BAB II
DASAR TEORI DAN TINJAUAN PUSAKA

2.2 Penelitian Terdahulu

Untuk mendapatkan hasil penelitian yang optimal, dilakukan kajian terhadap penelitian-penelitian terdahulu yang dapat dijadikan referensi. Hal ini bertujuan untuk memperoleh perbandingan kelebihan dan kekurangan dari masing-masing perancangan.

Pada penelitian sebelumnya yang dilakukan oleh *Fikriansyah Martunus* (2020) dengan judul "Implementasi Face Recognition Dengan OpenCV pada 'Smart CCTV' Untuk Keamanan Brankas Berbasis IoT", permasalahan utama yang diangkat adalah bagaimana memastikan keamanan brankas dari potensi ancaman, seperti pembobolan atau perusakan, dengan memanfaatkan teknologi modern. Meskipun brankas telah dirancang dengan tingkat proteksi sesuai standar internasional, ancaman terhadap keamanan barang tetap menjadi perhatian serius. Ketiadaan pengawasan yang efektif, terutama saat tidak ada orang di sekitar, menjadi celah yang dapat dimanfaatkan oleh pelaku kejahatan. Oleh karena itu, penelitian ini mengusulkan solusi inovatif yang mampu memantau aktivitas di sekitar brankas secara *real-time*, mengenali wajah orang yang mendekatinya, dan memberikan peringatan jika terdeteksi individu yang tidak dikenal. Namun, tantangan utama dalam penelitian ini adalah mencapai tingkat akurasi tinggi dalam pengenalan wajah serta memastikan sistem berjalan dengan baik agar hasilnya dapat diandalkan.

Selanjutnya, penelitian yang dilakukan oleh *Ahmad Muzakki* (2022) dengan judul "Implementasi Motion Detection Dengan OpenCV pada Raspberry Pi Camera" membahas permasalahan dalam implementasi *motion detection* menggunakan kamera dan *Raspberry Pi*. Sistem ini harus bekerja secara optimal dalam mendeteksi pergerakan, sementara metode *background subtraction* yang digunakan perlu dikalibrasi dengan baik agar dapat membedakan antara pergerakan nyata dan perubahan cahaya atau gangguan lainnya. Selain itu, sistem bergantung pada koneksi internet untuk mengirimkan notifikasi melalui *Telegram*, yang dapat

menjadi kendala jika terjadi gangguan jaringan. Keamanan dan keandalan sistem dalam memberikan notifikasi secara *real-time* juga menjadi tantangan agar pengguna dapat menerima informasi dengan akurat dan tepat waktu.

2.3 OpenCV

OpenCV (*Open Computer Vision*) merupakan sebuah *library* open-source yang menyediakan *API* (*Application Programming Interface*) untuk pemrosesan citra (*image processing*). OpenCV pertama kali dikembangkan oleh Intel pada tahun 1999 oleh Gary Bradsky dan mulai dirilis secara publik pada tahun 2000. Pada tahun 2005, OpenCV berhasil memenangkan DARPA Grand Challenge.

Saat ini, OpenCV mendukung berbagai algoritma yang berkaitan dengan Computer Vision dan Machine Learning. Selain itu, OpenCV juga dapat digunakan dalam berbagai bahasa pemrograman, seperti C++, Python, Java, dan lainnya. OpenCV tersedia di berbagai platform, termasuk Windows, Linux, macOS, Android, dan iOS.

Salah satu keunggulan OpenCV adalah kemampuannya dalam melakukan operasi berkecepatan tinggi (*high-speed*) menggunakan GPU melalui antarmuka berbasis CUDA dan OpenCL. Kombinasi terbaik untuk pemrosesan gambar yang cepat adalah dengan mengintegrasikan OpenCV, *C++ API*, dan bahasa pemrograman Python.

