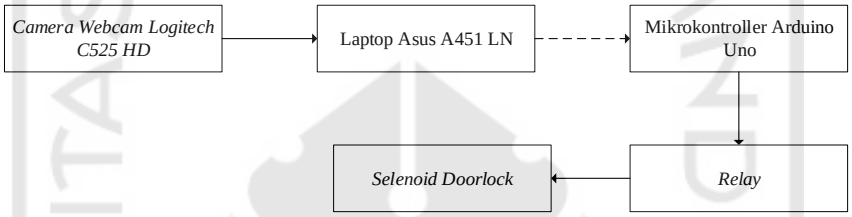
BAB 3
METODOLOGI

3.1 Perancangan Perangkat Keras

Perancangan sistem ini bertujuan menggunakan pengenalan wajah untuk keamanan rumah. Pintu rumah dapat terbuka jika wajah terdeteksi sebagai anggota penghuni rumah. Sistem keamanan rumah akan disimulasikan menggunakan *LabVIEW 2017* dan mikrokontroller Arduino Uno. Aktuator pembuka kunci rumah akan disimulasikan menggunakan *Selenoid Doorlock* 12 Volt DC. Diagram blok dari perancangan perangkat keras seperti pada Gambar 3.1 berikut:

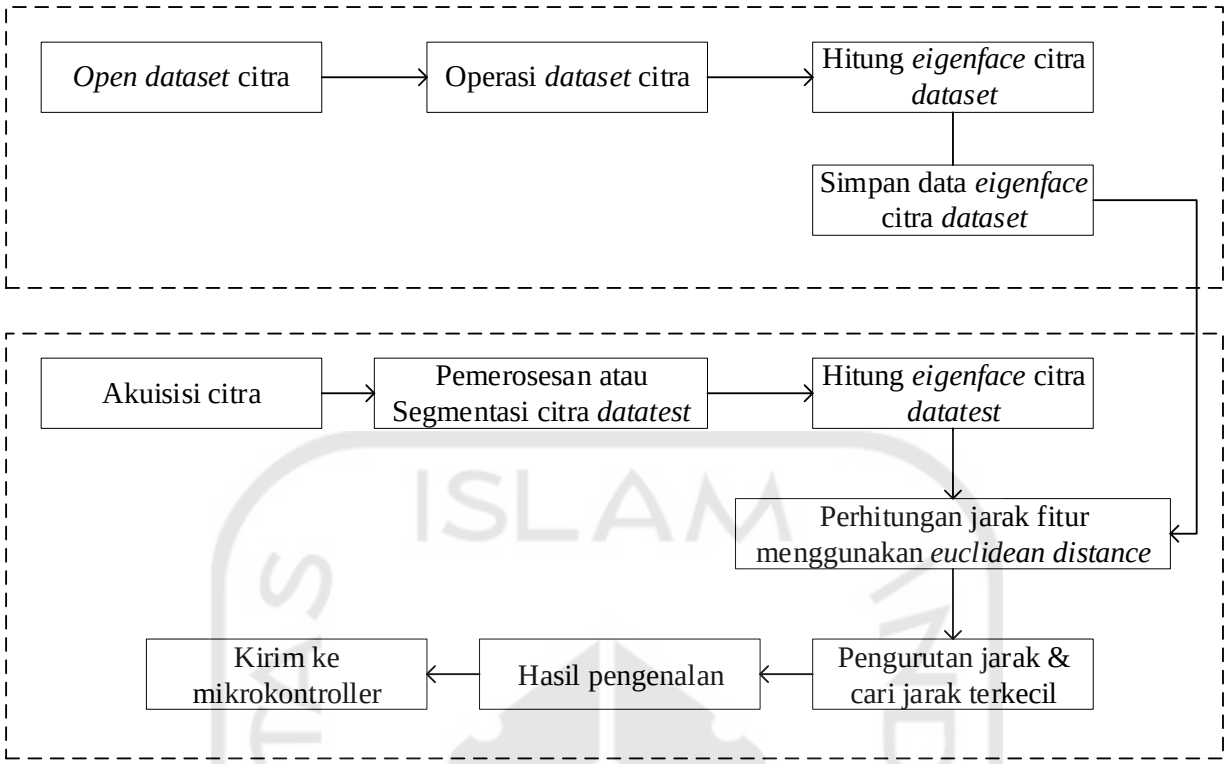


Gambar 3.1 Blok diagram perangkat keras

Kamera yang digunakan adalah *Logitech C525 portable HD Webcam 720p 30fps 8MP* (*Mega Pixel*). Kamera tersebut terhubung dengan laptop Asus A451 LN menggunakan perpanjangan kabel USB 1,5 meter. Pada laptop citra di proses menggunakan *LabVIEW software*. Selain itu laptop berfungsi untuk memonitor proses dan mengubah parameter yang dibutuhkan. Mikrokontroller Arduino Uno menerima data dalam bentuk data serial atau *string* dari *LabVIEW software*. Data dikirim menggunakan *Bluetooth* bawaan laptop yaitu *Qualcomm Atheros* AR3012 *Communications*. Sedangkan pada mikrokontroller Arduino Uno