fitur *Haar-like* dan KLT *tracker* yang dilakukan pada penelitian ini mampu mendeteksi wajah objek yang terpantau langsung oleh kamera CCTV. Wajah objek yang terdeteksi kemudian terdapat *bounding box* sebagai tanda objek tersebut telah terhitung sebagai objek yang terdeteksi. Sistem ini dapat menghitung jumlah objek mulai dari objek berjumlah satu orang, dua orang, dan tiga orang. Rekaman video CCTV secara *offline* atau non *real-time* digunakan pada penelitian ini.
2. Hasil pengujian sistem *people counting* untuk objek berjumlah satu orang dihasilkan *error* deteksi sebesar 0 (objek dapat terdeteksi di semua titik). Kemudian untuk objek berjumlah dua orang dihasilkan *error* deteksi sebesar 0 (masing-masing objek dapat terdeteksi di titik yang ditentukan). Kemudian untuk objek berjumlah tiga orang terdapat *error* deteksi sebesar 2 pada percobaan nomor 3 (objek terdeteksi lebih dari objek riil).
3. Hasil pengukuran koordinat objek untuk objek berjumlah satu orang dihasilkan rata-rata *error* pengukuran sebesar 0,19%. Kemudian untuk objek berjumlah dua orang dihasilkan rata-rata *error* pengukuran sebesar 1,79% untuk objek pertama dan 0,29% untuk objek kedua. Kemudian untuk objek berjumlah tiga orang dihasilkan rata-rata *error* pengukuran sebesar 1,54% untuk objek pertama, 0,36% untuk objek kedua, dan 1,29% untuk objek ketiga.

## 5.2 Saran

1. Untuk hasil yang lebih baik disarankan penempatan kamera CCTV yang tepat agar intensitas cahaya yang diterima oleh kamera CCTV dapat mendeteksi objek dengan baik.
2. Pengukuran koordinat objek terintegrasi dengan sistem *people counting*. Sehingga hasil jarak langsung diketahui dari objek yang terdeteksi oleh kamera CCTV beserta jumlah orang yang terdeteksi di dalam satu sistem yang terintegrasi.
3. Mengembangkan deteksi objek ketika wajah objek membelakangi kamera agar objek tetap dapat terdeteksi ketika wajah objek membelakangi kamera.

# DAFTAR PUSTAKA


[1] H. Li, E. C. L. Chan, X. Guo, J. Xiao, K. Wu, and L. M. Ni, "Wi-Counter: Smartphone-Based People Counter Using Crowdsourced Wi-Fi Signal Data," *IEEE Trans. Human-Machine Syst.*, vol. 45, no. 4, pp. 442–452, 2015, doi: 10.1109/THMS.2015.2401391.

[2] Z. S. Dhaief, "People Counting Technology," *Int. J. Mod. Trends Eng. Res.*, vol. 3, no. 11, pp. 210–214, 2016, doi: 10.21884/ijmter.2016.3142.abprk.

[3] D. Lefloch, F. A. Cheikh, J. Y. Hardeberg, P. Gouton, and R. Picot-Clemente, "Real-Time People Counting System Using A Single Video Camera," *Real-Time Image Process. 2008*, vol. 6811, no. June 2014, p. 681109, 2008, doi: 10.1117/12.766499.

[4] B. Yuliandra and T. Agung Budi, "People Counting Menggunakan Extended CAMSHIFT dan Fitur Haar-like," pp. 1–10.

[5] P. Sharma, P. M. Kokare, and M. H. Kolekar, "Performance Comparison of KLT and CAMSHIFT Algorithms for Video Object Tracking," *Recent Trends Commun. Comput. Electron.*, vol. 524, pp. 323–331, 2019, doi: 10.1007/978-981-13-2685-1.

[6] E. S. Wahyuni, R. R. Alinra, and H. Setiawan, "People Counting for Indoor Monitoring," *Int. Conf. Comput. Eng. Des.*, 2017, doi: https://doi.org/10.1109/CED.2017.8308112.

[7] A. Mohamed, A. Issam, B. Mohamed, and B. Abdellatif, "Real-Time Detection of Vehicles Using The Haar-Like Features and Artificial Neuron Networks," *Procedia Comput. Sci.*, vol. 73, no. Awict, pp. 24–31, 2015, doi: 10.1016/j.procs.2015.12.044.

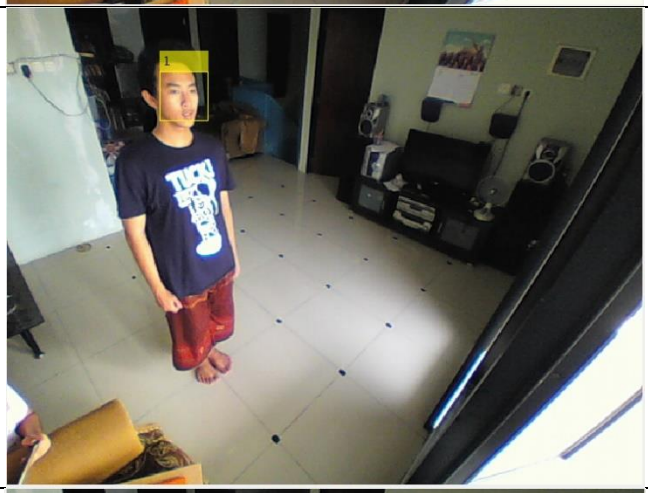
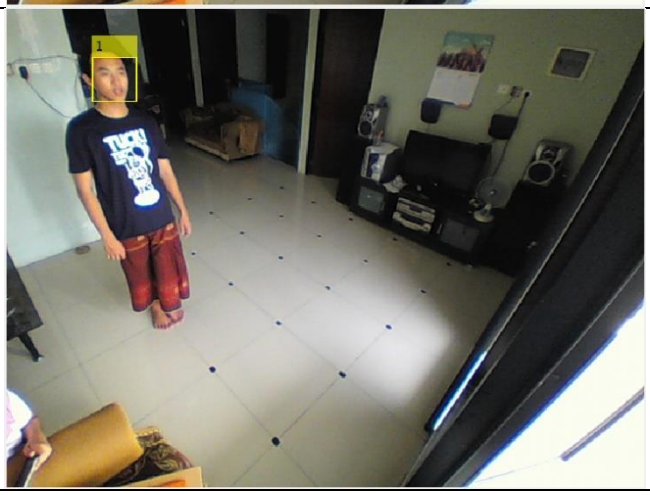
[8] S. Chau, J. Banjarnahor, D. Irfansyah, S. Kumala, and J. Banjarnahor, "Analisis Pendeteksian Pola Wajah Menggunakan Metode Haar-Like Feature," *J. Informatics Telecommun. Eng.*, vol. 2, no. 2, p. 69, 2019, doi: 10.31289/jite.v2i2.2133.