Adapun fitur-fitur utama yang terdapat dalam OpenCV antara lain:

1. Manipulasi data citra – mencakup alokasi, rilis, duplikasi, pengaturan, dan konversi.
2. Dasar pengolahan citra – meliputi penerapan filter, deteksi tepi, deteksi sudut, pengambilan sampel, konversi warna, operasi morfologi, dan analisis histogram.
3. Analisis struktur – mencakup identifikasi komponen yang berhubungan, pengolahan kontur, transformasi jarak, variasi momen, transformasi *Hough*, perkiraan poligonal, dan penyesuaian garis.
4. Analisis gerakan – termasuk *optical flow*, segmentasi gerakan, dan pelacakan objek (*tracking*).

5. Pengenalan objek – menggunakan metode *eigen* dan *Hidden Markov Model (HMM)*.
6. Graphical User Interface (GUI) – mencakup tampilan citra/video, penanganan *mouse* dan *keyboard*, serta penggunaan *scroll bar*.

Modul-modul yang terdapat pada OpenCV antara lain:

1. CV – berisi algoritma untuk pemrosesan citra dan *computer vision*.
2. ML – pustaka (*library*) untuk *machine learning*.
3. Highgui – modul untuk tampilan GUI, serta pemrosesan *image* dan *video I/O*.
4. CXCORE – menyediakan struktur data, dukungan XML, dan fungsi-fungsi grafis.
5. CvAux – modul *open-source* untuk *computer vision* yang dikembangkan oleh Intel Corporation.

2.4 CvZone

CVZone adalah pustaka (*library*) berbasis *OpenCV* yang dirancang untuk mempermudah implementasi teknik Computer Vision. CVZone menyediakan berbagai modul yang telah dikemas secara praktis untuk mendukung deteksi wajah, pose, tangan, serta pemrosesan citra dalam proyek berbasis kecerdasan buatan (AI) dan visi komputer.

- CVZone dibangun di atas OpenCV dan pustaka lain seperti MediaPipe.
- Mempermudah pengolahan citra dengan fungsi yang lebih simpel dibandingkan OpenCV murni.
- Memiliki berbagai modul untuk deteksi wajah, tangan, pose, dan interaksi lainnya.

2.4.1 PoseModule dalam CVZone

PoseModule adalah salah satu komponen dalam *CVZone* yang digunakan untuk Pose Estimation, yaitu deteksi pose tubuh manusia dalam gambar atau video. Modul ini memanfaatkan *MediaPipe Pose* di latar belakang untuk mendeteksi titik-titik tubuh manusia (*landmarks*).

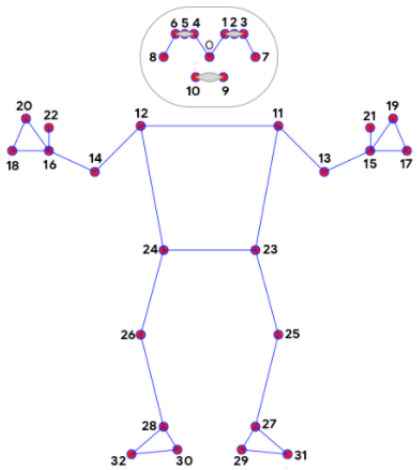
Bagaimana PoseModule bekerja:

1. Menggunakan model *pre-trained* dari MediaPipe Pose untuk mendeteksi 33 titik landmark pada tubuh manusia.
2. Menampilkan bounding box (kotak pembatas) di sekitar tubuh manusia yang terdeteksi.
3. Menyediakan fungsi tambahan, seperti:

   `findPose(img)`: Mendeteksi pose manusia pada gambar.

   `findPosition(img)`: Mengembalikan koordinat *landmark* tubuh dalam gambar.

2.4.2 MediaPipe



Gambar 2. 1 Pose Landmarks

MediaPipe adalah *framework* Machine Learning (ML) dari Google yang digunakan untuk pemrosesan citra secara *real-time*, termasuk Face Detection, Hand Tracking, dan Pose Estimation.

Bagaimana MediaPipe Pose bekerja:

1. Menggunakan model deep learning berbasis Convolutional Neural Network (CNN) untuk mendeteksi 33 titik *landmark* pada tubuh manusia.
2. Memanfaatkan metode BlazePose, yang dikembangkan oleh Google, untuk mendeteksi posisi tubuh dalam bentuk 2D maupun 3D.
3. Memiliki kecepatan dan akurasi tinggi, bahkan pada perangkat dengan spesifikasi rendah.