data diterima oleh *Bluetooth HC 05*. Data serial yang dikirim dari LabVIEW digunakan untuk menentukan kondisi *output port* digital Arduino Uno yang terhubung ke *relay*. Kondisi *port* digital Arduino Uno tersebut bisa *High* atau *Low*. *Relay* berfungsi sebagai saklar untuk *Selenoid Doorlock*. Kondisi *relay ON* menandakan bahwa *supply* tegangan 12 VDC untuk *Selenoid Doorlock* terhubung. Sedangkan kondisi *relay OFF* menandakan *supply* tegangan 12 VDC untuk *Selenoid Doorlock* tidak terhubung.

3.2 Perancangan Perangkat Lunak

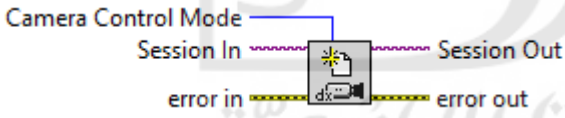
Secara keseluruhan proses pengenalan wajah menggunakan metode *eigenface* pada LabVIEW digambarkan sebagai berikut:



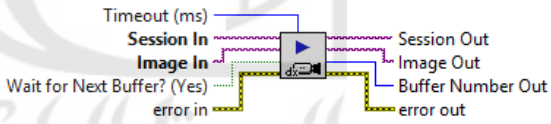
Gambar 3.2 Blok diagram perancangan perangkat lunak

3.3 Akuisisi Citra

Pada *LabVIEW* proses akuisisi citra menggunakan blok diagram *IMAQdx Open Camera.vi.* Blok tersebut berfungsi untuk mengkases kamera yang terhubung ke laptop. Kamera yang diakses bisa kamera internal laptop atau ekternal laptop. Blok diagram *IMAQdx Open Camera* kemudian dihubungkan dengan blok diagram *IMAQdx Configure Grab*. Fungsi dari blok tersebut untuk mengatur resolusi kamera.