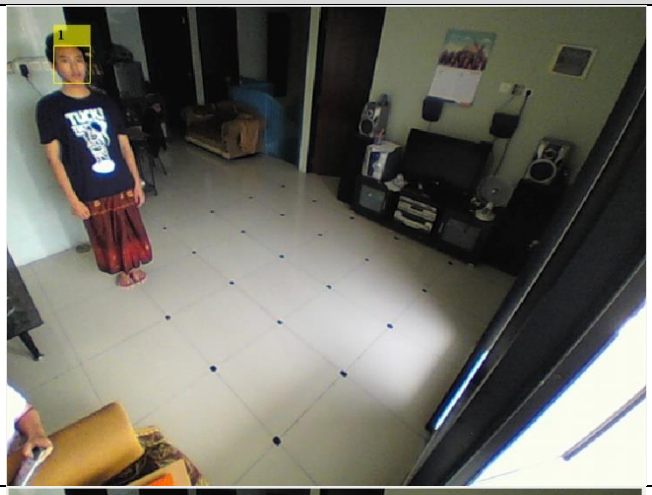
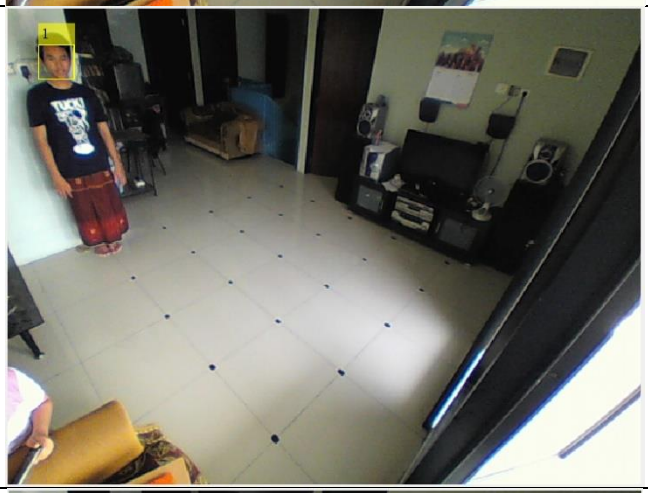
[9] C. C. Hsu, M. C. Lu, W. Y. Wang, and Y. Y. Lu, "Distance Measurement Based on Pixel Variation of CCD Images," *ISA Trans.*, vol. 48, no. 4, pp. 389–395, 2009, doi: 10.1016/j.isatra.2009.05.005.

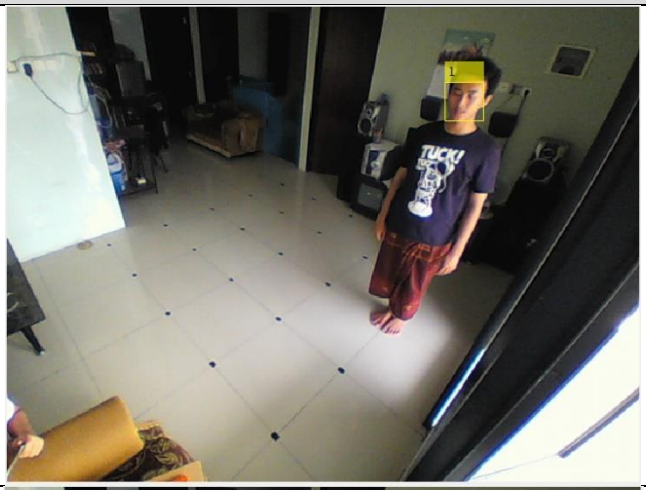
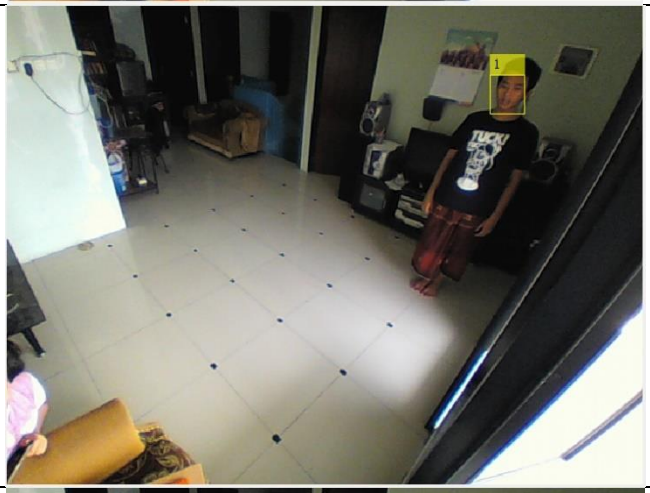
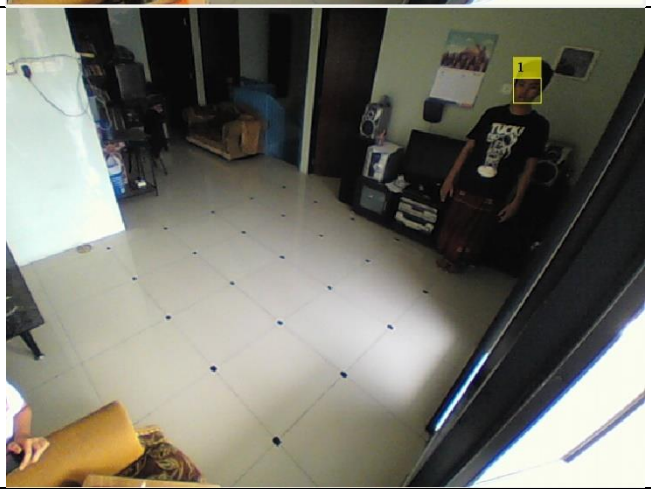
[10] P. Devireddy, "Persons Counting by Head Detection in Real Time," *GitHub*, 2019. [Online]. Available: https://github.com/Pramod-Devireddy/head_detection. [Accessed: 27-Mar-2020].

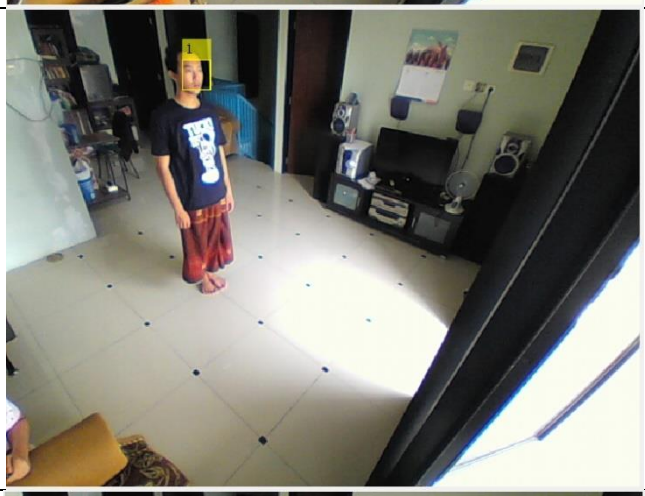
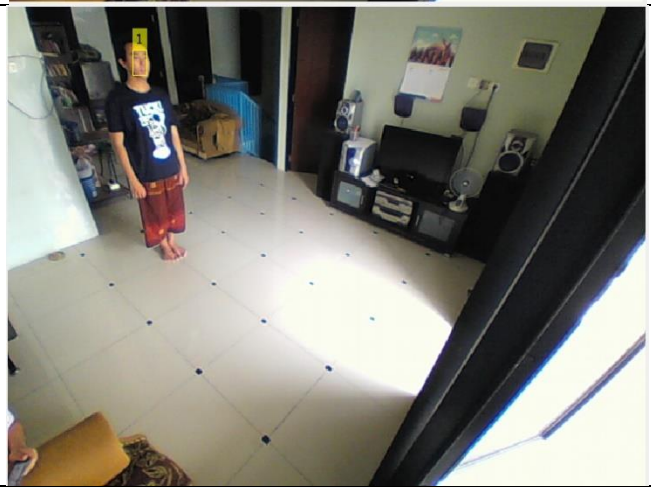
# LAMPIRAN

