BAB III
METODOLOGI PENELITIAN

Bab ini akan membahas waktu dan tempat penelitian, serta tentang implementasi Teknik deteksi tepi yang digunakan pada pengolahan citra digital untuk penaksiran bobot sapi. Adapun juga akan menjelaskan tahap perancangan sistem yang akan dilakukan oleh penulis.

A. Waktu dan Tempat Penelitian

Waktu dan tempat penelitian merupakan serangkaian gambaran umum yang menjelaskan lokasi serta waktu dalam mengumpulkan data dalam sebuah penelitian atau riset.

1. Waktu Penelitian

Penelitian dilaksanakan pada semester ganjil dan genap tahun akademik 2022/2023, tabel waktu penelitian sebagai berikut :

Tabel 3.1 waktu penelitian

No	Kegiatan	Tahun 2022-2023 Bulan ke-					
		1	2	3	4	5	6
1	Tahap Persiapan						
	a. Studi literatur	■					
	b. Rumusan Masalah	■					
	c. Penetapan Metode	■					
	d. Seminar Proposal		■				
2	Tahap penelitian						
	a. Pengambilan Data			■	■		
	b. Analisis Data			■	■	■	
	c. Penulisan Skripsi				■	■	
3	Tahap Akhir						
	a. Sidang Skripsi						■

2. Tempat Penelitian

Dalam penelitian ini, penulis memfokuskan studi pada peternakan Queen Farm, sebuah peternakan yang terletak di Jl. Sawo, Rt 1 / Rw 4, Desa Sumingkir, Kecamatan Jeruklegi, Kabupaten Cilacap. Queen Farm didirikan pada tahun 2011 dengan mengadopsi metode peternakan fattening, dengan cara menitipkan sapi pada peternak lokal didaerah kebasen dan Jatilawang. Meskipun pada tahun pertama hanya mampu menjual dua ekor sapi, peternakan ini berhasil meningkatkan produktivitasnya secara signifikan dari tahun ke tahun, dan saat ini menjadi salah satu pilihan utama masyarakat ketika menjelang hari raya Idul Adha.

Pada awalnya, Queen Farm memiliki sebuah kandang dengan kapasitas untuk menampung 20 ekor sapi di daerah kebasen. Namun, peternakan menghadapi kendala karena jaraknya yang jauh dari para pembeli. Oleh karena itu, pada tahun 2020, peternakan Queen Farm mengambil keputusan strategis dengan membangun sebuah kandang baru di daerah Jeruklegi yang mampu menampung hingga 50 ekor sapi. Nama "Queen Farm" secara resmi diadopsi pada tahun 2017 dan peternakan ini dimiliki oleh bapak Alhadiruna. Saat ini, Queen Farm telah menjadi pengusaha yang sukses dan mampu mempekerjakan tiga orang karyawan yang berdedikasi.

Dengan semakin berkembangnya usahanya, Queen Farm berhasil memperluas jangkauan penjualannya hingga meliputi wilayah Cilacap dan sekitarnya, bahkan hingga daerah Magelang. Merespon permintaan yang semakin tinggi, peternakan ini berencana untuk memperluas kapasitas kandang baru pada tahun 2023 dengan menambah fasilitas yang mampu menampung hingga 24 ekor sapi. Selanjutnya, Queen Farm memiliki cita-cita untuk mengembangkan kandang briding guna mempermudah proses perolehan bakalan sapi dan mengurangi biaya operasional secara signifikan. Dalam usahanya, peternakan ini mampu menambah berat sapi rata-rata sebesar 6 hingga 8 ons setiap harinya halini diketahui dengan cara penimbangan rutin setiap bulan.

B. Kebutuhan Hardware dan Software

Analisis kebutuhan hardware merupakan analisis kebutuhan akan perangkat keras komputer yang diperlukan selama mengembangkan sistem, sedangkan analisis kebutuhan software untuk mendukung pengoprasian dan pembuatan sistem. Kebutuhan-kebutuhan akan hardware dan software dijelaskan sebagai berikut :

1. Kebutuhan perangkat keras

Tabel 3.2 kebutuhan perangkat keras

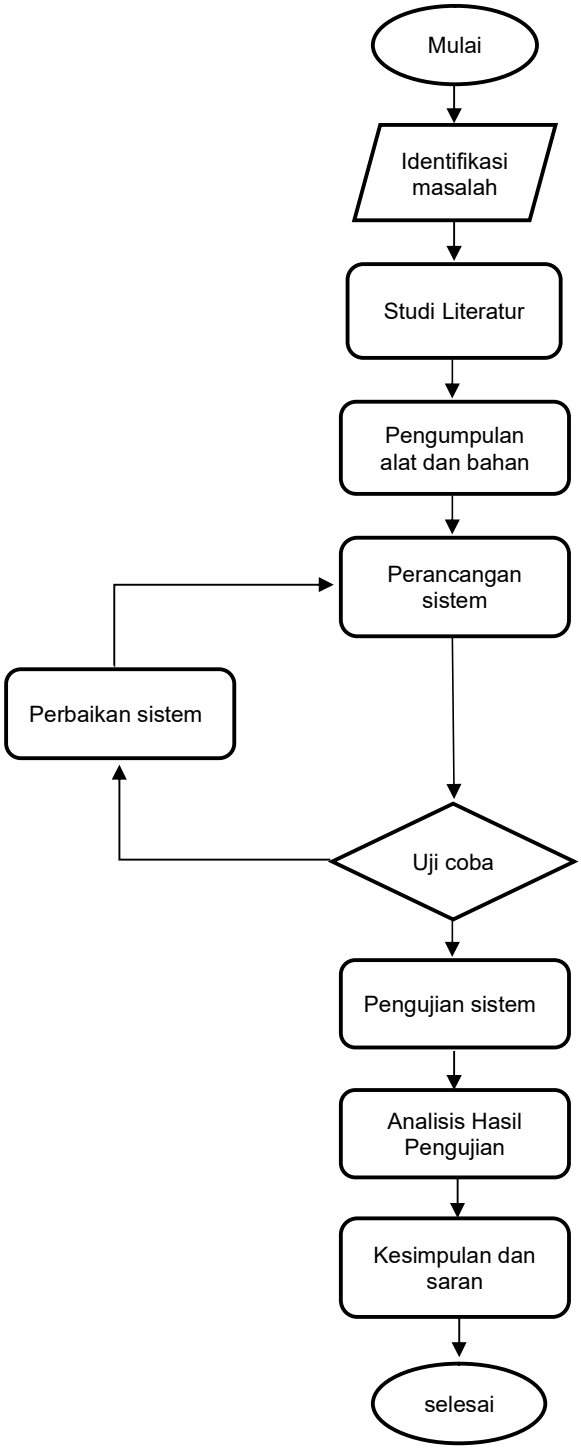
No.	Kebutuhan Perangkat Keras	Keterangan
1.	Laptop	Digunakan untuk mencari referensi, membuat diagram, membuat program aplikasi.
2.	Smart Phone	Digunakan untuk mengambil gambar atau citra sapi

2. Kebutuhan Perangkat Lunak

Tabel 3. 3 Kebutuhan perangkat lunak

No.	Kebutuhan Perangkat Lunak	Keterangan
1.	*Windows 10 32-bit*	*Operating System* pada laptop digunakan untuk pengembangan aplikasi
2.	Visual Studio Code	Digunakan untuk membuat program aplikasi pengolahan citra

C. Alur Penelitian



Gambar 3.1 Alur penelitian

Dalam melakukan penelitian pembuatan sistem ini dilakukan perancangan setelah mengetahui latar belakang dari sistem yang dibuat. Setelah menentukan alat dan bahan apa saja yang dibutuhkan untuk membuat merancang sistem, Dalam menyusun tugas akhir ini, terdapat beberapa tahapan yang dilakukan, adapun tahapan dalam metodologi penelitian ini adalah :

1. Pada tahap awal menentukan batasan masalah dan rumusan masalah. Penulis menentukan inti dari permasalahan yang ada, melakukan analisa tentang apa-apa yang perlu diteliti dan yang tidak perlu diteliti, melakukan pembatasan dari masalah yang akan dibahas sehingga tujuan penelitian menjadi terarah.
2. Studi Literatur pada tahap pengumpulan informasi, disini penulis mengumpulkan data dan referensi yang didapat dari buku-buku, dan juga jurnal dari internet mengenai hal-hal yang berkaitan dengan pengolahan citra digital, deteksi tepi dan rumus penaksiran bobot sapi.
3. Analisa kebutuhan sistem yaitu dengan menentukan alat-alat dan bahan yang dibutuhkan untuk perancangan sistem.
4. Perancangan sistem pada tahap ini penulis melalukan pengkodean di laptop untuk menjalankan rangkaian program yang telah dirancang.
5. Uji coba sistem disini penulis memeriksa dan mengevaluasi sistem yang telah dibuta untuk memastikan bahwa sistem berfungsi dengan baik dan sesuai dengan tujuan yang telah ditentukan.
6. Perbaiakan sistem adalah proses memberbaiki masalah atau kegagalan dalam sistem yang telah ditemukan oleh penulis pada saat uji coba sistem.
7. Pengujian sistem disini penulis melakukan pengujian dari sistem yang telah dibuat pada tahap sebelumnya ke dalam studi kasus yang diambil.
8. Analisis hasil pengujian pada tahap ini penulis menganalisa dan menjabarkan hasil dari proses pengujian sistem yang telah dilakukan dan melakukan uji coba langsung pada ternak sapi.
9. Pada tahap terakhir penulis membuat kesimpulan tentang kelebihan dan kekurangan dari penelitian ini dan juga saran agar kedepannya lebih maksimal

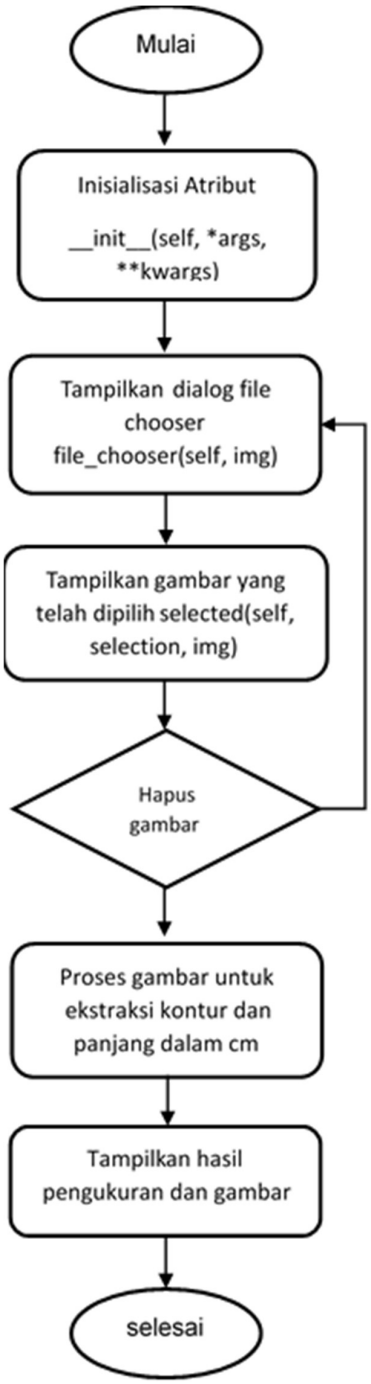
dalam melakukan penelitian sehingga meminimalisir kesalahan ataupun kegagalan yang akan terjadi.