Gambar 3.3 *IMAQdx Open Camera.vi*



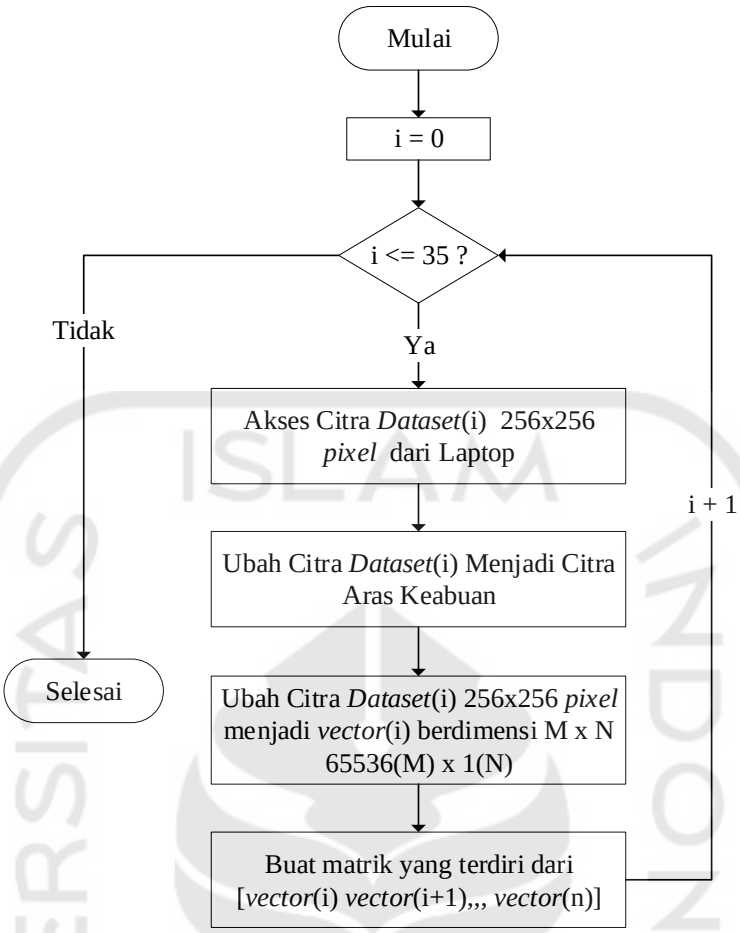
Gambar 3.4 *IMAQdx Grab.vi*

Gambar 3.5 *IMAQdx Configure Grab.vi*

Gambar 3.6 *IMAQdx Close Camera.vi*

3.4 Membuka Dan Mengoperasikan Citra *Dataset*

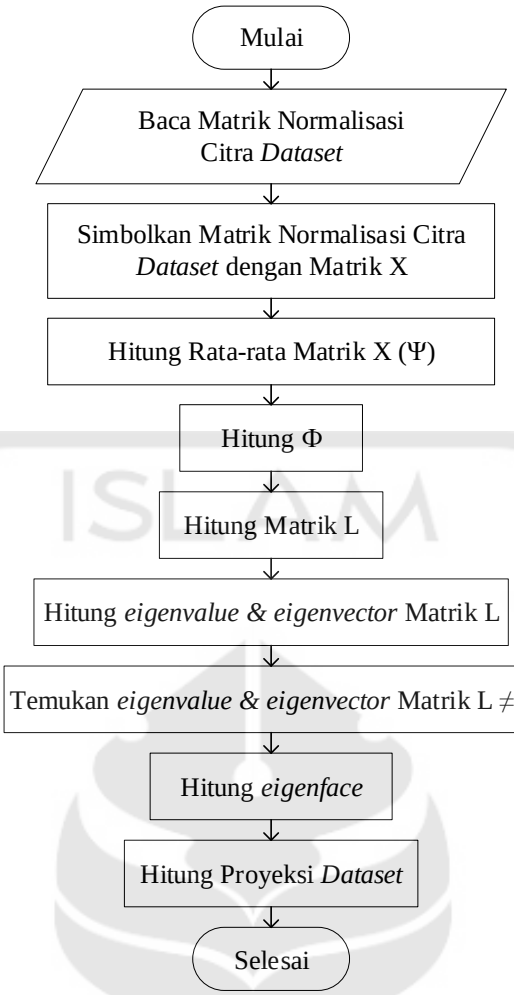
Berdasarkan Gambar 3.2 proses yang terjadi pada kondisi *open dataset* citra digambarkan dengan *flowchart* pada Gambar 3.7 berikut:



Gambar 3.7 *Flowchart* proses *open dataset* citra

Berdasarkan Gambar 3.7 diatas jumlah *dataset* yang digunakan berjumlah 36 buah citra. Masing-masing citra *dataset* memiliki resolusi 256x256 *pixel* dengan format .jpg. Citra *dataset* tersebut berisi citra dari orang-orang yang di ijinkan untuk masuk ke kamar. Proses pengambilan citra *dataset* menggunakan kamera *smartphone*. Kemudian resolusinya diubah dari *default* bawaan *smartphone* menjadi resolusi 256x256 *pixel* menggunakan *software photoshop*. Setelah di *edit* menggunakan *software* photosohop citra *dataset* disimpan dalam format .jpg dan RGB. Pada proses ini dilakukan perubahan citra dari citra RGB menjadi citra aras keabuan *(grayscale)*. Sehingga nilai dari masing-masing titik pada tiap *pixel* nya memiliki nilai 0 sampai 255 saja. Kemudian citra *dataset* tersebut diubah menjadi *vector* yang memiliki dimensi M = 65536 dan N = 1. Kemudian masing-masing *vector* tersebut dihimpun menjadi sebuah matrik. Matrik tersebut dalam penelitian ini diberi nama matrik Normalisasi Citra *Dataset*.