Hubungan MediaPipe dengan PoseModule dalam CVZone

1. PoseModule dalam CVZone sebenarnya adalah wrapper (pembungkus) dari MediaPipe Pose.
2. CVZone menyederhanakan penggunaan MediaPipe dengan fungsi-fungsi yang lebih intuitif.
3. MediaPipe menangani perhitungan landmark pose, sedangkan CVZone membantu memvisualisasikan dan mengolah data hasil deteksi dengan lebih mudah.

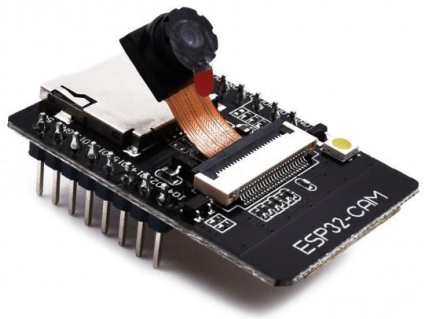
2.5 Pyglet

Pyglet adalah pustaka (*library*) Python yang memungkinkan pengembang membuat aplikasi multimedia, seperti game atau aplikasi visual, dengan dukungan grafis dan audio. Dalam konteks ini, Pyglet digunakan untuk memutar berbagai format file audio, termasuk WAV, MP3, dan Ogg. Pyglet juga mendukung fitur pemutaran audio berulang (looping) serta penundaan pemutaran (delayed playback).

Contoh sederhana untuk pemutaran suara:

```
import pyglet
sound = pyglet.media.load('sound.wav', streaming=False)
sound.play()
pyglet.app.run()
```

2.6 OV2640 With ESP32 CAM

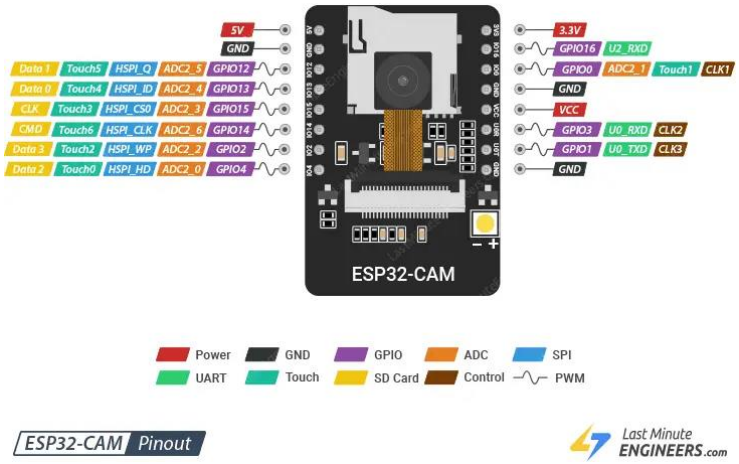


Gambar 2. 2 OV2640 With ESP32 CAM

ESP32-CAM memiliki spesifikasi sebagai berikut:

Tabel 2. 1 Spesifikasi ESP32-CAM

Spesifikasi	Penjelasan
WiFi+Bluetooth module	ESP-32S
RAM	Internal 512KB + External 4MB PSRAM
Modul Kamera	OV2640 2MP
Lampu Kilat (*Flash Light*)	LED bawaan pada papan (*built-in on board*)
Tegangan Operasional	3.3/5 Vdc
Dukungan Kartu microSD	Slot kartu TF bawaan, mendukung hingga 4 GB untuk penyimpanan data
Konsumsi Daya	• *Flash off*: 180mA@5V. • *Flash on and brightness max*: 310mA@5V. • *Deep-Sleep: as low as* 6mA@5V. • *Modern-Sleep: as low as* 20mA@5V. • *Light-Sleep: as low as* 6.7mA@5V
Dimensi	40.5mm x 27mm x 4.5mm