D. Perancangan Sistem

Setelah menentukan alat dan bahan yang dibutuhkan pada penelitian ini, maka langkah selanjutnya adalah perancangan sistem keseluruhan, untuk perancangan sistem dalam penelitian ini penulis membagi menjadi tiga perancangan yaitu perancanagn sistem pengukuran panjang sapi, perancangan pengukuran lingkar dada sapi dan yang terakhir perancangan penaksiran bobot sapi dengan rumus winter. Pada tahap perancangan ini akan mengimplementasikan konsep dan dasar teori yang dibahas sebelumnya. Supaya tujuan dari perencanaan bisa tercapai dengan baik.

1. Perancangan pengukuran panjang sapi

berikut ini adalah gambar dan penjelasan flowchat proses pengolahan citra digital pengukuran panjang sapi.



Gambar 3. 2 Flowchat pengukuran panjang sapi

Flowchart diatas menunjukkan alur dari sebuah program yang digunakan untuk mengukur panjang sapi berdasarkan gambar yang diinputkan oleh pengguna. Program ini dibangun dengan menggunakan bahasa pemrograman

Python, framework Kivy dan library open cv untuk antarmuka pengguna.

Pertama-tama, saat program dijalankan, objek dari class Panjangsapi akan dibuat dan inisialisasi akan dilakukan menggunakan method constructor (__init__). Method ini akan melakukan beberapa inisialisasi atribut pada objek dan juga memanggil constructor dari superclass MDScreen.

```
class Panjangsapi(MDScreen):
    img1 = ObjectProperty(None)
    img2 = ObjectProperty(None)
    length_cm = NumericProperty(0)

     def __init__(self, *args, **kwargs):
            Builder.load_file("kv/panjangsapi.kv")
            super().__init__(*args, **kwargs)
            self.img1 = self.ids.img1
            self.img2 = self.ids.img2
            self.length_label = self.ids.length_label
```

Gambar 3.3 Class panjang sapi

Dalam kode program di atas, terdapat definisi kelas Panjangsapi yang merupakan turunan dari kelas MDScreen dari library kivymd.uix.screen. Selain itu, terdapat tiga properti yang didefinisikan dalam kelas Panjangsapi. img1: Properti ini merupakan objek dari kelas ObjectProperty yang digunakan untuk mereferensikan objek gambar pertama. img2: Properti ini juga merupakan objek dari kelas ObjectProperty yang digunakan untuk mereferensikan objek gambar kedua. length_cm: Properti ini merupakan objek dari kelas NumericProperty yang digunakan untuk merepresentasikan panjang dalam satuan sentimeter (cm). Properti ini awalnya diinisialisasi dengan nilai 0.

Properti *img1* dan *img2* digunakan untuk menghubungkan objek gambar dalam antarmuka pengguna dengan logika aplikasi. Dalam kasus ini , kami memiliki gambar-gambar sapi yang ditampilkan di antarmuka pengguna, dan kami dapat menggunakan properti ini untuk mengakses atau memanipulasi gambar-gambar sapi tersebut dalam kode Python. Properti *length_cm* digunakan untuk menyimpan panjang dalam satuan sentimeter (cm). Properti ini dapat

diperbarui dan digunakan dalam logika aplikasi, misalnya untuk menghitung atau memproses panjang sapi berdasarkan gambar yang dipilih atau data lainnya.

Metode *__init__* dalam kelas Panjangsapi memiliki beberapa baris kode yang menjalankan fungsi-fungsi yaitu. *Builder.load_file("kv/panjangsapi.kv"):* Memuat file KV yang berisi deklarasi tata letak dan properti antarmuka pengguna. *super().__init__(*args, **kwargs):* Memanggil konstruktor dari kelas induk *(MDScreen)* untuk melakukan inisialisasi yang diperlukan oleh kelas tersebut. *self.img1 = self.ids.img1:* Menginisialisasi properti *img1* dengan objek yang direferensikan oleh *ids.img1* dalam file KV. *self.img2 = self.ids.img2:* Menginisialisasi properti img2 dengan objek yang direferensikan oleh *ids.img2* dalam file KV. *self.length_label = self.ids.length_label:* Menginisialisasi properti *length_label* dengan objek yang direferensikan oleh *ids.length_label* dalam file KV. Dengan demikian, konstruktor ini memuat file KV, melakukan inisialisasi kelas induk, dan menghubungkan objek-objek antarmuka pengguna dengan properti-properti yang sesuai dalam kelas Panjangsapi.

```
def file_chooser(self, img):
    filechooser.open_file(on_selection=lambda x: self.selected(x,img))
```

Gambar 3.4 File chooser

Fungsi *file_chooser* digunakan untuk membuka pemilih berkas *(file chooser)* sistem operasi yang memungkinkan pengguna memilih berkas dari sistem file. Parameter *img* digunakan untuk menentukan objek gambar yang akan diperbarui dengan berkas yang dipilih oleh pengguna. Di dalam fungsi ini, *filechooser.open_file* dipanggil dengan menggunakan argumen *on_selection. on_selection* adalah sebuah fungsi lambda yang akan dijalankan ketika pengguna memilih berkas. Fungsi lambda ini memanggil metode *selected* dengan argumen *x* (berkas yang dipilih) dan *img* (objek gambar yang akan diperbarui).

Dengan demikian, fungsi *file_chooser* mengintegrasikan pemilih berkas sistem operasi dengan aplikasi dan memperbarui objek gambar dengan berkas yang dipilih oleh pengguna.

Pengguna dapat memilih gambar yang akan diukur panjang sapi dengan

```
def selected(self, selection, img):
        if selection:
            img.source = selection[0]
```

Gambar 3.5 Selected

menggunakan method file_chooser. Method ini akan menampilkan file chooser pada bagian input gambar dan memanggil method open_file dari objek filechooser untuk menampilkan dialog file chooser.

Fungsi *selected* digunakan untuk mengatur sumber *(source)* objek gambar *img* berdasarkan berkas yang dipilih oleh pengguna. Parameter *selection* berisi daftar berkas yang dipilih oleh pengguna. Di dalam fungsi ini, terdapat kondisi *if selection:* yang memeriksa apakah ada berkas yang dipilih oleh pengguna. Jika ada berkas yang dipilih, maka *img.source* akan diatur menjadi berkas pertama dalam daftar *selection*. Dengan demikian, fungsi *selected* mengambil berkas yang dipilih oleh pengguna dan mengaturnya sebagai sumber gambar untuk objek gambar *img*.

```
def proses_gambar(self):
    if not self.img1.source:
        return
    with Image.open(self.img1.source) as im:
        im = np.array(im)
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)
        kernel = np.ones((5,5), np.uint8)
        morph = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=3)
        contours, _ = cv2.findContours(morph, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cow_contour = max(contours, key=cv2.contourArea)
        rect = cv2.minAreaRect(cow_contour)
```

Gambar 3.6 Proses_gambar

Fungsi *proses_gambar* digunakan untuk memproses gambar yang dipilih pada objek *img1*. Fungsi ini akan melakukan beberapa operasi pemrosesan gambar menggunakan library *PIL (Pillow)* dan *cv2 (OpenCV)*. Di dalam fungsi ini terdapat beberapa langkah pemrosesan gambar, proses yang pertama dilakukan yaitu Mengecek apakah objek *img1* memiliki sumber gambar *(source)*. Jika tidak ada, maka fungsi akan mengembalikan nilai. Proses selanjutnya membuka gambar dengan menggunakan *Image.open* dari library *PIL* dan mengubahnya menjadi *array numpy* dengan *np.array(im)*. Setelah gambar dirubah menjadi *array numpy*

```
box = cv2.boxPoints(rect)
box = np.intp(box)
cv2.drawContours(im,[box],0,(0,0,255),2)
```

Gambar 3.7 Box

proses selanjutnya yaitu Mengubah gambar menjadi skala abu-abu *(grayscale)* menggunakan *cv2.cvtColor* dari library *cv2.* Selanjutnya menerapkan *thresholding* pada gambar *grayscale* dengan *cv2.threshold* untuk menghasilkan gambar *biner (binary image).* Selanjutnya menerapkan operasi *morphological opening* pada gambar biner dengan *cv2.morphologyEx* untuk menghaluskan dan memperbaiki bentuk objek. Mencari kontur pada gambar hasil operasi *morphological opening* menggunakan *cv2.findContours.* Memilih kontur dengan luas terbesar sebagai kontur sapi dengan *max(contours, key=cv2.contourArea).* Menghitung *rectangle* minimum yang melingkupi kontur sapi dengan *cv2.minAreaRect.* Langkah-langkah di atas merupakan tahap pemrosesan gambar umum untuk mendapatkan kontur dan informasi lainnya terkait objek yang terdapat dalam gambar.