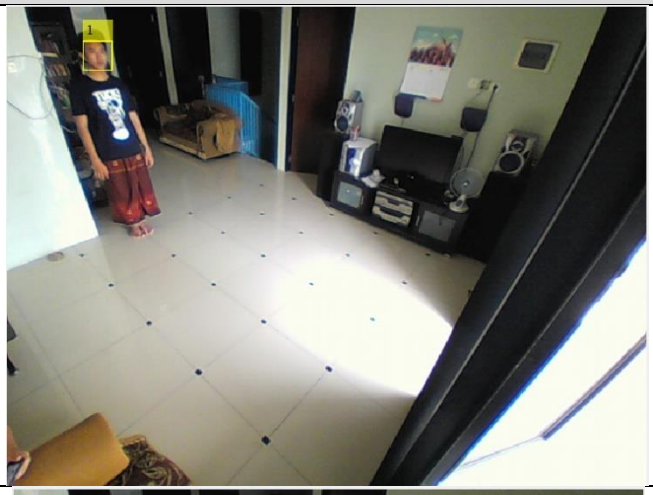
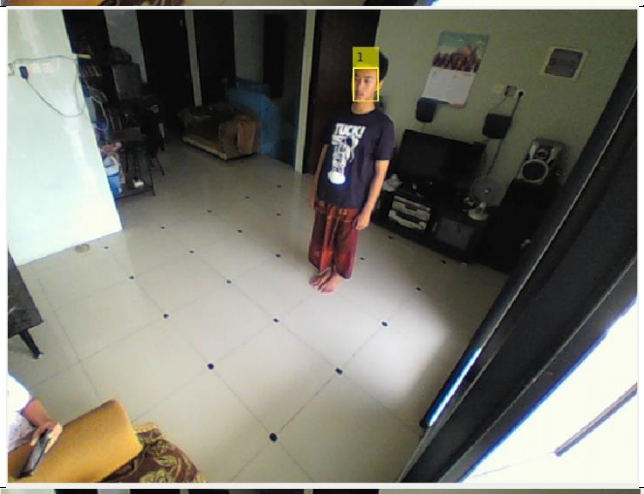
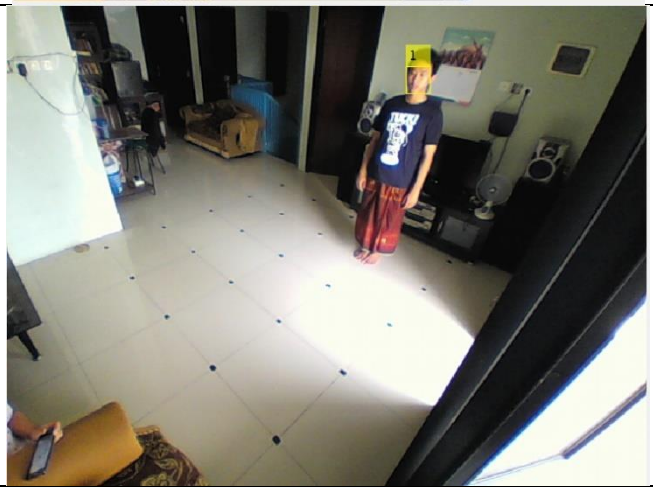
1. Pengujian Objek Satu Orang

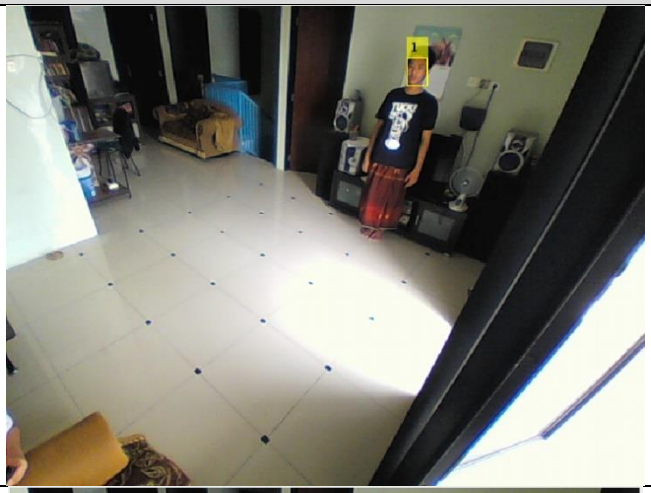
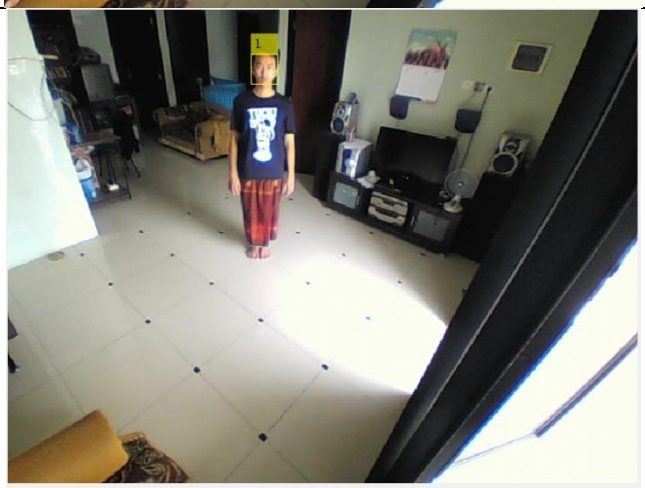
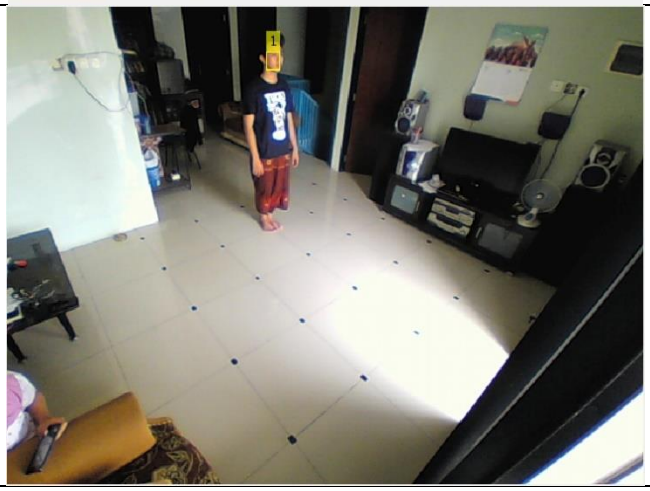
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 1 |  | 0,0 | 1 | 1 |
| 2 |  | 0,1 | 1 | 1 |
| 3 |  | 0,2 | 1 | 1 |