Berdasarkan Gambar 3.2 proses yang terjadi pada kondisi operasi *dataset* citra adalah proses pembelajaran citra *dataset*. Pelajari citra *dataset* adalah proses mengubah 36 citra *dataset* ke ruang *dataset* menggunakan algoritma *eigenface*. *Flowchart* dari proses tersebut teradapat pada Gambar 3.8 sebagai berikut:



Gambar 3.8 *Flowchart* proses operasi citra *dataset*

Matrik X adalah matrik yang berasal dari himpunan *training set* citra wajah ($\Gamma_1\ \Gamma_2\ \Gamma_3\ \Gamma_4\ \Gamma_5, \ldots\ldots\ldots \Gamma_m$) yang memiliki dimensi M = 65536 dan N = 1. Matrik X memiliki dimensi 65536 baris dan 36 kolom. Kemudian *average face vector* (Ψ) memiliki 65536 elemen. Persamaan yang digunakan untuk mendapatkan *average face vector* adalah Persamaan (2.1).

Nilai selisih antara matrik X dan *average face vector* (Ψ) atau (Φ) dinyatakan dalam sebuah matrik A yang memiliki dimensi 65536 baris dan 36 kolom. Persamaan yang digunakan untuk mendapatkan matrik A adalah Persamaan (2.2).

Matrik L diperoleh dengan mengalikan matrik A *transpose* dengan matrik A. matrik A memiliki dimensi 65536 baris dan 36 kolom. Sedangkan matrik A *transpose* memiliki dimensi 36 baris dan 65536 kolom. Maka dimensi untuk matrik L adalah 36 baris dan 36 kolom sesuai Persamaan (2.4).

Kemudian untuk menemukan *eigenvalue* dan *eigenvector* dari matrik L menggunakan *code mathscript eig(L)* pada *LabVIEW 2017*. Sintak *eig(L)* memiliki arti mencari nilai *eigenvalue* dan e*igenvector* dari matrik L. Kemudian *eigenvector* yang digunakan adalah *eigenvector* yang memiliki *eigenvalue* tidak sama dengan nol.

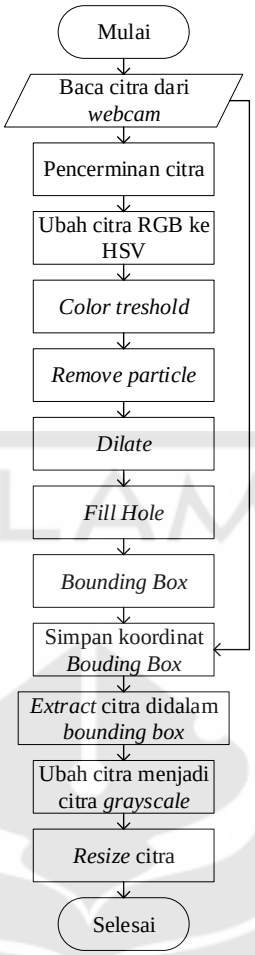
Eigenface dihitung dengan mengalikan matrik A dengan *eigenvector*. Jika matrik A memiliki dimensi 65536 x 36 dan *eigenvector* yang baru memiliki dimensi 36 x 36 maka matrik *eigenface* memiliki dimensi 65536 x 36. Persamaan yang digunakan untuk mencari matrik *eigenface* adalah Persamaan (2.6). Kemudian mencari hasil proyeksi citra *dataset* menggunakan Persamaan (2.10).

3.5 Pemerosesan Citra

Pemerosesan citra atau proses segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra. Segmentasi citra membagi citra ke dalam beberapa daerah dengan setiap objek memiliki kemiripan atribut. Pada citra yang mengandung satu objek, objek dibedakan dari latar belakangnya. Segmentasi citra bertujuan menghasilkan citra biner dengan menggunakan *thresholding* untuk mengubah citra menjadi citra biner. Proses binerisasi menggunakan nilai batas (*thresholding)* untuk dapat mengubah nilai *pixel* menjadi warna hitam atau merah[2]. Pada penelitian ini proses *face detection* dapat dikatakan sebagai proses segmentasi citra. Contoh gambar sebuah citra sebelum dan sesudah di *threshold* terdapat pada Gambar 3.10 dan Gambar 3.11.