Box = cv2.boxPoints(rect): Menggunakan *cv2.boxPoints* dari library *cv2* untuk menghitung koordinat titik-titik sudut dari *rectangle* minimum *(rect)* yang melingkupi kontur sapi. Hasilnya disimpan dalam variabel *box. Box = np.intp(box):* Mengubah tipe data *box* menjadi bilangan bulat *(np.intp)* untuk memastikan bahwa koordinat titik-titik sudut berupa nilai bilangan bulat.

Cv2.drawContours(im, [box], 0, (0, 0, 255), 2): Menggambar kontur berupa segi empat yang melingkupi sapi pada gambar asli *(im). cv2.drawContours* digunakan untuk menggambar kontur pada gambar dengan menggunakan koordinat titik-titik sudut *(box).* Parameter 0 menunjukkan indeks kontur yang akan digambar. *(0, 0, 255)* adalah warna garis yang digunakan (merah), dan 2 adalah ketebalan garis.

Length = max(rect[1]): Menghitung panjang dari *rectangle* minimum *(rect). rect[1]* mengakses dimensi panjang dan lebar dari *rect.* Nilai maksimum dari dimensi tersebut diambil dengan menggunakan fungsi *max(),* dan hasilnya disimpan dalam variabel *length.*

Dengan demikian, baris kode tersebut menghitung koordinat sudut-sudut dari *rectangle* minimum yang melingkupi kontur sapi, menggambar kontur berbentuk segi empat pada gambar asli, dan menandai kontur sapi dengan warna

```
scale_factor = 0.1
length_cm = length * scale_factor
self.length_cm = length_cm
self.length_label.text = "Panjang Sapi : {:.2f} Cm".format(length_cm)
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
cv2.imwrite('hasil.png', im)
self.img2.source = 'hasil.png'
```

Gambar 3.8 *scale_factor*

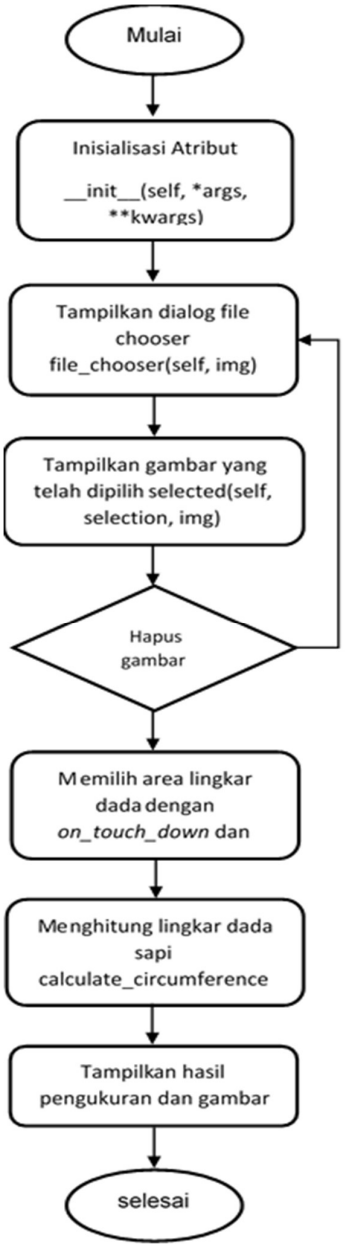
merah. Selain itu, baris kode terakhir juga menghitung panjang dari *rectangle* minimum tersebut dan menyimpannya dalam variabel *length.*

Pada program diatas *scale_factor = 0.1:* Membuat variabel *scale_factor* dengan nilai 0.1. Variabel ini digunakan untuk mengalikan panjang dari *rectangle* minimum *(length)* agar dapat dikonversi menjadi satuan centimeter. *length_cm = length * scale_factor:* Mengalikan panjang dari *rectangle* minimum *(length)* dengan *scale_factor* untuk mendapatkan panjang dalam satuan centimeter. Hasilnya disimpan dalam variabel *length_cm. self.length_cm = length_cm:* Menyimpan nilai panjang dalam satuan centimeter *(length_cm)* ke dalam atribut *length_cm* pada objek *self. self.length_label.text = "Panjang Sapi : {:.2f} Cm".format(length_cm):* Mengatur teks pada objek *length_label* dengan format *string* yang menampilkan nilai panjang sapi dalam satuan centimeter. *im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB):* Mengubah warna gambar dari *BGR (Blue-Green-Red)* menjadi *RGB (Red-Green-Blue)* menggunakan *cv2.cvtColor.* Hal ini dilakukan untuk memastikan tampilan warna yang benar pada gambar. *cv2.imwrite('hasil.png', im):* Menyimpan gambar hasil pemrosesan dengan nama file *'hasil.png'* menggunakan *cv2.imwrite.* Gambar tersebut akan disimpan dalam format PNG. *self.img2.source = 'hasil.png'*: Menetapkan sumber gambar *(source)* pada objek *img2* menjadi *'hasil.png'*. *self.img2.reload():* Memuat ulang gambar pada objek *img2* untuk menampilkan gambar hasil pemrosesan.

Dengan demikian, baris kode tersebut melakukan beberapa tindakan seperti menghitung panjang dalam satuan centimeter, menampilkan panjang pada label, mengubah warna gambar, menyimpan gambar hasil pemrosesan, dan menampilkan gambar tersebut pada objek *img2.*

2. perancangan pengukuran lingkar dada sapi

Berikut ini adalah gambar dan penjelasan flowchat proses pengolahan citra digital pengukuran lingkar dada sapi.



Gambar 3.9 *flowchart lingkar dada*

Flowchart diatas menunjukkan alur dari sebuah program yang digunakan untuk mengukur lingkar dada sapi berdasarkan gambar yang diinputkan oleh

```
class Lingkardada(MDScreen):
    img1 = ObjectProperty(None)
    points = ListProperty([])
    lines = ListProperty([])
    circumference_cm = NumericProperty(0)
```

Gambar 3.10 *class lingkar dada*

pengguna. Gambar yang dimaksud adalah gambar sapi dari tampak depan atau belakang. Berikut program klas lingkardada.

Kode program di atas mendefinisikan kelas *Lingkardada* yang merupakan turunan dari kelas *MDScreen* dalam *framework KivyMD*. Kelas ini digunakan untuk mengatur layar yang menampilkan lingkaran. Pada kelas *Lingkardada*, terdapat tiga properti yang didefinisikan:

a. *img1* adalah properti objek *(ObjectProperty)* yang digunakan untuk menunjukkan objek gambar pada tampilan antarmuka. Properti ini diinisialisasi dengan nilai *None.*

b. *points* adalah properti daftar *(ListProperty)* yang digunakan untuk menyimpan titik-titik yang membentuk lingkaran. Properti ini diinisialisasi dengan daftar kosong.

c. *lines* adalah properti daftar *(ListProperty)* yang digunakan untuk menyimpan garis-garis yang menghubungkan titik-titik pada lingkaran. Properti ini juga diinisialisasi dengan daftar kosong **[]**.

d. Selain itu, terdapat juga properti *circumference_cm* yang merupakan properti numerik *(NumericProperty)* yang digunakan untuk menyimpan nilai panjang keliling lingkaran dalam satuan sentimeter. Properti ini diinisialisasi dengan nilai *0*.

Dengan mendefinisikan properti-properti ini, kita dapat mengakses dan

mengatur nilai-nilai properti tersebut saat mengembangkan aplikasi dengan menggunakan kelas *Lingkardada.*

Pertama-tama, saat program dijalankan, objek dari class lingkar dada sapi akan dibuat dan inisialisasi akan dilakukan menggunakan method constructor (__init__). Method ini akan melakukan beberapa inisialisasi atribut pada objek dan juga memanggil constructor dari superclass MDScreen. Berikut kode program method constructor :

```
def __init__(self, *args, **kwargs):
        Builder.load_file("kv/lingkardada.kv")
        super().__init__(*args, **kwargs)
        self.img1 = self.ids.img1
```

Gambar 3.11 Metode init

Kode program di atas merupakan metode *__init__* dalam kelas *Lingkardada.* Metode ini digunakan untuk melakukan inisialisasi objek kelas. Pada metode *__init__,* terdapat beberapa langkah yang dilakukan:

a. *Builder.load_file("kv/lingkardada.kv")* digunakan untuk memuat file *KV (Kivy Language)* yang menggambarkan tampilan antarmuka aplikasi. Dalam kasus ini, file *KV* yang dimuat adalah *"lingkardada.kv"* yang berada dalam folder *"kv"*.

b. *super().__init__(*args, **kwargs)* memanggil metode *__init__* dari kelas induk *(superkelas) MDScreen*. Hal ini penting untuk melakukan inisialisasi kelas *MDScreen* secara keseluruhan.

c. *self.img1 = self.ids.img1* menginisialisasi properti *img1* dengan nilai *self.ids.img1*. *self.ids* adalah atribut yang memberikan akses ke objek dengan ID yang didefinisikan dalam file KV. Dalam hal ini, *img1* adalah ID yang diberikan pada objek gambar dalam tampilan antarmuka.

Dengan melakukan langkah-langkah di atas, kita dapat memuat tampilan antarmuka dari file KV, melakukan inisialisasi kelas *MDScreen,* dan mengakses

objek gambar dengan menggunakan properti *img1*.

Setelah itu program akan menunggu input dari pengguna. Jika pengguna memilih file gambar dengan memanggil method file_chooser(), maka akan dipanggil fungsi selected() untuk memilih file gambar yang telah dipilih oleh

```
def file_chooser(self, img):
    filechooser.open_file(on_selection=lambda x: self.selected(x,img))
```

Gambar 3.12 *file chooser*

pengguna. Apabila file telah dipilih, maka gambar akan ditampilkan pada objek img1. ***Berikut fungsi*** file_chooser()**:**

Kode program di atas merupakan metode *file_chooser* dalam kelas *Lingkardada.* Metode ini digunakan untuk membuka jendela pemilih file untuk memilih gambar.

Pada metode *file_chooser*, terdapat beberapa langkah yang dilakukan:

a. Saat metode *file_chooser* dipanggil, kita memberikan argumen *img*, yang merupakan objek gambar yang akan diatur.

b. *filechooser.open_file* digunakan untuk membuka jendela pemilih file sistem operasi. Ketika pengguna memilih file gambar, fungsi *on_selection* akan dipanggil.

c. *lambda x: self.selected(x,img)* adalah fungsi lambda yang digunakan untuk memanggil metode *selected* dengan argumen *x* (yang merupakan file yang dipilih) dan *img* (objek gambar yang akan diatur).