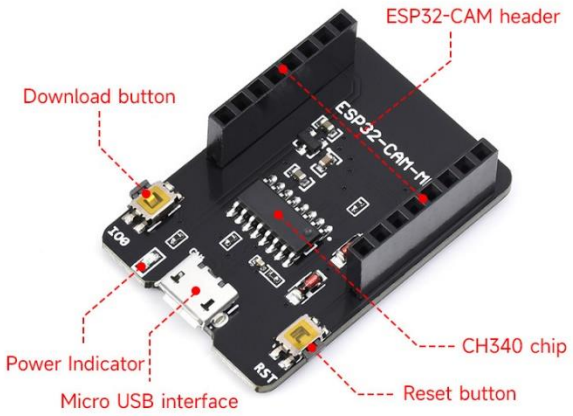
Gambar 2. 3 Pinout ESP32-CAM

Pinout pada ESP32-CAM Sebagai Berikut:

Tabel 2. 2 *Pinout ESP32-CAM*

PIN	Penjelasan
Power Pins	Terdapat dua pin daya: 5V dan 3V3. ESP32-CAM dapat diberi daya melalui pin 3.3V atau 5V. Karena banyak pengguna melaporkan masalah saat memberi daya perangkat menggunakan 3.3V, disarankan agar ESP32-CAM selalu diberi daya melalui pin 5V. Pin VCC biasanya mengeluarkan 3.3V dari regulator tegangan bawaan. Namun, pin ini dapat dikonfigurasi untuk mengeluarkan 5V dengan menggunakan link Zero-ohm di dekat pin VCC
GND	Adalah Ground pin
GPIO	Chip ESP32-S memiliki total 32 pin GPIO, namun karena banyak dari pin tersebut digunakan secara internal untuk kamera dan PSRAM, ESP32-CAM hanya memiliki 10 pin GPIO yang tersedia. Pin-pin ini dapat diberikan berbagai tugas perifer, seperti UART, SPI, ADC, dan Touch.

UART	Pins Chip ESP32-S sebenarnya memiliki dua *interface* UART, yaitu UART0 dan UART2. Namun, hanya pin RX (GPIO 16) dari UART2 yang tersedia, sehingga UART0 menjadi satu-satunya UART yang dapat digunakan pada ESP32-CAM (GPIO 1 dan GPIO 3). Selain itu, karena ESP32-CAM tidak memiliki port USB, pin-pin ini harus digunakan untuk flashing serta menghubungkan perangkat UART seperti GPS, sensor sidik jari, sensor jarak, dan sebagainya.
MicroSD Card Pins	Pin-pin ini digunakan untuk berinteraksi dengan kartu microSD. Jika Anda tidak menggunakan kartu microSD, Anda bisa menggunakan pin-pin ini sebagai input dan output biasa.
ADC Pins	Pada ESP32-CAM, hanya pin ADC2 yang tersedia. Namun, karena pin ADC2 digunakan secara internal oleh driver Wi-Fi, pin-pin ini tidak dapat digunakan ketika Wi-Fi diaktifkan.
Touch Pins	ESP32-CAM memiliki 7 GPIO sensitif sentuhan kapasitif. Ketika muatan kapasitif (seperti jari manusia) berada dalam jarak dekat dengan GPIO, ESP32 mendeteksi perubahan kapasitansi tersebut.
SPI Pins	ESP32-CAM hanya memiliki satu SPI (VSPI) yang dapat digunakan dalam mode slave dan master.
PWM Pins	ESP32-CAM memiliki 10 saluran (semua pin GPIO) pin PWM yang dikendalikan oleh pengontrol PWM. Output PWM ini dapat digunakan untuk menggerakkan motor digital dan LED.