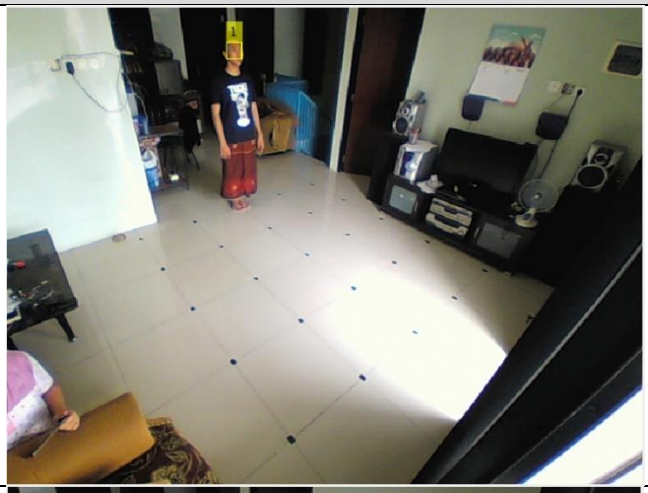
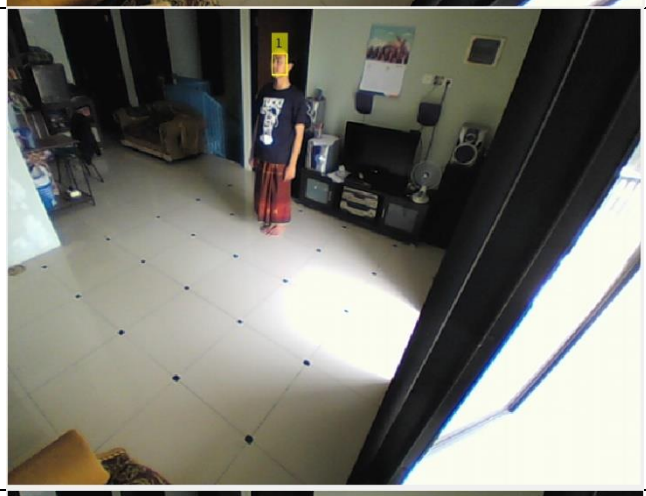
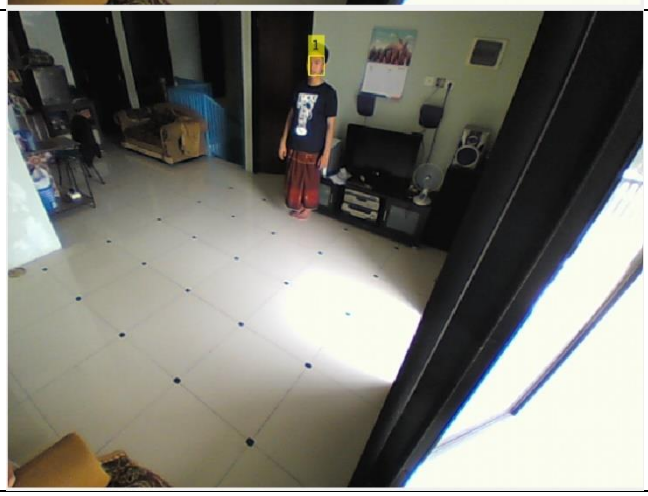
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 4 |  | 0,3 | 1 | 1 |
| 5 |  | 0,4 | 1 | 1 |
| 6 |  | 1,0 | 1 | 1 |

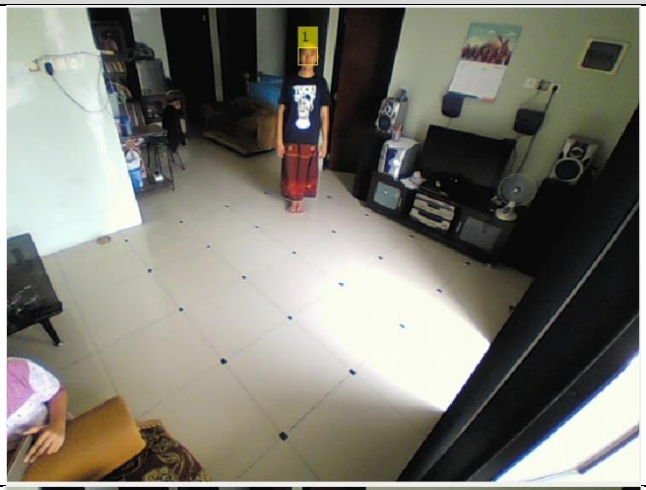
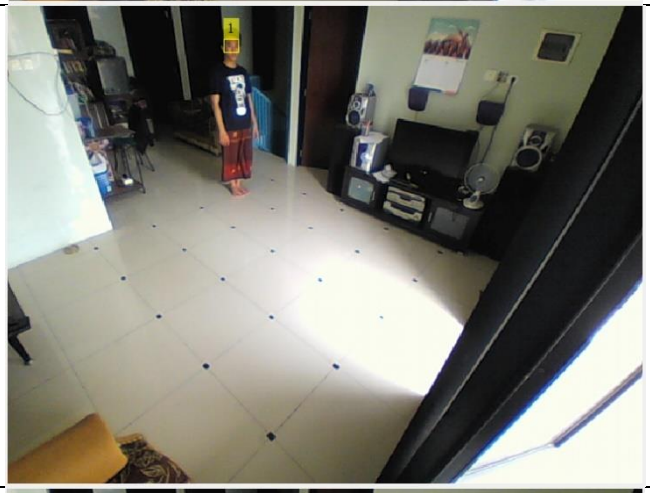
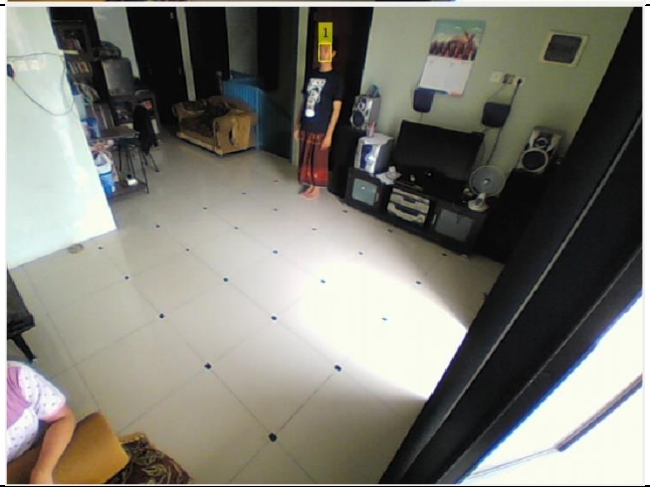
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 7 |  | 2,0 | 1 | 1 |
| 8 |  | 3,0 | 1 | 1 |
| 9 |  | 4,0 | 1 | 1 |

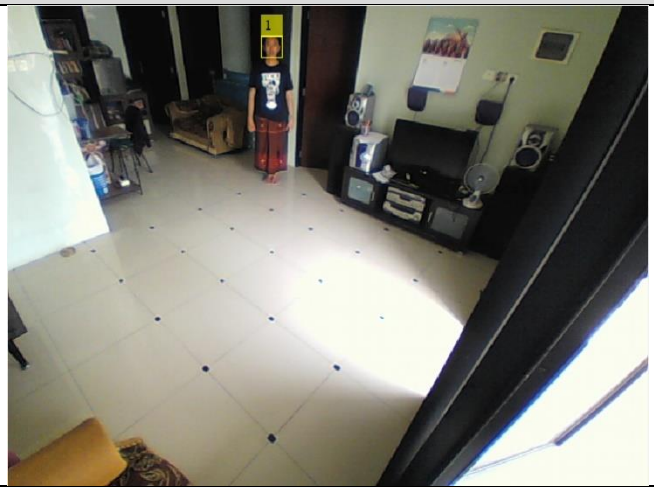
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 10 |  | 1,1 | 1 | 1 |
| 11 |  | 1,2 | 1 | 1 |
| 12 |  | 1,3 | 1 | 1 |

| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 13 |  | 1,4 | 1 | 1 |
| 14 |  | 2,1 | 1 | 1 |
| 15 |  | 3,1 | 1 | 1 |