Dengan menggunakan *filechooser.open_file* dan fungsi lambda, kita dapat membuka jendela pemilih file, dan ketika pengguna memilih file gambar, metode *selected* akan dipanggil dengan argumen yang sesuai untuk memproses gambar yang dipilih dan mengaturnya ke objek gambar yang ditentukan. Berikut metode *selected* yang digunakan.

```
def selected(self, selection, img):
        if selection:
            img.source = selection[0]
```

Gambar 3.13 *Selected*

Kode program di atas merupakan metode *selected* dalam kelas *Lingkardada.* Metode ini digunakan untuk mengatur sumber gambar dari file yang dipilih ke objek gambar yang ditentukan.

Pada metode *selected*, terdapat beberapa langkah yang dilakukan:

a. Metode menerima dua argumen, yaitu *selection* yang berisi file yang dipilih dan *img* yang merupakan objek gambar yang akan diatur.

b. Pada baris *if selection:*, kita melakukan pengecekan apakah ada file yang dipilih. Jika *selection* tidak kosong (ada file yang dipilih), maka kode di dalam blok if akan dieksekusi.

c. Pada baris *img.source = selection[0]*, kita mengatur properti *source* dari objek gambar *img* dengan nilai *selection[0]. selection[0]* mengambil *path* atau lokasi file yang dipilih. Dengan mengatur properti *source*, objek gambar akan menampilkan gambar dari file yang dipilih.

Dengan menggunakan metode *selected*, ketika pengguna memilih file gambar melalui jendela pemilih file, metode ini akan dipanggil. File gambar yang dipilih akan diatur sebagai sumber gambar pada objek gambar yang ditentukan, sehingga gambar akan ditampilkan di antarmuka aplikasi.

Pengguna juga dapat menghapus gambar yang telah dipilih dengan memanggil fungsi clear(). Saat fungsi clear() dipanggil, program akan menghapus gambar pada objek img1, menghapus semua titik yang telah ditandai, dan mengatur nilai circumference_cm menjadi 0. Kemudian, program juga akan menghapus semua

```
def clear(self):
        self.img1.source = ''
        self.points = []
        self.circumference_cm = 0
        self.ids.circumference_label.text = "Lingkar dada: 0.0 Cm"
        for line in self.lines:
            self.img1.canvas.remove(line)
        self.lines = []
```

Gambar 3.14 Clear

garis pada canvas dengan menggunakan method canvas.remove().

Kode program di atas merupakan metode *clear* dalam kelas *Lingkardada*. Metode ini digunakan untuk menghapus semua data dan *meresе*t tampilan pada aplikasi.

Pada metode *clear*, terdapat beberapa langkah yang dilakukan yaitu:

a. Pada baris *self.img1.source = ''*, kita mengatur properti *source* dari objek gambar *img1* dengan nilai string kosong. Hal ini menghapus gambar yang ditampilkan pada objek gambar.

b. Pada baris *self.points = []*, kita mengatur properti *points* dengan daftar kosong *[]*. Dengan melakukan ini, semua titik yang sebelumnya terkumpul akan dihapus.

c. Pada baris *self.circumference_cm = 0*, kita mengatur properti *circumference_cm* dengan nilai *0*. Ini mengatur panjang keliling lingkaran menjadi 0.

d. Pada baris *self.ids.circumference_label.text = "Lingkar dada: 0.0 Cm"*, kita mengatur teks pada label yang menampilkan panjang keliling lingkaran *menjadi "Lingkar dada: 0.0 Cm"* untuk menunjukkan bahwa panjang keliling lingkaran saat ini adalah 0.

e. Pada *loop for line in self.lines,* kita mengakses setiap objek garis dalam properti *lines*.

f. Pada baris *self.img1.canvas.remove(line)*, kita menghapus setiap garis dari objek gambar *img1* dengan menggunakan metode *canvas.remove().* Ini menghapus tampilan garis dari objek gambar.

g. Pada baris *self.lines = []*, kita mengatur properti *lines* dengan daftar kosong *[]*. Ini menghapus semua garis yang sebelumnya terkumpul.

```
def on_touch_down(self, touch):
        if self.img1.collide_point(*touch.pos):
            self.points.append(touch.pos)
            with self.img1.canvas:
                Color(1, 0, 0)
                line = Line(points=(touch.pos[0]-5, touch.pos[1]-5), width=3)
                self.lines.append(line)
        return super().on_touch_down(touch)
```

Gambar 3. 15 *on touch down*

Dengan melakukan langkah-langkah di atas, metode *clear* menghapus gambar, titik-titik, garis-garis, dan mereset nilai-nilai properti yang terkait, sehingga tampilan aplikasi kembali ke kondisi awal.

Kode program di atas merupakan metode *on_touch_down* dalam kelas *Lingkardada*. Metode ini digunakan untuk menangani *event* ketika pengguna melakukan tekan pada layar *(touch down event).*

Pada metode *on_touch_down*, terdapat beberapa langkah yang dilakukan:

a. Pada baris *if self.img1.collide_point(*touch.pos):*, kita melakukan pengecekan apakah posisi tekan pengguna berada di dalam objek gambar *img1*. Jika posisi tekan berada di dalam objek gambar, maka kode di dalam blok *if* akan dieksekusi.

b. Pada baris *self.points.append(touch.pos),* kita menambahkan posisi tekan pengguna ke dalam properti *points*. Dengan melakukan ini, kita mengumpulkan titik-titik yang membentuk lingkaran berdasarkan posisi tekan pengguna.

c. Pada blok *with self.img1.canvas:*, kita mengatur konteks penggambaran pada objek gambar *img1*. Semua perintah penggambaran yang diberikan setelah ini akan diterapkan pada objek gambar.

d. Pada baris *Color(1, 0, 0)*, kita mengatur warna yang akan digunakan dalam penggambaran. Di sini, warna yang digunakan adalah merah dengan nilai RGB (1, 0, 0).

e. Pada baris *line = Line(points=(touch.pos[0]-5, touch.pos[1]-5), width=3),* kita membuat objek garis *(Line)* dengan posisi awal yang berdasarkan posisi tekan pengguna *(touch.pos)* dikurangi dengan 5, serta lebar garis sebesar 3.

f. Pada baris *self.lines.append(line),* kita menambahkan objek garis yang baru saja dibuat ke dalam properti *lines*. Dengan melakukan ini, kita mengumpulkan garis-garis yang menghubungkan titik-titik pada lingkaran.

g. Pada baris *return super().on_touch_down(touch)*, kita memanggil metode *on_touch_down* dari kelas induk (*superkelas*) untuk menangani *event touch down* secara umum.

Dengan menggunakan metode *on_touch_down*, ketika pengguna melakukan tekan pada layar di dalam objek gambar *img1*, posisi tekan akan diambil, titik akan ditambahkan ke properti *points,* dan garis akan digambar dan ditambahkan ke properti *lines.*

```
def on_touch_up(self, touch):
        if self.img1.collide_point(*touch.pos):
            self.points.append(touch.pos)
            with self.img1.canvas:
                Color(1, 0, 0)
                line = Line(points=self.points, width=3)
                self.lines.append(line)
        return super().on_touch_up(touch)
```

Gambar 3.16 *On touch up*

Kode program di atas merupakan metode *on_touch_up* dalam kelas *Lingkardada*. Metode ini digunakan untuk menangani event ketika pengguna melepaskan tekanan dari layar *(touch up event).*

Pada metode *on_touch_up*, terdapat beberapa langkah yang dilakukan:

a. Pada baris *if self.img1.collide_point(*touch.pos):*, kita melakukan pengecekan apakah posisi lepas tekan pengguna berada di dalam objek gambar *img1*. Jika posisi lepas tekan berada di dalam objek gambar, maka kode di dalam *blok if* akan dieksekusi.

b. Pada baris *self.points.append(touch.pos),* kita menambahkan posisi lepas tekan pengguna ke dalam properti *points.* Dengan melakukan ini, kita memperbarui titik terakhir pada lingkaran.

c. Pada blok *with self.img1.canvas:,* kita mengatur konteks penggambaran pada objek gambar *img1*. Semua perintah penggambaran yang diberikan setelah ini akan diterapkan pada objek gambar.

d. Pada baris *Color(1, 0, 0)*, kita mengatur warna yang akan digunakan dalam penggambaran. Di sini, warna yang digunakan adalah merah dengan nilai *RGB (1, 0, 0).*

e. Pada baris *line = Line(points=self.points, width=3),* kita membuat objek garis *(Line)* dengan titik-titik yang membentuk lingkaran, yang terdapat dalam properti *points*. Lebar garis ditetapkan sebagai 3.

f. Pada baris *self.lines.append(line)*, kita menambahkan objek garis yang baru saja dibuat ke dalam properti *lines.* Dengan melakukan ini, kita mengumpulkan garis-garis yang menghubungkan titik-titik pada lingkaran.

g. Pada baris *return super().on_touch_up(touch),* kita memanggil metode *on_touch_up* dari kelas induk *(superkelas)* untuk menangani *event touch up* secara umum.