Gambar 2. 4 ESP-32 CAM Adapter

CH340 berperan sebagai *USB-to-Serial converter* yang memungkinkan ESP32-CAM untuk berkomunikasi dengan komputer melalui USB. Karena ESP32-CAM tidak memiliki USB *interface* bawaan, maka modul CH340 (seperti CH340G, CH340C, atau CH340T) sering digunakan untuk:

1. *Mem-flash firmware* ke ESP32-CAM, dengan menyediakan USB-to-UART interface untuk mengunggah kode langsung dari Arduino IDE atau platform lsainnya.
2. Menyediakan komunikasi serial untuk *debugging* dan monitoring, sehingga *log* dan data dari ESP32-CAM dapat dikirim ke komputer melalui *serial interface.*
3. Menyederhanakan koneksi, karena CH340 menghilangkan kebutuhan akan konverter eksternal seperti FTDI atau CP2102.

ESP32-CAM tidak memiliki chip *USB-to-Serial* bawaan, sehingga memerlukan CH340 module atau FTDI adapter secara eksternal untuk pemrograman dan komunikasi.

2.7 MJPEG

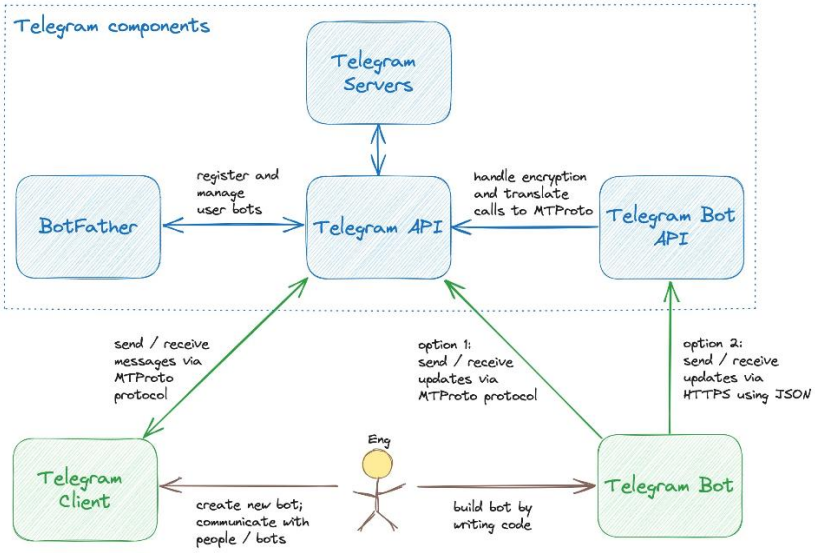
MJPEG (Motion JPEG) adalah format video yang terdiri dari rangkaian gambar JPEG (Joint Photographic Experts Group) yang dikirimkan secara berurutan untuk membentuk video.

Sederhananya, MJPEG adalah sekumpulan foto JPEG yang ditampilkan secara cepat sehingga terlihat seperti video.

Cara Kerja MJPEG

1. Kamera menangkap gambar (frame) dalam format JPEG
2. Setiap gambar dikirim ke klien (browser, aplikasi, VLC, dll.)
3. Klien menampilkan gambar secara berurutan dengan kecepatan tinggi
4. Mata manusia menangkapnya sebagai video bergerak

2.8 Bot Telegram



Gambar 2. 5 Telegram system components

Telegram adalah aplikasi pesan instan berbasis *cloud* yang memungkinkan pengguna untuk mengirim pesan teks, suara, gambar, video, serta berbagai jenis dokumen dengan cepat dan aman. Aplikasi ini dikembangkan oleh Pavel Durov dan pertama kali dirilis pada tahun 2013. Telegram memiliki fitur enkripsi *end-to-end* untuk percakapan rahasia serta mendukung pembuatan grup dengan anggota dalam jumlah besar, *bot* otomatis, serta integrasi *API* untuk keperluan pengembangan.

1. Telegram API

   API yang memungkinkan pengembang membuat *bot* otomatis untuk berbagai fungsi, seperti menjawab pesan atau mengelola grup.

   . Cara Kerja Komponen Sistem Telegram Bot

1) Pengguna mengirim pesan ke *bot*.
2) Telegram Server meneruskan pesan ke *endpoint bot*.
3) Bot memproses pesan dan menjalankan perintah.
4) *Bot* mengirim respons kembali melalui Telegram Server.
5) Pengguna menerima balasan dari *bot*.