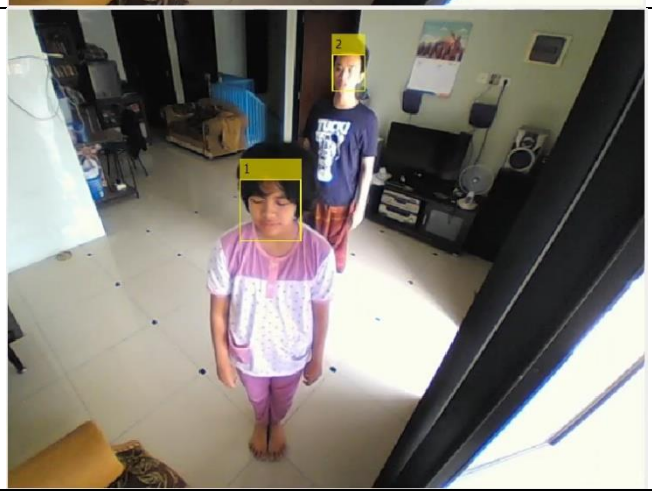
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 16 |  | 4,1 | 1 | 1 |
| 17 |  | 2,2 | 1 | 1 |
| 18 |  | 2,3 | 1 | 1 |

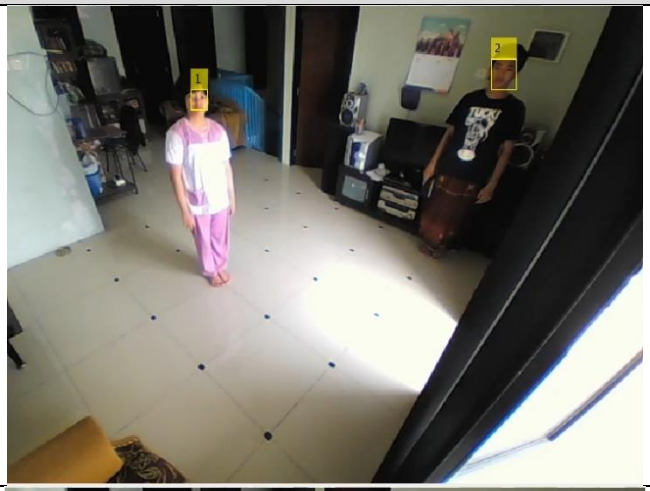
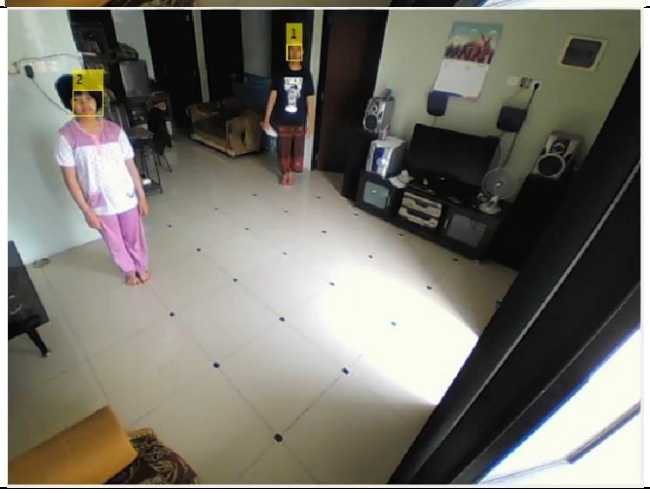
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 19 |  | 2,4 | 1 | 1 |
| 20 |  | 3,2 | 1 | 1 |
| 21 |  | 4,2 | 1 | 1 |

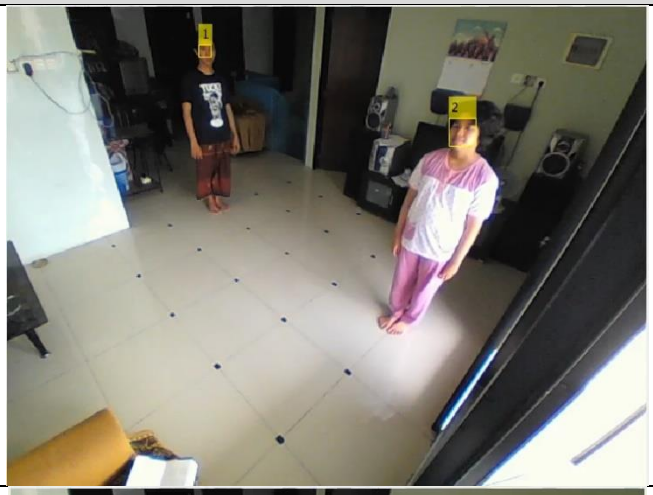
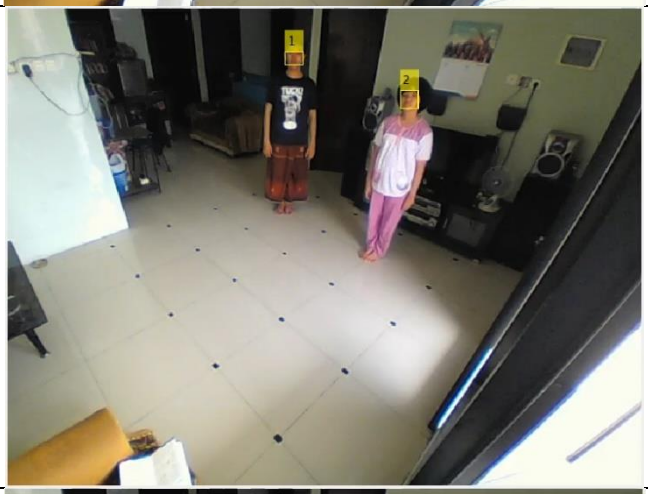
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 22 |  | 3,3 | 1 | 1 |
| 23 |  | 3,4 | 1 | 1 |
| 24 |  | 4,3 | 1 | 1 |

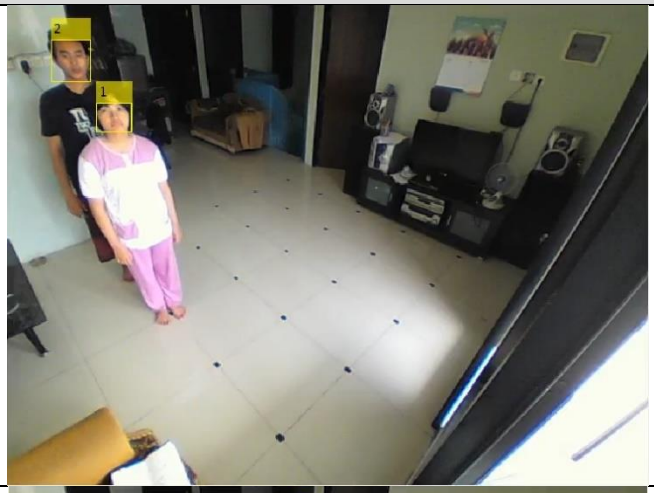
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 25 |  | 4,4 | 1 | 1 |

2. Pengujian Objek Dua Orang

| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 1 |  | Objek 1: 0,0<br>Objek 2: 1,1 | 2 | 2 |
| 2 |  | Objek 1: 0,0<br>Objek 2: 2,1 | 2 | 2 |