Dengan menggunakan metode *on_touch_up*, ketika pengguna melepaskan tekanan dari layar di dalam objek gambar *img1,* posisi lepas tekan akan diambil, titik akan ditambahkan ke properti *points,* dan garis akan digambar berdasarkan semua titik yang terkumpul sejauh ini.

```
def calculate_circumference(self):
        if len(self.points) > 2:
            circumference = 0
            for i in range(len(self.points)-1):
                x1, y1 = self.points[i]
                x2, y2 = self.points[i+1]
                diameter = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
                circumference += diameter
```

Gambar 3.17 calculate circumference

Kode program di atas merupakan metode *calculate_circumference* dalam *kelas Lingkardada*. Metode ini digunakan untuk menghitung keliling lingkaran berdasarkan titik-titik yang telah dikumpulkan. Pada metode *calculate_circumference*, terdapat beberapa langkah yang dilakukan:

a. Pada baris *if len(self.points) > 2:*, kita melakukan pengecekan apakah terdapat cukup banyak titik untuk membentuk lingkaran. Kondisi ini memastikan bahwa terdapat minimal 3 titik yang telah dikumpulkan sebelumnya untuk menghitung keliling lingkaran.

b. Pada baris *circumference = 0*, kita menginisialisasi variabel *circumference* dengan nilai awal 0. Variabel ini akan digunakan untuk mengakumulasi nilai diameter dan menghitung keliling.

c. Pada *loop for i in range(len(self.points)-1):*, kita melakukan iterasi sebanyak jumlah titik dikurangi 1. Hal ini dilakukan karena setiap titik akan membentuk diameter dengan titik berikutnya.

d. Pada baris *x1, y1 = self.points[i] dan x2, y2 = self.points[i+1],* kita mendapatkan koordinat x dan y dari titik saat ini dan titik berikutnya yang membentuk diameter.

e. Pada baris *diameter = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5*, kita menghitung panjang diameter berdasarkan rumus jarak antara dua titik di bidang kartesian. Rumus ini menggunakan *teorema Pythagoras*.

f. Pada baris *circumference* += *diameter*, kita menambahkan nilai diameter ke variabel *circumference.* Dengan melakukan ini, kita mengakumulasi nilai diameter dari setiap segmen lingkaran.

Setelah melakukan iterasi untuk semua segmen lingkaran, nilai keliling lingkaran akan terkumpul dalam variabel *circumference*. Namun, tidak ada langkah lanjutan dalam kode untuk memperbarui properti *circumference_cm* atau menampilkan nilai keliling lingkaran ke antarmuka pengguna.

```
x1, y1 = self.points[-1]
x2, y2 = self.points[0]
diameter = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
circumference += diameter
```

Gambar 3.18 *self point*

Kode program di atas merupakan bagian dari metode *calculate_circumference* dalam kelas *Lingkardada*. Kode ini digunakan untuk menghitung diameter dan menambahkannya ke total keliling lingkaran. Pada bagian ini, kita mengambil koordinat x dan y dari titik terakhir *(self.points[-1])* dan titik pertama *(self.points[0])*. Dengan demikian, kita membentuk diameter dengan menghubungkan titik terakhir dan titik pertama pada lingkaran. Kemudian, menggunakan rumus jarak antara dua titik di bidang kartesian, yaitu *((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5*, kita menghitung panjang diameter tersebut. Selanjutnya, nilai diameter diakumulasi ke dalam variabel *circumference* dengan menggunakan operator penjumlahan (+=). Dengan melakukan ini, kita menambahkan panjang diameter tersebut ke total keliling lingkaran yang telah dihitung sebelumnya.

Dengan bagian kode ini, kita memastikan bahwa segmen terakhir pada lingkaran, yang menghubungkan titik terakhir dan titik pertama, juga dihitung dalam perhitungan keliling lingkaran

Kode program di atas juga merupakan bagian dari metode *calculate_circumference* dalam kelas *Lingkardada*. Kode ini digunakan untuk mengkonversi nilai keliling lingkaran dari satuan piksel ke sentimeter dan memperbarui properti *circumference_cm* serta menampilkan nilai keliling

lingkaran di antarmuka pengguna.

```
cm_per_pixel = 0.45
        self.circumference_cm = circumference * cm_per_pixel
        self.ids.circumference_label.text = f"Lingkar dada: {self.circumference_cm
:.2f} Cm"
```

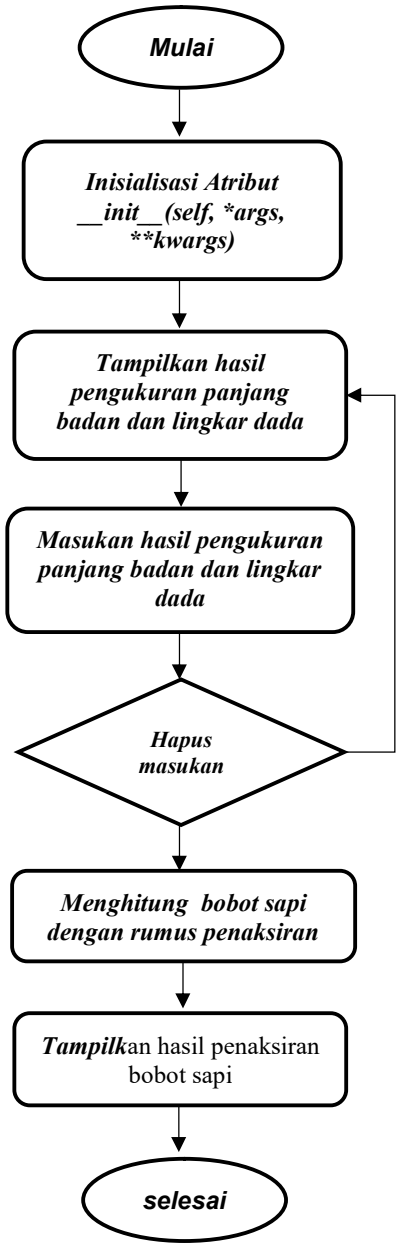
Gambar 3.19 *cm per pixel*

Baris *cm_per_pixel = 0.45*, kita mendefinisikan faktor konversi antara piksel dan sentimeter. Nilai ini merepresentasikan berapa sentimeter setara dengan satu piksel pada gambar. Selanjutnya, pada baris *self.circumference_cm = circumference * cm_per_pixel*, kita mengalikan nilai keliling lingkaran dengan faktor konversi tersebut. Dengan melakukan ini, kita mendapatkan nilai keliling lingkaran dalam satuan sentimeter.

Baris *self.ids.circumference_label.text = f"Lingkar dada: {self.circumference_cm:.2f} Cm",* kita memperbarui teks yang ditampilkan pada label dengan menggunakan f-string. Label tersebut akan menampilkan nilai keliling lingkaran yang telah dikonversi ke sentimeter dengan dua angka desimal.
Dengan kode ini, kita mengubah skala dari nilai keliling lingkaran yang semula dalam satuan piksel menjadi satuan sentimeter, sehingga dapat memberikan informasi yang lebih relevan dan lebih mudah dipahami oleh pengguna.

3. Perancangan sisitem penaksiran bobot sapi

berikut ini adalah gambar dan penjelasan flowchat proses penaksiran bobot sapi yang akan dilakukan.



Gambar 3.20 Flowchat pengolahan citra

Flowchart diatas menunjukkan alur dari sebuah program yang digunakan untuk penaksiran bobot sapi berdasarkan hasil dari proses perhitungan panjang badan sapi dan perhitungan lingkar dada sapi.

```
class Rumus(MDScreen):
    lingkar_dada = NumericProperty()
    panjang_badan = NumericProperty()
```

Gambar 3.21 *class rumus*

Kode di atas mendefinisikan kelas *Rumus* yang merupakan turunan dari kelas *MDScreen* dari modul *kivymd.uix.screen*. Kelas ini digunakan untuk membuat layar dalam aplikasi dengan nama "Rumus".

Kelas *Rumus* memiliki dua properti, yaitu *lingkar_dada* dan *panjang_badan*, yang keduanya merupakan *NumericProperty* dari modul *kivy.properties*. Properti-properti ini digunakan untuk menyimpan nilai numerik yang terkait dengan perhitungan rumus.

Properti *lingkar_dada* digunakan untuk menyimpan lingkar dada, sementara properti *panjang_badan* digunakan untuk menyimpan panjang badan. Kedua properti ini dideklarasikan sebagai *NumericProperty*, yang berarti mereka hanya dapat menyimpan nilai numerik. Deklarasi properti ini memungkinkan penggunaan *binding* atau *referencing* dengan properti ini dalam tampilan antarmuka pengguna *(UI)* atau dalam logika aplikasi.

```
def __init__(self, **kwargs):
        Builder.load_file("kv/rumus.kv")
        super().__init__(**kwargs)
```

Gambar 3.22 *metode init*

Metode `__init__` di atas adalah metode *konstruktor* yang digunakan untuk menginisialisasi objek dari kelas *`Rumus`*. Berikut adalah penjelasan singkat untuk setiap baris kode:

a. *Builder.load_file("kv/rumus.kv")`:* Ini memuat file KV dengan nama *"rumus.kv"* menggunakan metode *`load_file`* dari kelas *`Builder`*. File KV

mengandung deskripsi tampilan antarmuka pengguna *(UI)* yang akan digunakan oleh kelas *`Rumus`*. Dengan memuat file KV, tampilan akan diatur sesuai dengan deskripsi yang diberikan.

b. *super().__init__(**kwargs)`*: Ini memanggil metode konstruktor dari kelas induk, yaitu *`MDScreen`*, menggunakan fungsi *`super()`*. Metode konstruktor ini menginisialisasi objek *`MDScreen`* dengan argumen yang diteruskan melalui *`kwargs` (keyword arguments).* Dengan memanggil metode *konstruktor* kelas induk, semua inisialisasi yang diperlukan oleh *`MDScreen`* akan dilakukan, seperti mengatur tata letak, menghubungkan properti, dan sebagainya.