3.6 Face Detection

Face detection adalah proses mengambil bagian wajah sampai leher saja dari keseluruhan *frame* citra. Proses *face detection* harus dilakukan sebelum proses membandingkan citra *datatest* dengan citra *dataset*. *Flowchart* proses *face detection* ditunjukan Gambar 3.9 berikut.

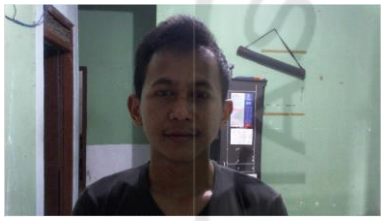


Gambar 3.9 *Flowchart face detection*

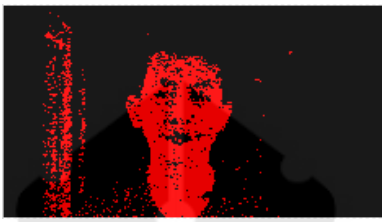
Proses *color threshold* bertujuan untuk mendapatkan bagian wajah sampai leher objek. *Color threshold* dilakukan dengan cara mengatur parameter *hue* minimum, *hue* maksimum, *saturation* minimum, *saturation* maksimum, *value* minimum dan *value* maksimum dari blok *IMAQ ColorTreshold VI* pada LabVIEW. Nilai parameter terbaik untuk masing-masing parameter didapatkan dari blok *IMAQ ColorHistogram VI* pada LabVIEW. Hasil dari *color threshold* ditunjukan oleh Gambar 3.11 yang mana Gambar 3.11 menunjukan bahwa proses *color threshold* meloloskan citra berwarna kulit dengan nilai biner 1. Proses *remove particle* menggunakan blok *IMAQ RemoveParticle VI* pada LabVIEW. Proses *remove particle* bertujuan untuk memperbaiki kualitas citra Gambar 3.11 sehingga didapatkan citra yang hanya terdiri dari bagian wajah sampai leher objek. Hasil dari *remove particle* ditunjukan oleh Gambar 3.12. Proses *dilate* menggunakan blok *IMAQ Morphology VI* pada LabVIEW. Proses *dilate* bertujuan untuk mendapatkan efek pelebaran terhadap *pixel* bernilai 1. Proses *fill hole* menggunakan blok *IMAQ FillHole VI* pada LabVIEW. Proses *fill hole* bertujuan menambah kualitas citra dengan cara mengisi partikel berlubang dengan nilai 1 pada Gambar 3.13. Hasil proses *fill hole* ditunjukan oleh Gambar 3.14.

Proses *bounding box* bertujuan untuk mendeteksi objek dan mengetahui koordinat objek. Proses *bounding box* menggunakan blok *IMAQ CountObjek 2 VI* pada LabVIEW. Koordinat objek

yang didapatkan digambarkan dengan *bounding box* berwarna merah disekitar objek. Hasil proses *bounding box* ditunjukan Gambar 3.15. *Bounding box* tersebut kemudian diletakan pada citra hasil pembacaan *webcam* seperti yang terdapat pada Gambar 3.18. Proses *exstract* bertujuan untuk mengambil bagian objek yang terdapat pada *bounding box*. Proses *exstract* menggunakan blok *IMAQ Exstract VI*. Hasil dari proses *exstract* ditunjukan oleh Gambar 3.16. Kemudian proses yang terakhir adalah proses *resize* citra pada Gambar 3.16 menggunakan blok *IMAQ Resample VI*. Proses *resize* bertujuan untuk mengubah resolusi citra Gambar 3.16 menjadi resolusi citra yang bersesuaian dengan citra *dataset* yaitu 256 x 256. Hasil proses *resize* ditunjukan oleh Gambar 3.17. Gambar untuk masing-masing blok yang digunakan pada proses *face detetction* terdapat pada lampiran.