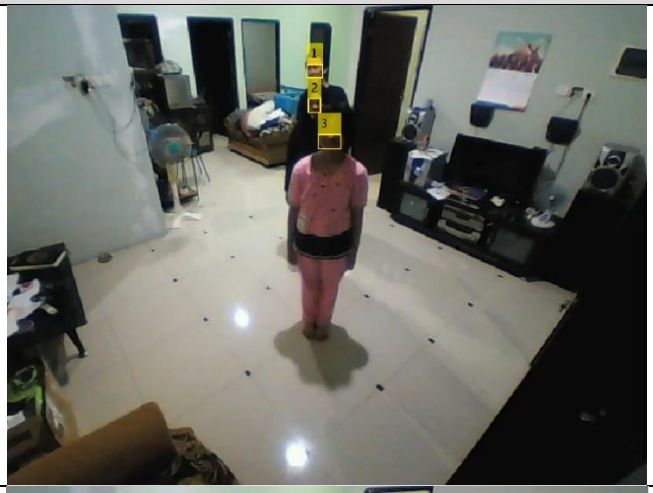
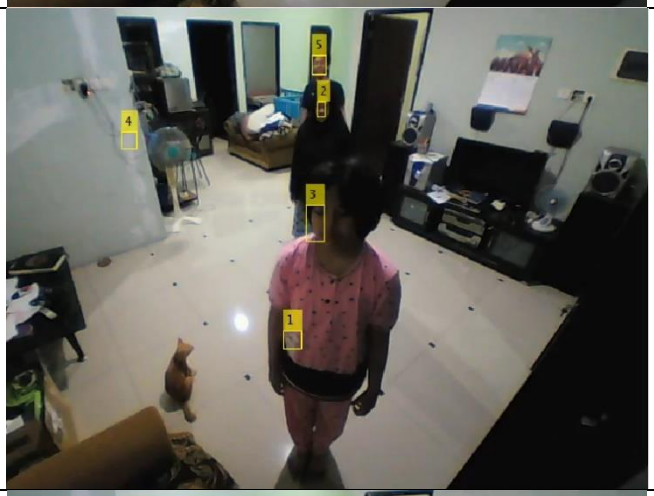
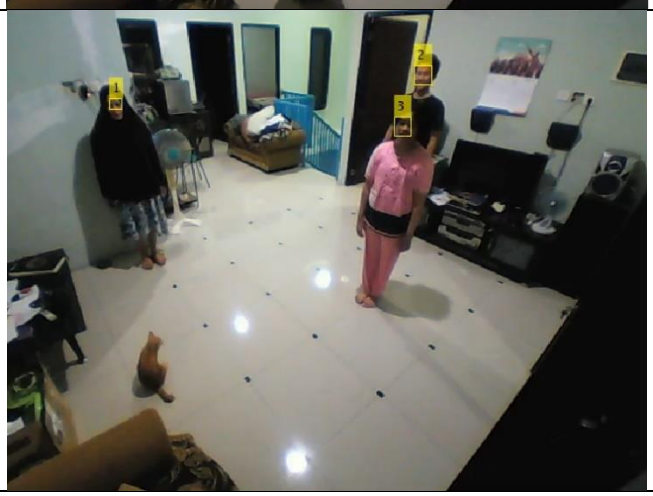
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 3 |  | Objek 1: 1,2<br>Objek 2: 4,0 | 2 | 2 |
| 4 |  | Objek 1: 4,1<br>Objek 2: 0,4 | 2 | 2 |
| 5 |  | Objek 1: 4,4<br>Objek 2: 0,3 | 2 | 2 |
| 6 | | Objek 1: 2,4 | 2 | 2 |

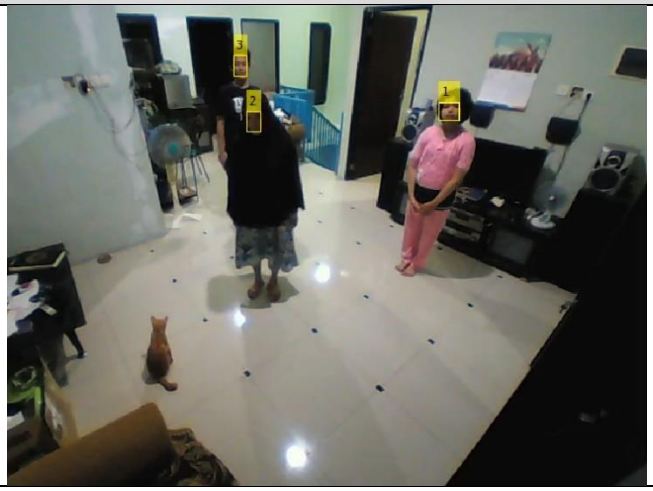
| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| |  | Objek 2: 2,0 | | |
| 7 |  | Objek 1: 3,3<br>Objek 2: 3,1 | 2 | 2 |
| 8 |  | Objek 1: 1,0<br>Objek 2: 1,3 | 2 | 2 |
| 9 | | Objek 1: 0,2 | 2 | 2 |

| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| |  | Objek 2: 0,3 | | |
| 10 |  | Objek 1: 0,0<br>Objek 2: 4,4 | 2 | 2 |

3. Pengujian Objek Tiga Orang

| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 1 |  | Objek 1: 0,1<br>Objek 2: 0,2<br>Objek 3: 0,0 | 3 | 3 |

| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| 2 |  | Objek 1: 3,3<br>Objek 2: 2,2<br>Objek 3: 1,1 | 3 | 3 |
| 3 |  | Objek 1: 0,0<br>Objek 2: 2,2<br>Objek 3: 4,4 | 3 | 5 |
| 4 |  | Objek 1: 2,1<br>Objek 2: 4,2<br>Objek 3: 0,4 | 3 | 3 |
| 5 | | Objek 1: 3,1<br>Objek 2: 1,2 | 3 | 3 |

| No | Hasil Visualisasi *People Counting* | Posisi (x,y) | Jumlah Objek Riil | Jumlah Objek Terdeteksi |
|---|---|---|---|---|
| |  | Objek 3: 2,4 | | |

4. Program utama deteksi objek

```
function varargout = HeadCount(varargin)
% HEADCOUNT MATLAB code for HeadCount.fig
%   HEADCOUNT, by itself, creates a new HEADCOUNT or raises the existing
%   singleton*.

%   H = HEADCOUNT returns the handle to a new HEADCOUNT or the handle
%   to the existing singleton*.

%   HEADCOUNT('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in HEADCOUNT.M with the given input arguments.

%   HEADCOUNT('Property','Value',...) creates a new HEADCOUNT or raises the
%   existing singleton*.  Starting from the left, property value pairs are
%   applied to the GUI before HeadCount_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to HeadCount_OpeningFcn via varargin.

%   *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%   instance to run (singleton)".

% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help HeadCount
% Last Modified by GUIDE v2.5 22-Feb-2016 00:13:48
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @HeadCount_OpeningFcn, ...
                   'gui_OutputFcn',  @HeadCount_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

% --- Executes just before HeadCount is made visible.
function HeadCount_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to HeadCount (see VARARGIN)

% Choose default command line output for HeadCount
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes HeadCount wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = HeadCount_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

%% Tombol start untuk memulai program
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global stop
stop = false;
vidObj = webcam();
faceDetector = vision.CascadeObjectDetector('Haar.xml','MergeThreshold',5); %
detektor wajah dengan Haar
tracker = MultiObjectTrackerKLT; % fungsi tracking wajah dengan KLT

%% Pre-processing ke Grayscale dan bounding box wajah terdeteksi
bboxes = [];
while isempty(bboxes)
    pause(1);
    framergb = snapshot(vidObj);
    %frame = flip(rgb2gray(framergb),2);
    frame = rgb2gray(framergb);
    bboxes = 1*faceDetector.step(imresize(frame, 1));
    axes(handles.axes1);
end
tracker.addDetections(frame, bboxes);

%% Looping program hingga program berhenti
frameNumber = 0;
keepRunning = true;
while keepRunning
    if stop == false
    %framergb = flip(snapshot(vidObj),2);
    framergb = snapshot(vidObj);
    frame = rgb2gray(framergb);
    if mod(frameNumber, 5) == 0
        bboxes = 1*faceDetector.step(imresize(frame, 1));
```

```
        if ~isempty(bboxes)
            tracker.addDetections(frame, bboxes);
        end
    else
        % Track faces
        tracker.track(frame);
    end
    % menampilkan wajah terdeteksi dan terhitung dengan bounding box dan
tracking wajah
    set(handles.text2,'String',num2str(tracker.NextId-1));
    displayFrame = insertObjectAnnotation(framergb, 'rectangle',...
        tracker.Bboxes, tracker.BoxIds);
    imshow(displayFrame);
    frameNumber = frameNumber + 1;
    else
%         clear vidObj;
        return;
    end
end

%% Tombol stop untuk memberhentikan program
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global stop
stop = true;
```