Dalam konteks ini, metode *`__init__`* digunakan untuk menginisialisasi objek *`Rumus`* dengan memuat tampilan dari file KV dan melakukan inisialisasi kelas induk.

```
def on_pre_enter(self, *args):
        lingkardada = self.manager.get_screen("lingkardada")
        circumference_cm = lingkardada.circumference_cm
        self.ids.circumference_label.text = "Lingkar dada : {:.2f}
Cm".format(circumference_cm)
        panjangsapi = self.manager.get_screen("panjangsapi")
        length_cm = panjangsapi.length_cm
        self.ids. length_label.text = "Panjang sapi : {:.2f} Cm".format(length_cm)
```

Gambar 3.23 *on pre enter*

Metode *`on_pre_enter`* di atas adalah metode yang dipanggil sebelum masuk ke layar *`Rumus`* dalam penanganan navigasi antarmuka pengguna. Berikut adalah penjelasan singkat untuk setiap baris kode:

a. *lingkardada = self.manager.get_screen("lingkardada")`:* Ini mendapatkan objek layar dengan nama *"lingkardada"* dari manajer layar *(ScreenManager)* yang mengelola tampilan. Dalam hal ini, diasumsikan bahwa ada sebuah layar dengan nama *"lingkardada"* yang telah didefinisikan dan terhubung ke manajer layar.

b. *circumference_cm = lingkardada.circumference_cm`:* Ini mengambil nilai atribut *`circumference_cm` dari layar "lingkardada".* Atribut ini mewakili nilai lingkar dada yang telah dihitung sebelumnya.

c. *self.ids.circumference_label.text = "Lingkar dada : {:.2f} Cm".format(circumference_cm)`:* Ini mengatur teks dari objek label dengan ID *"circumference_label"* yang ada di tampilan *`Rumus`*. Label ini akan menampilkan informasi tentang lingkar dada dengan format desimal dua angka diikuti oleh satuan *"Cm".*

d. *panjangsapi = self.manager.get_screen("panjangsapi")`:* Ini mendapatkan objek layar dengan nama "*panjangsapi"* dari manajer layar.

e. *length_cm = panjangsapi.length_cm`:* Ini mengambil nilai atribut *`length_cm`* dari layar *"panjangsapi".* Atribut ini mewakili nilai panjang sapi yang telah dihitung sebelumnya.

f. *self.ids.length_label.text = "Panjang sapi : {:.2f} Cm".format(length_cm)`:* Ini mengatur teks dari objek label dengan ID *"length_label"* yang ada di tampilan *`Rumus`*. Label ini akan menampilkan informasi tentang panjang sapi dengan format desimal dua angka diikuti oleh satuan *"Cm".*

Dalam konteks ini, metode *`on_pre_enter`* digunakan untuk mendapatkan nilai lingkar dada dan panjang sapi dari layar *"lingkardada"* dan *"panjangsapi"* yang terkait, dan menampilkan nilai-nilai ini dalam tampilan *`Rumus`* sebelum layar tersebut ditampilkan kepada pengguna.

```
def hitung_bobot_sapi(self):
        bb = round(((self.lingkar_dada ** 2) + (self.panjang_badan ** 2)) /
10815.15, 2)
        self.ids.bobot_sapi.text = "Berat Badan Sapi : {}
kg".format(str(bb).replace('.', ''))
```

Gambar 3.24 hitung bobot sapi

Metode *hitung_bobot_sapi* di atas adalah metode yang digunakan untuk

menghitung berat badan sapi berdasarkan nilai lingkar dada *(lingkar_dada)* dan panjang badan *(panjang_badan).* Berikut adalah penjelasan singkat untuk setiap baris kode:

a. *bb = round(((self.lingkar_dada ** 2) + (self.panjang_badan ** 2)) / 10815.15, 2):* Ini menghitung berat badan sapi dengan menggunakan rumus yang diberikan. Nilai lingkar dada dan panjang badan dikuadratkan, kemudian dijumlahkan dan dibagi dengan *10815.15.* Hasilnya dibulatkan menjadi dua angka desimal dan disimpan dalam variabel *bb.*

b. *self.ids.bobot_sapi.text = "Berat Badan Sapi : {} kg".format(str(bb).replace('.', '')):* Ini mengatur teks dari objek label dengan ID *"bobot_sapi"* yang ada di tampilan *Rumus.* Label ini akan menampilkan informasi tentang berat badan sapi dengan format *"Berat Badan Sapi : [nilai] kg"*, di mana [nilai] adalah nilai berat badan sapi yang telah dihitung sebelumnya. Sebelum memasukkan nilai ke dalam string format, titik desimal dihapus menggunakan metode *replace('.', '')* untuk menghasilkan tampilan berat badan yang lebih sesuai.

Dalam konteks ini, metode *hitung_bobot_sapi* digunakan untuk menghitung dan menampilkan berat badan sapi berdasarkan nilai lingkar dada dan

```
def set_lingkar_dada(self, value):
        if value:
            self.lingkar_dada = float(value)
        else:
            self.lingkar_dada = 0.0
```

Gambar 3.25 *set lingkar dada*

panjang badan yang telah ditentukan dalam tampilan *Rumus.*

Metode *`set_lingkar_dada`* di atas digunakan untuk mengatur nilai atribut *`lingkar_dada`* dalam kelas *`Rumus`* berdasarkan nilai yang diberikan sebagai argumen *`value`*. Berikut adalah penjelasan singkat untuk setiap baris kode:

a. *`if value:`*: Ini adalah kondisi yang memeriksa apakah nilai *`value`* ada atau tidak kosong *(None, 0, False, atau string kosong)*. Jika nilai *`value`* ada, maka kondisi ini bernilai *True*.

b. *`self.lingkar_dada = float(value)`*: Jika nilai *`value`* ada, maka nilai *`value`* akan diubah menjadi tipe data *float* dan akan diassign ke atribut *`lingkar_dada`*. Ini memungkinkan atribut *`lingkar_dada`* untuk menyimpan nilai lingkar dada dalam bentuk desimal.

c. *`else:`*: Jika nilai *`value`* kosong atau tidak ada, maka kondisi ini akan dieksekusi.

d. *`self.lingkar_dada = 0.0`*: Dalam kondisi ini, atribut *`lingkar_dada`* akan diassign dengan nilai float 0.0. Ini berarti jika tidak ada nilai lingkar dada yang diberikan, maka nilai lingkar dada akan diatur menjadi 0.0.

Metode *`set_lingkar_dada`* ini memungkinkan pengaturan nilai lingkar dada dalam kelas *`Rumus`* dengan memperhatikan kemungkinan bahwa nilai yang diberikan

```
def set_panjang_badan(self, value):
        if value:
            self.panjang_badan = float(value)
        else:
            self.panjang_badan = 0.0
```

Gambar 3.26 *set Panjang badan*

dapat berupa nilai numerik atau dapat kosong.

Metode *`set_panjang_badan`* di atas digunakan untuk mengatur nilai atribut *`panjang_badan`* dalam kelas *`Rumus`* berdasarkan nilai yang diberikan sebagai argumen *`value`*. Berikut adalah penjelasan singkat untuk setiap baris kode:

a. *`if value:`*: Ini adalah kondisi yang memeriksa apakah nilai *`value`* ada atau tidak kosong (*None, 0, False*, atau string kosong). Jika nilai *`value`* ada, maka kondisi ini bernilai *True*.

b. *`self.panjang_badan = float(value)`*: Jika nilai *`value`* ada, maka nilai *`value`* akan diubah menjadi tipe data float dan akan diassign ke atribut *`panjang_badan`*. Ini memungkinkan atribut *`panjang_badan`* untuk menyimpan nilai panjang badan dalam bentuk desimal.

c. *`else:`*: Jika nilai *`value`* kosong atau tidak ada, maka kondisi ini akan dieksekusi.

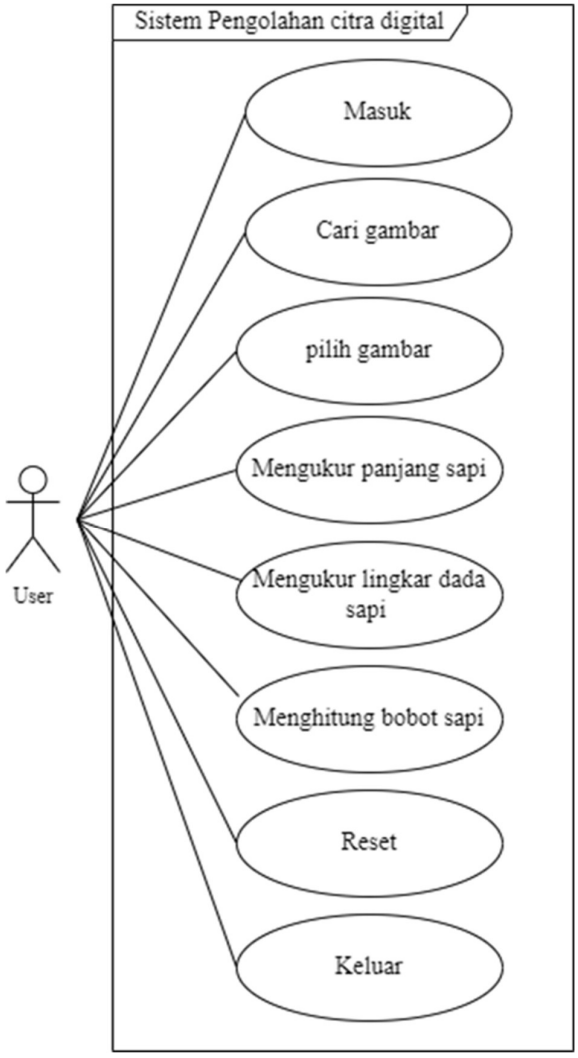
d. *`self.panjang_badan = 0.0`*: Dalam kondisi ini, atribut *`panjang_badan`* akan diassign dengan nilai float 0.0. Ini berarti jika tidak ada nilai panjang badan yang diberikan, maka nilai panjang badan akan diatur menjadi 0.0.

Metode *`set_panjang_badan`* ini memungkinkan pengaturan nilai panjang badan dalam kelas *`Rumus`* dengan memperhatikan kemungkinan bahwa nilai yang diberikan dapat berupa nilai numerik atau dapat kosong.

4. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. *Use Case Diagram* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Ade Hendini 2016).

Dalam penelitian ini penulis membuat use case diagram pada sistem pengolahan citra digital pada gambar berikut:



Gambar 3.27 *Use Case Diagram*

a. Masuk

Nama *Use Case*	: Masuk
Aktor	: *User*
Deskripsi	: User harus masuk terlebih dahulu untuk dapat mengakses aplikasi
Pre-Condition	: *User* membuka aplikasi kemudian menekan tombol masuk
Post-Condition	: Sistem akan menampilkan tombol masuk dan ketika pengguna telah menekan tobol masuk sistem akan membuka halaman selanjutnya.
Alur skenario	:

1. Pengguna membuka aplikasi penaksiran bobot
2. Aplikasi menampilkan halaman masuk
3. pengguna menekan tombol masuk
4. aplikasi membuka halaman selanjutnya.

b. Cari gambar

Nama *Use Case*	: Cari gambar
Aktor	: *User*
Deskripsi	: User memilih gambar yang akan diproses
Pre-Condition	: *User* berada dihalaman utama lalu menekan tombol pilih gambar
Post-Condition	: Sistem akan masuk kepencarian gambar
Alur skenario	:

1. Pengguna berada pada halaman hitung panjang sapi atau lingkar dada sapi
2. Pengguna menekan tombol pilih gambar.
3. Sitem membuka galeri dari perangkat pengguna.

c. pilih gambar

Nama *Use Case*	: pilih gambar
Aktor	: *User*
Deskripsi	: pengguna memilih gambar yang ingin digunakan untuk

menghitung panjang badan dan lingkar dada sapi.