2. Telegram *Client*

   Aplikasi yang digunakan pengguna untuk mengakses Telegram, tersedia di berbagai platform.

3. Telegram Server

   Server berbasis *cloud* yang menyimpan dan mengelola pesan serta menangani pengiriman secara *real-time*.

4. MTProto Protocol (Mobile Transport Protocol)

   MTProto adalah protokol komunikasi yang dikembangkan oleh Telegram untuk meningkatkan kecepatan dan keamanan dalam pengiriman pesan. Protokol ini dirancang untuk mengoptimalkan penggunaan *bandwidth* dan melindungi data dari serangan siber.

5. Metode Pengiriman yang disediakan oleh Telegram *Bot API*. Ada beberapa metode yang dapat digunakan untuk merancang sebuah *Bot* di Telegram. Untuk program yang dibutuhkan saat ini adalah:

   a. *sendPhoto* — gunakan *method* ini untuk mengirim foto.

   Untuk berkomunikasi dengan Telegram, *bot* menggunakan *URL API Telegram* yang dibentuk sebagai berikut:

   https://api.telegram.org/bot/*TOKEN*/*METHOD*

- *TOKEN*: Token yang diberikan oleh BotFather.
- *METHOD*: *Method API* yang digunakan, seperti *sendMessage*, *sendPhoto*, atau lainnya.

2.8.1 python-telegram-bot

python-telegram-bot adalah pustaka (library) Python yang digunakan untuk berinteraksi dengan Telegram Bot *API*. Pustaka ini mempermudah pembuatan dan pengelolaan *bot* Telegram tanpa harus melakukan permintaan *HTTP* secara manual. Dengan menggunakan python-telegram-bot, pengembang dapat membuat *bot* yang mampu menerima pesan, mengirim teks, gambar, video, serta merespons perintah tertentu dari pengguna.

CaraKerjapython-telegram-bot, *Bot* Telegram bekerja berdasarkan *API* Telegram, di mana setiap komunikasi antara *bot* dan pengguna dilakukan melalui permintaan *HTTP*. python-telegram-bot menyederhanakan komunikasi ini dengan menyediakan *wrapper* yang lebih mudah digunakan. Untuk program yang dibutuhkan saat ini adalah:

1. Autentikasi dengan Token *API*: *Bot* harus memiliki token *API* yang diperoleh dari BotFather untuk mengakses *API* Telegram.
2. Polling atau *Webhook*: *Bot* dapat menerima pesan dari pengguna menggunakan dua metode:
   - *Polling*: *Bot* terus menerus mengecek pembaruan dari server Telegram.
   - *Webhook*: Server Telegram mengirimkan data langsung ke *endpoint* yang telah ditentukan.
3. Dispatcher & Handler: Pesan yang diterima oleh *bot* akan diproses oleh *dispatcher*, yang akan meneruskannya ke *handler* yang sesuai (misalnya perintah /start akan diproses oleh *CommandHandler*('start', *fungsi_start*)).
4. Bot Memberikan Respons: *Bot* dapat mengirim teks, gambar, video, atau file lainnya sebagai respons terhadap interaksi pengguna.

Program diatas cara kerja bot Telegram hanya menggunakan autentikasi dengan Token API dan mengirim gambar menggunakan metode *HTTP request* dengan *library requests*.

2.8.2 Library Requests

Requests adalah pustaka (*library*) Python yang digunakan untuk melakukan permintaan *HTTP* (*HTTP request*) dengan cara yang sederhana dan efisien. *Library* ini sering digunakan untuk berkomunikasi dengan *API* web, termasuk Telegram Bot *API*, *RESTful API*, dan berbagai layanan berbasis web.

Library requests menyediakan berbagai metode *HTTP*, seperti:

- GET : Mengambil data dari server.
- POST : Mengirim data ke server.
- PUT : Memperbarui data di server.
- DELETE : Menghapus data di server.

Dalam konteks program kali ini, requests digunakan untuk mengirim gambar ke Telegram Bot API melalui metode POST