5. Program fungsi pelacakan wajah KLT *tracker*

```
% MultiObjectTrackerKLT implements tracking multiple objects using the
% Kanade-Lucas-Tomasi (KLT) algorithm.
% tracker = MultiObjectTrackerKLT() creates the multiple object tracker.
%
% MultiObjectTrackerKLT properties:
%   PointTracker - a vision.PointTracker object
%   Bboxes       - object bounding boxes
%   BoxIds       - ids associated with each bounding box
%   Points       - tracked points from all objects
%   PointIds     - ids associated with each point
%   NextId       - the next object will have this id
%   BoxScores    - and indicator of whether or not an object is lost
%
% MultiObjectTrackerKLT methods:
%   addDetections - add detected bounding boxes
%   track         - track the objects

% Copyright 2013-2014 The MathWorks, Inc

classdef MultiObjectTrackerKLT < handle
    properties
        % PointTracker A vision.PointTracker object
        PointTracker;

        % Bboxes M-by-4 matrix of [x y w h] object bounding boxes
        Bboxes = [];

        % BoxIds M-by-1 array containing ids associated with each bounding
box
        BoxIds = [];
```

```
        % Points M-by-2 matrix containing tracked points from all objects
        Points = [];

        % PointIds M-by-1 array containing object id associated with each
        %   point. This array keeps track of which point belongs to which
object.
        PointIds = [];

        % NextId The next new object will have this id.
        NextId = 1;

        % BoxScores M-by-1 array. Low box score means that we probably lost
the object.
        BoxScores = [];
    end

    methods
        %-------------------------------------------------------------------
        function this = MultiObjectTrackerKLT()
        % Constructor
            this.PointTracker = ...
                vision.PointTracker('MaxBidirectionalError', 2);
        end

        %-------------------------------------------------------------------
        function addDetections(this, I, bboxes)
        % addDetections Add detected bounding boxes.
        % addDetections(tracker, I, bboxes) adds detected bounding boxes.
        % tracker is the MultiObjectTrackerKLT object, I is the current
        % frame, and bboxes is an M-by-4 array of [x y w h] bounding boxes.
        % This method determines whether a detection belongs to an existing
        % object, or whether it is a brand new object.
            for i = 1:size(bboxes, 1)
                % Determine if the detection belongs to one of the existing
                % objects.
                boxIdx = this.findMatchingBox(bboxes(i, :));

                if isempty(boxIdx)
                    % This is a brand new object.
                    this.Bboxes = [this.Bboxes; bboxes(i, :)];
                    points = detectMinEigenFeatures(I, 'ROI', bboxes(i, :));
                    points = points.Location;
                    this.BoxIds(end+1) = this.NextId;
                    idx = ones(size(points, 1), 1) * this.NextId;
                    this.PointIds = [this.PointIds; idx];
                    this.NextId = this.NextId + 1;
                    this.Points = [this.Points; points];
                    this.BoxScores(end+1) = 1;

                else % The object already exists.

                    % Delete the matched box
                    currentBoxScore = this.deleteBox(boxIdx);

                    % Replace with new box
                    this.Bboxes = [this.Bboxes; bboxes(i, :)];

                    % Re-detect the points. This is how we replace the
                    % points, which invariably get lost as we track.
                    points = detectMinEigenFeatures(I, 'ROI', bboxes(i, :));
                    points = points.Location;
```

```
                    this.BoxIds(end+1) = boxIdx;
                    idx = ones(size(points, 1), 1) * boxIdx;
                    this.PointIds = [this.PointIds; idx];
                    this.Points = [this.Points; points];
                    this.BoxScores(end+1) = currentBoxScore + 1;
                end
            end

            % Determine which objects are no longer tracked.
            minBoxScore = -2;
            this.BoxScores(this.BoxScores < 3) = ...
                this.BoxScores(this.BoxScores < 3) - 0.5;
            boxesToRemoveIds = this.BoxIds(this.BoxScores < minBoxScore);
            while ~isempty(boxesToRemoveIds)
                this.deleteBox(boxesToRemoveIds(1));
                boxesToRemoveIds = this.BoxIds(this.BoxScores < minBoxScore);
            end

            % Update the point tracker.
            if this.PointTracker.isLocked()
                this.PointTracker.setPoints(this.Points);
            else
                this.PointTracker.initialize(this.Points, I);
            end
        end

        %--------------------------------------------------------------------
        function track(this, I)
        % TRACK Track the objects.
        % TRACK(tracker, I) tracks the objects into frame I. tracker is the
        % MultiObjectTrackerKLT object, I is the current video frame. This
        % method updates the points and the object bounding boxes.
            [newPoints, isFound] = this.PointTracker.step(I);
            this.Points = newPoints(isFound, :);
            this.PointIds = this.PointIds(isFound);
            generateNewBoxes(this);
            if ~isempty(this.Points)
                this.PointTracker.setPoints(this.Points);
            end
        end
    end

    methods(Access=private)
        %--------------------------------------------------------------------
        function boxIdx = findMatchingBox(this, box)
        % Determine which tracked object (if any) the new detection belongs
to.
            boxIdx = [];
            for i = 1:size(this.Bboxes, 1)
                area = rectint(this.Bboxes(i,:), box);
                if area > 0.2 * this.Bboxes(i, 3) * this.Bboxes(i, 4)
                    boxIdx = this.BoxIds(i);
                    return;
                end
            end
        end

        %--------------------------------------------------------------------
        function currentScore = deleteBox(this, boxIdx)
        % Delete object.
            this.Bboxes(this.BoxIds == boxIdx, :) = [];
            this.Points(this.PointIds == boxIdx, :) = [];
```

```
            this.PointIds(this.PointIds == boxIdx) = [];
            currentScore = this.BoxScores(this.BoxIds == boxIdx);
            this.BoxScores(this.BoxIds == boxIdx) = [];
            this.BoxIds(this.BoxIds == boxIdx) = [];
        end

        %--------------------------------------------------------------------
        function generateNewBoxes(this)
        % Get bounding boxes for each object from tracked points.
            oldBoxIds = this.BoxIds;
            oldScores = this.BoxScores;
            this.BoxIds = unique(this.PointIds);
            numBoxes = numel(this.BoxIds);
            this.Bboxes = zeros(numBoxes, 4);
            this.BoxScores = zeros(numBoxes, 1);
            for i = 1:numBoxes
                points = this.Points(this.PointIds == this.BoxIds(i), :);
                newBox = getBoundingBox(points);
                this.Bboxes(i, :) = newBox;
                this.BoxScores(i) = oldScores(oldBoxIds == this.BoxIds(i));
            end
        end
    end
end

%-------------------------------------------------------------------------
function bbox = getBoundingBox(points)
x1 = min(points(:, 1));
y1 = min(points(:, 2));
x2 = max(points(:, 1));
y2 = max(points(:, 2));
bbox = [x1 y1 x2 - x1 y2 - y1];
end
```