Pre-Condition : *User* menekan tombol open

Post-Condition : Sistem akan menunggu pengguna memilih gambar lalu ditampilkan pada layar.

Alur skenario :

1. pengguna berada pada galeri pemilihan gambar.
2. Pengguna memilih gambar yang akan diproses untuk pengukuran panjang sapi dan lingkar dada sapi.
3. Sistem menampilkan gamabar yang telah dipilih oleh pengguna.

d. Mengukur panjang sapi

Nama *Use Case* : mengukur panjang sapi

Aktor : User

Deskripsi : mengukur panjang sapi pada gambar yang diunggah

Pre-Condition : *User* telah mengunggah gambar sapai lalu pengguna menekan tombol proses

Post-Condition : Sistem menampilkan hasil pemrosesan pada jendala hasil

Alur Skenario :

1. Pengguna membuka halaman ukur panjang sapi.
2. Aplikasi menampilkan antarmuka pengguna.
3. Pengguna memilih opsi untuk memasukkan gambar sapi.
4. Aplikasi membuka file chooser dan meminta pengguna untuk memilih gambar sapi yang ingin diukur.
5. Pengguna memilih gambar sapi yang ingin diukur dan memilih opsi untuk memproses gambar.
6. Aplikasi memproses gambar dengan menggunakan algoritma yang telah diimplementasikan.
7. Aplikasi menampilkan gambar hasil pemrosesan pada antarmuka pengguna.
8. Aplikasi menampilkan hasil pengukuran panjang sapi dalam satuan sentimeter pada antarmuka pengguna.
9. Pengguna dapat memilih untuk mengukur gambar sapi lain atau melanjutkan

proses menghitung lingkar dada sapi.

e. Mengukur lingkar dada sapi

Nama *Use Case*	: mengukur lingkar dada sapi
Aktor	: User
Deskripsi	: mengukur lingkar dada sapi pada gambar yang telah diunggah
Pre-Condition	: *User* telah menyiapkan gambar yang akan diproses pada halaman hitung lingkar dada sapi lalu pengguna memilih bagian lingkar dada yang ingin diukur.
Post-Condition	: setelah lingkar dada berhasil dihitung maka hasil pengukuran akan ditampilkan dalam satuan cm pada label yang tersedia.

Alur Skenario:

1. Pengguna membuka halaman pengukur lingkar dada.
2. Aplikasi menampilkan tampilan untuk mengukur lingkar dada.
3. Pengguna memilih opsi untuk mengambil gambar lingkar dada.
4. Aplikasi membuka jendela pemilihan file.
5. Pengguna memilih file gambar lingkar dada yang ingin diukur.
6. Aplikasi menampilkan gambar lingkar dada yang dipilih pengguna.
7. Pengguna melakukan tap pada titik awal pengukuran lingkar dada pada gambar.
8. Aplikasi menampilkan titik awal pada gambar lingkar dada.
9. Pengguna melakukan tap pada titik berikutnya pada gambar untuk mengukur lingkar dada.
10. Aplikasi menampilkan garis penghubung antara titik sebelumnya dan titik sekarang pada gambar lingkar dada.
11. Langkah 9 dan 10 diulang sampai semua titik sudut lingkar dada telah terukur.
12. Aplikasi menghitung lingkar dada berdasarkan posisi titik sudut yang telah terukur.
13. Aplikasi menampilkan hasil pengukuran lingkar dada dalam satuan cm pada

label yang tersedia.

14. Pengguna dapat mengulangi pengukuran lingkar dada atau keluar melanjutkan ke proses selanjutnya.

f. Menghitung bobot sapi

Nama use case	: menghitung bobot sapi
Aktor	: user
Deskripsi	: menghitung bobot sapi dengan panjang badan sapi dan lingkar dada sapi yang telah diproses sebelumnya.
Pre-Condition	: user memasukan hasil perhitungan panjang badan dan lingkar dada.
Post-Condition	: Aplikasi menghitung dan menampilkan hasil perhitungan bobot sapi.
Alur sekenario	:

1. pengguna berada pada halaman perhitungan bobot.
2. Aplikasi menampilkan halaman hitung bobot sapi.
3. User memasukkan lingkar dada pada kolom yang tersedia
4. Aplikasi menyimpan nilai lingkar dada
5. User memasukkan panjang badan pada kolom yang tersedia
6. Aplikasi menyimpan nilai panjang badan
7. User menekan tombol hitung bobot sapi
8. Aplikasi menghitung bobot sapi menggunakan rumus: ((lingkar_dada ** 2) + (panjang_badan ** 2)) / 10815.15
9. Aplikasi menampilkan hasil perhitungan bobot sapi ke dalam tampilan.

g. Reset

Nama Use Case	: Reset
Aktor	: User
Deskripsi	: Mereset semua gambar dan Hasil proses yang telah dilakukan sistem
Pre-Condition	: User menekan tombol reset

Post-Condition	: Sistem mereset semua gambar dan hasil proses yang telah dilakukan dan kembali ketampilan awal sebelum pilih gambar.
Alur skenario	:

1. pengguna berada di halaman hitung panjang sapi, hitung lingkar dada sapi dan hitung bobot sapi.
2. Pengguna menakantombol hapus.
3. Sistem menghapus gambar dan hasil pemrosesan gambar yang telah diupload oleh pengguna
4. Sistem menghapus label yang telah ditampilkan.

h. Keluar

Nama Use Case	: keluar
Aktor	: User
Deskripsi	: Proses ini merupakan tahap akhir yang dilakukan user atau dalam menggunakan aplikasi
Pre-Condition	: User menekan tombol keluar.
Post-Condition	: Sistem akan memutuskan koneksi dan keluar dari aplikasi.
Alur skenario	:

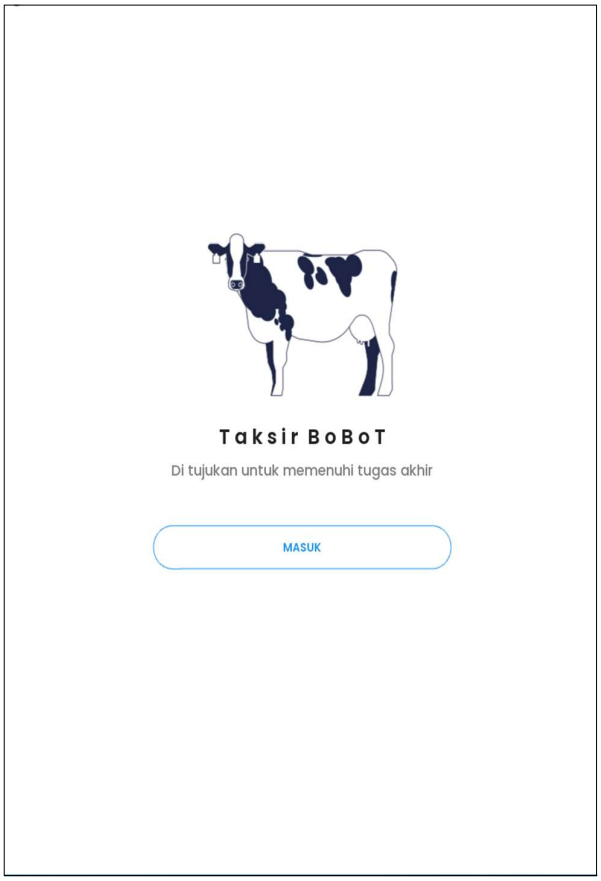
1. Pengguna berada di halaman hitung bobot sapi
2. Pengguna menekan tombol keluar.
3. Sistem berhenti dan ditutup secara otomatis

5. Prototype

Prototype sebagai pemodelan dasar gambaran dari suatu pengembangan program. Prototype digunakan sebagai contoh gambaran dari suatu rancangan aplikasi. Berikut merupakan prototype pada aplikasi yang akan dibuat pada penelitian ini :

a. *Welcome screen*

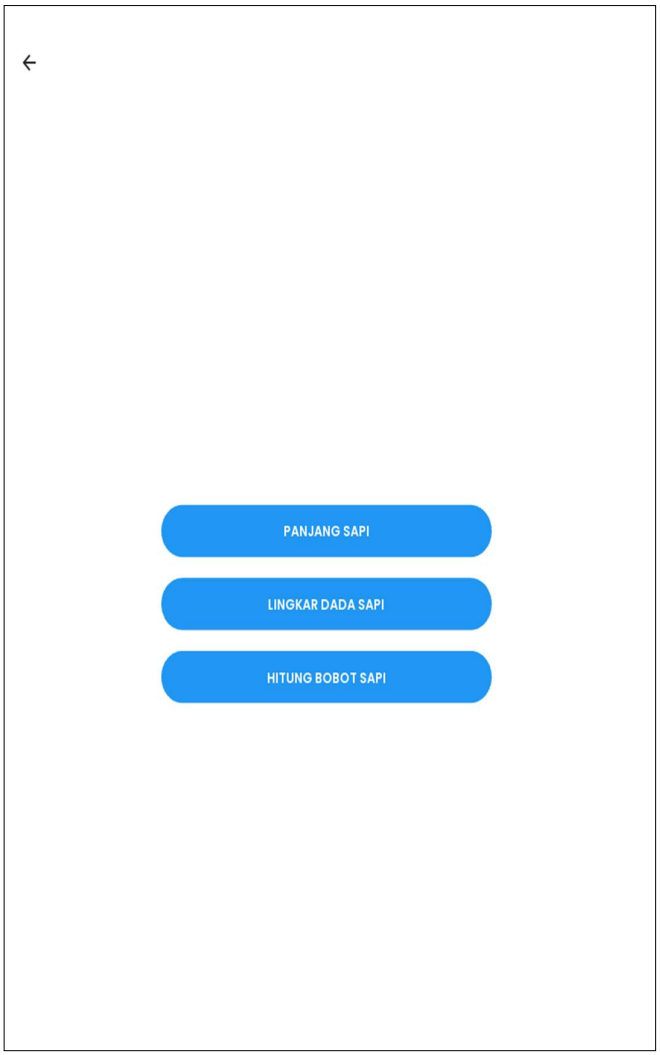
Welcome screen adalah halaman atau tampilan pertama yang muncul ketika pengguna membuka aplikasi. tujuan dari *welcome screen* adalah memberikan pengenalan dan menyambuat pengguna sebelum pengguna memasuki halaman utama aplikasi.