Gambar 3.10 Jendela kamera



Gambar 3.11 *Color treshold*

Gambar 3.12 *Remove particle*

Gambar 3.13 *Dilate*

Gambar 3.14 *Fill hole*

Gambar 3.15 *Bounding box*

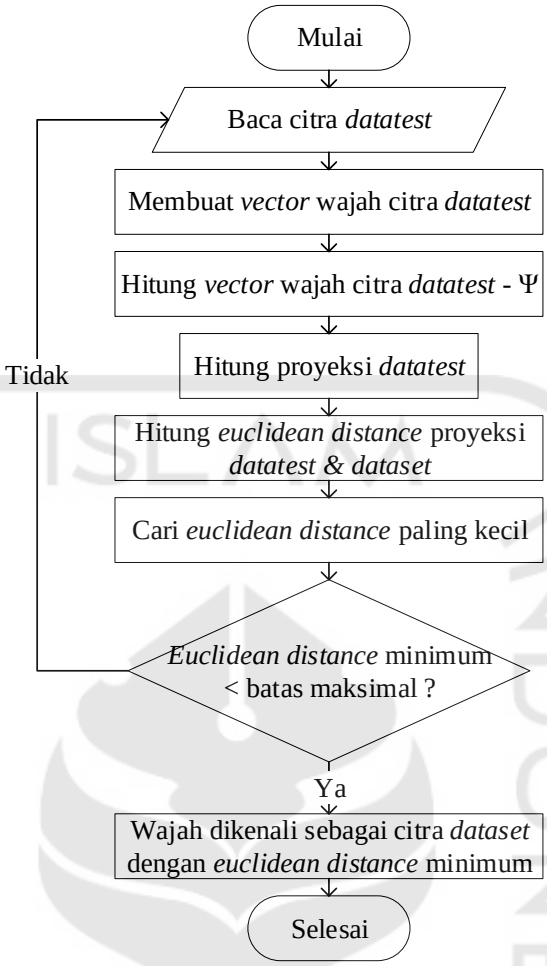
Gambar 3.16 *IMAQ Exstract*

Gambar 3.17 *IMAQ Resample*

Gambar 3.18 Hasil *Face Detection*

3.7 Face Recognition

Face recognition adalah proses mengidentifikasi wajah sehingga dapat ditentukan apakah wajah dikenali atau tidak. Prinsip dasar dari *face recognition* adalah menghitung jarak *euclidean* antara proyeksi citra *datatest* dengan proyeksi citra *dataset*. *Flowchart* proses *face recognition* ditunjukan Gambar 3.19 berikut.



Gambar 3.19 *Flowchart face recognition*

Baca citra *datatest* adalah proses mengambil data keluaran dari blok *IMAQ Resample VI* seperti pada Gambar 3.17. Citra yang ditunjukan oleh Gambar 3.17 adalah contoh citra yang dijadikan citra *datatest*. Proses membuat *vector* wajah adalah proses mengubah citra seperti pada Gambar 3.17 menjadi *vector* berukuran 65536 x 1 jika resolusi citra *datatest* adalah 256 x 256 piksel. Proses selanjutnya adalah mencari selisih antara *vector* citra *datatest* dengan *average face vector* (Ψ) yang diperoleh sesuai Persamaan (2.1). Proses meghitung proyeksi *datatest* menggunakan Persamaan (2.7). Kemudian proses yang terakhir adalah menghitung *euclidean distance* antara nilai proyeksi *datatest* dengan proyeksi *dataset* menggunakan Persamaan (2.12). Hasil akhir dari perhitungan tersebut adalah 36 nilai *euclidean distance*. Kemudian dari 36 nilai tersebut dicari nilai yang paling kecil.

Nilai batas maksimal dari hasil perhitungan jarak *euclidean* citra *datatest* dengan citra *dataset* diperlukan karena tidak setiap citra *datatest* terdapat pada *dataset.* Tabel berikut merupakan nilai batas maksimal dari masing – masing citra *dataset.*

Tabel 3.1 Data batas jarak *euclidean* maksimal

Objek	Jarak *Euclidean* Maksimal(10^{16})
Objek 1	1,1267
Objek 2	1,0611
Objek 3	1,3107
Objek 4	1,2452
Objek 5	1,0521
Objek 6	1,1444

Metode yang digunakan untuk mencari nilai batas maksimal adalah dengan menentukan nilai paling kecil perhitungan dari jarak *euclidean* citra *datatest* dengan citra *dataset* yang bersesuaian. Percobaan dilakukan dengan menguji setiap citra *datatest* yang terdapat pada citra *dataset*. Percobaan dilakukan dengan kondisi citra *datatest* yang bervariasi dari segi jarak objek (citra *datatest*) dengan kamera.