6. Program pengukuran jarak berbasis piksel

```
clear all; close all; clc;
%% Ukur Jarak Kamera dengan Objek (Koordinat X)
%% Jarak h1

l_satu_x    = 50;  %50 cm / 115 piksel
N_satu_x    = 51;  %piksel
D_satu_x    = 627; %278.3 cm / 640 piksel
Nmax_satu_x = 640; %piksel

x120  = l_satu_x/N_satu_x;
x_120 = D_satu_x/Nmax_satu_x;
xx120 = [x120 x_120]
%% Jarak h2

l_dua_x    = 50;  %50 cm / 191 piksel
N_dua_x    = 61;  %piksel
D_dua_x    = 525; %167.56 cm / 640 piksel
Nmax_dua_x = 640; %piksel

x60  = l_dua_x/N_dua_x;
x_60 = D_dua_x/Nmax_dua_x;
xx60 = [x60 x_60]

%% Ukur Jarak Kamera dengan Objek (Koordinat Y)
%% Jarak h1
```

```
l_satu_y    = 50;  %50 cm / 80 piksel
N_satu_y    = 51;  %piksel
D_satu_y    = 627; %400 cm / 640 piksel
Nmax_satu_y = 640; %piksel

y180  = l_satu_y/N_satu_y;
y_180 = D_satu_y/Nmax_satu_y;
yy180 = [y180 y_180]
%% Jarak h2

l_dua_y    = 50;  %50 cm / 116 piksel
N_dua_y    = 61;  %piksel
D_dua_y    = 525; %275.86 cm / 640 piksel
Nmax_dua_y = 640; %piksel

y120  = l_dua_y/N_dua_y;
y_120 = D_dua_y/Nmax_dua_y;
yy120 = [y120 y_120]

%% Jarak riil dan piksel
%% Koordinat X

c_x  = D_satu_x/D_dua_x;
d_x  = N_dua_x/N_satu_x;
cd_x = [c_x d_x]
%% Koordinat Y

c_y  = D_satu_y/D_dua_y;
d_y  = N_dua_y/N_satu_y;
cd_y = [c_y d_y]

%% Jarak referensi
%% Koordinat X

hsatu_x = 300; %120 cm
hdua_x  = 240; %60 cm

delta_h_x  = hsatu_x - hdua_x %hasil berupa satuan cm
delta_hh_x = hdua_x - hsatu_x %hasil berupa satuan cm
%% Koordinat Y

hsatu_y = 300; %180 cm
hdua_y  = 240; %120 cm

delta_h_y  = hsatu_y - hdua_y %hasil berupa satuan cm
delta_hh_y = hdua_y - hsatu_y %hasil berupa satuan cm

%% Nilai hs (Jarak dari Optical Origin (OP) hingga ujung kamera/lensa)
%% Koordinat X

hm_satu_x = hsatu_x;
hm_dua_x  = hdua_x;
Dm_satu_x = D_satu_x;
Dm_dua_x  = D_dua_x;

hs_x = (hm_satu_x*Dm_dua_x - hm_dua_x*Dm_satu_x)/(Dm_satu_x - Dm_dua_x)
%% Koordinat Y

hm_satu_y = hsatu_y;
```

```
hm_dua_y  = hdua_y;
Dm_satu_y = D_satu_y;
Dm_dua_y  = D_dua_y;

hs_y = (hm_satu_y*Dm_dua_y - hm_dua_y*Dm_satu_y)/(Dm_satu_y - Dm_dua_y)

%% Jarak terukur
%% Koordinat X

h_satu_x        = (N_dua_x/(N_dua_x - N_satu_x))*delta_h_x - hs_x
h_dua_x         = (N_satu_x/(N_dua_x - N_satu_x))*delta_h_x - hs_x
delta_h_baru_x  = h_satu_x - h_dua_x
delta_hh_baru_x = h_dua_x - h_satu_x
%% Koordinat Y

h_satu_y        = (N_dua_y/(N_dua_y - N_satu_y))*delta_h_y - hs_y
h_dua_y         = (N_satu_y/(N_dua_y - N_satu_y))*delta_h_y - hs_y
delta_h_baru_y  = h_satu_y - h_dua_y
delta_hh_baru_y = h_dua_y - h_satu_y

%% Error dan RMSE jarak
%% Koordinat X

error_satu_x     = abs(((h_satu_x - hsatu_x)/hsatu_x)*100); %hasil dalam
satuan persen
error_dua_x      = abs(((h_dua_x - hdua_x)/hdua_x)*100);    %hasil dalam
satuan persen
error_satu_dua_x = [error_satu_x error_dua_x]

RMSE_satu_x     = sqrt(mean((hsatu_x - h_satu_x).^2));
RMSE_dua_x      = sqrt(mean((hdua_x - h_dua_x).^2));
RMSE_satu_dua_x = [RMSE_satu_x RMSE_dua_x]
%% Koordinat Y

error_satu_y     = abs(((h_satu_y - hsatu_y)/hsatu_y)*100); %hasil dalam
satuan persen
error_dua_y      = abs(((h_dua_y - hdua_y)/hdua_y)*100);    %hasil dalam
satuan persen
error_satu_dua_y = [error_satu_y error_dua_y]

RMSE_satu_y     = sqrt(mean((hsatu_y - h_satu_y).^2));
RMSE_dua_y      = sqrt(mean((hdua_y - h_dua_y).^2));
RMSE_satu_dua_y = [RMSE_satu_y RMSE_dua_y]

%% Ukur Jarak Piksel dengan titik referensi Pa dan Pb untuk mengetahui jumlah
piksel N_satu dan N_dua
%% Koordinat X

ML_satu_x = 257; %piksel
MR_satu_x = 308; %piksel

ML_dua_x  = 252; %piksel
MR_dua_x  = 313; %piksel

Nh_satu_x = MR_satu_x - ML_satu_x %hasil dalam satuan piksel
Nh_dua_x  = MR_dua_x - ML_dua_x   %hasil dalam satuan piksel
delta_n_x = Nh_dua_x - Nh_satu_x  %hasil dalam satuan piksel
%% Koordinat Y

ML_satu_y = 252; %piksel
```

```
MR_satu_y = 303; %piksel

ML_dua_y  = 247; %piksel
MR_dua_y  = 308; %piksel

Nh_satu_y = MR_satu_y - ML_satu_y %hasil dalam satuan piksel
Nh_dua_y  = MR_dua_y - ML_dua_y   %hasil dalam satuan piksel
delta_n_y = Nh_dua_y - Nh_satu_y  %hasil dalam satuan piksel
```