Gambar 3.28 *welcome screen*

b. Halaman utama

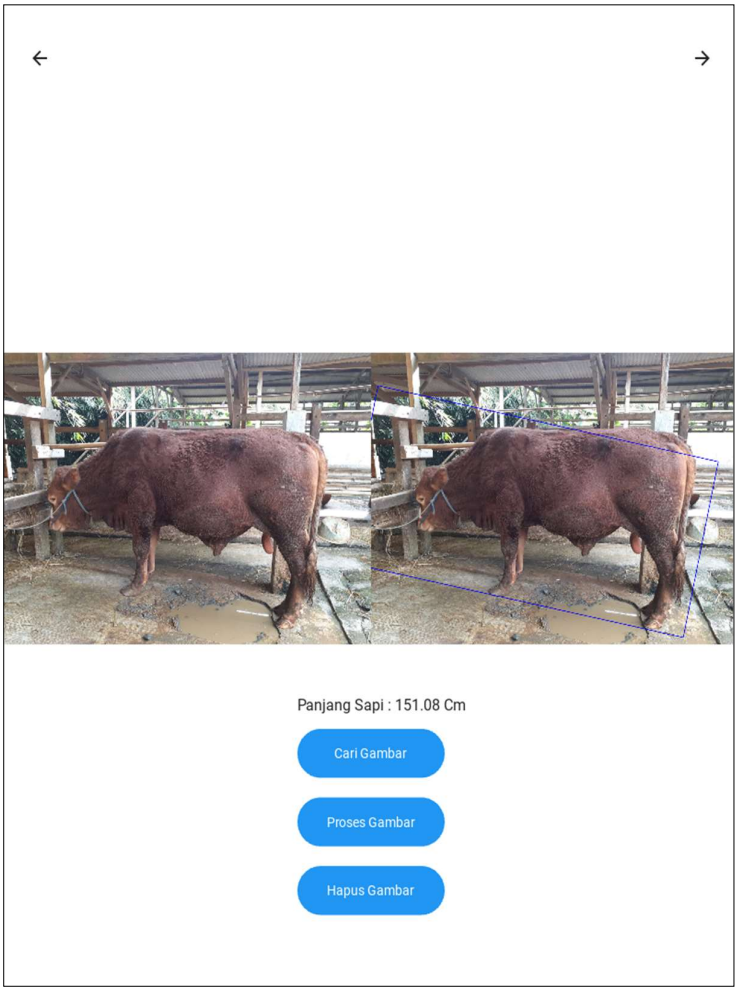
Pada halaman utama menampilkan beberapa tombol yaitu tombol untuk menghitung panjang sapi, menghitung lingkar dada sapi dan menghitung bobot sapi. Pengguna bisa menggunakan tombol tersebut untuk melanjutkan pada proses selanjutnya yaitu menghitung panjang sapi, menghitung lingkar dada sapi dan menghitung bobot sapi.



Gambar 3.29 halaman utama

c. Pengukuran panjang sapi.

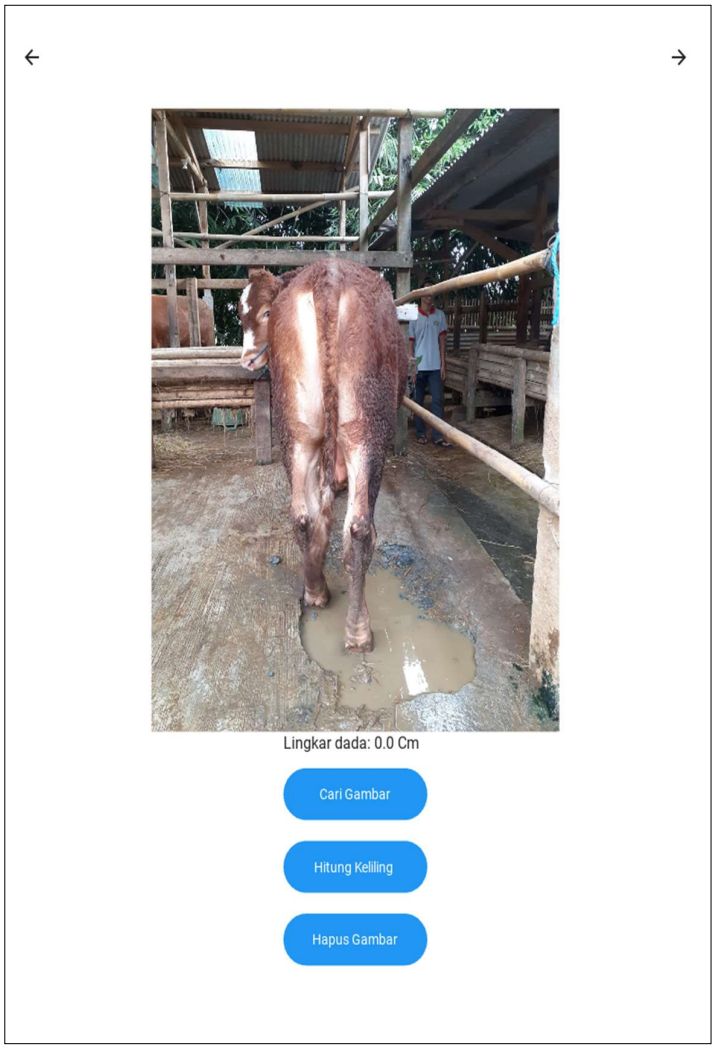
Pada halaman menghitung panjang badan sapi pengguna bisa memilih gambar sapi yang akan diperoses dengan menekan tombol cari gambar, setelah itu sistem akan menampilkan gambar yang telah dipilih pada layar. Pengguna mengukur panjang badan sapi dengan menekan tombol proses gambar dan sistem akan menampilkan gambar hasil pemrosesan gambar pada layar, sistem juga menampilkan hasil pengukuran panjang sapi dengan satuan Cm pada layar.



Gambar 3.30 panjang sapi

d. Halaman pengukuran lingkar dada sapi

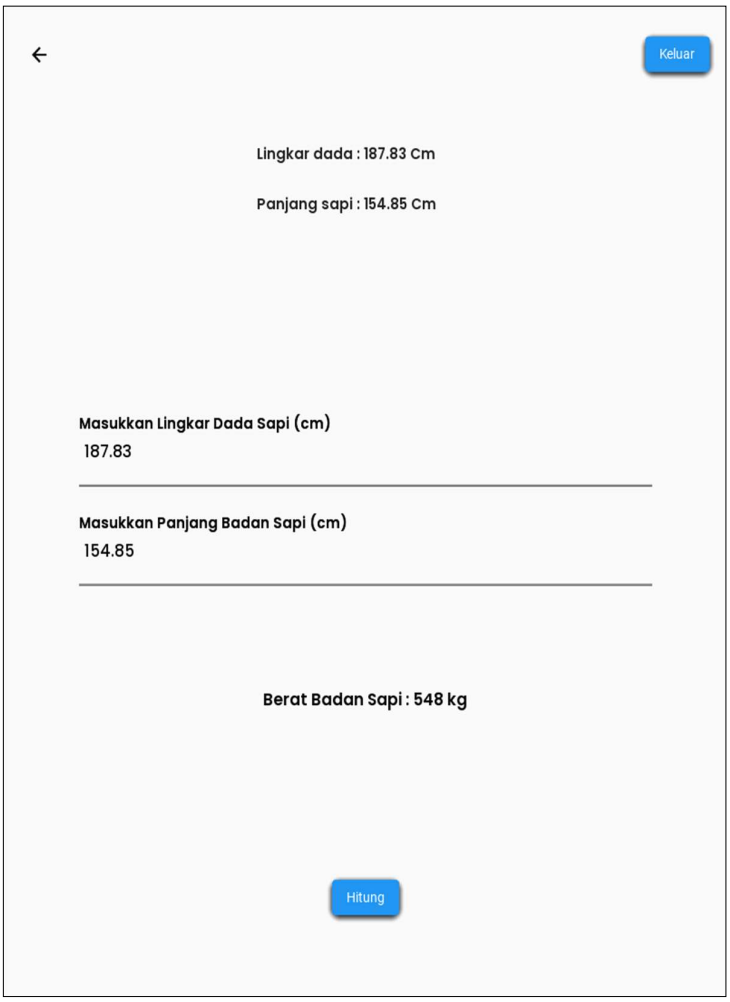
Pada halaman menghitung lingkar dada sapi pengguna bisa memilih gambar sapi yang akan diperoses dengan menekan tombol cari gambar, setelah itu sistem akan menampilkan gambar yang telah dipilih pada layar. Pengguna mengukur lingkar dada sapi dengan memilih area lingkar dada sapi pada gambar, sistem menampilkan hasil pengukuran lingkar dada sapi dengan satuan Cm pada layar.



Gambar 3.31 lingkar dada

e. Halaman penaksiran bobot

Pada halaman penaksiran bobot sapi menampilkan hasil dari pengukuran panjang sapi dan juga hasil pengukuran lingkar dada sapi. Pada halaman ini pengguna diminta untuk memasukan panjang sapi dan lingkar dada sapi yang telah dihitung sebelumnya, setelah itu pengguna bisa menekan tombol hitung untuk mengetahui penaksiran bobot sapi yang dihasilkan, hasil penaksiran bobot sapi ditampilkan pada layar dalam satuan kilogram. Pada halaman ini juga terdapat tombol kelura untuk keluar dari aplikasi.



Gambar 3.32 rumus penaksiran