Technical Journal RSC Aksantara 2026

Muhammad Ridwan Nasir Firdaus

Januari – Februari 2026

Daftar Isi

1 Kamis, 29 Januari 2026 **2**
1.1 Pengenalan RSC . . . 2
1.2 Arsitektur Sistem UAV . . . 2
1.2.1 Wahana . . . 2
1.2.2 Avionics . . . 2
1.2.3 Payload . . . 2
1.2.4 Sistem Komunikasi Radio . . . 2
1.2.5 Companion Computer (CC) . . . 2
1.3 Object Oriented Programming . . . 3
1.4 Tantangan . . . 3

2 Jumat, 30 Januari 2026 **3**
2.1 MAVLink . . . 3
2.1.1 Struktur . . . 3
2.1.2 Messages . . . 4
2.1.3 MAVProxy . . . 4
2.2 SITL . . . 4
2.3 Tantangan . . . 5

3 Minggu, 1 Februari 2026 **5**
3.1 Computer Vision . . . 5
3.1.1 Representasi Citra . . . 5
3.1.2 Operasi Neighborhood . . . 6
3.1.3 Thresholding dan Segmentasi . . . 6
3.2 AI Dalam Computer Vision . . . 7
3.2.1 Machine Learning . . . 7
3.3 Deep Learning . . . 8
3.4 Tantangan . . . 9

4 Senin, 2 Februari 2026 **9**
4.1 Hands-On 1 . . . 9

5 Selasa, 3 Februari 2026 **9**
5.1 THT 2 . . . 9

1 Kamis, 29 Januari 2026

1.1 Pengenalan RSC

RSC adalah departemen dalam Aksantara yang mengelola seluruh aspek terkait dengan perangkat lunak (*software*). Dalam RSC terdapat dua keilmuan, yaitu *Control and Perception* (ConCept) dan *Ground Control Station* (GCS).

Control and Perception Berfokus dalam pengembangan sistem navigasi untuk otonom dengan menggunakan algoritma tingkat tinggi berbasis ROS2 dan pengembangan persepsi dengan *Computer Vision*.

Ground Control Station Berfokus dalam pengembangan *software* Ground Control Station internal.

1.2 Arsitektur Sistem UAV

1.2.1 Wahana

- **Airframe**: Struktur utama drone yang terdiri atas *Fuselage*/badan, ekor, dan sayap untuk FW, sedangkan untuk VTOL terdiri atas *frame center* dan *arm*.
- **Sistem Propulsi**: Terdiri dari motor, propeler, *Electronic Speed Controller*(ESC), dan baterai LiPo.

1.2.2 Avionics

- **Flight Controller (FC)**: Berperan sebagai saraf motorik dari UAV yang memproses dan menyatukan data dari sensor serta untuk menjalankan algoritma kontrol penerbangan. Dalam FC terdapat komponen lain seperti MCU, barometer, IMU, dan *blackbox*.
- **Modul GNSS**: Menerima sinyal dari sistem satelit seperti GPS, Galileo, dan BeiDou untuk menentukan lokasi, mengukur kecepatan, dan waktu UAV.
- **Airspeed Sensor**: Menghitung kecepatan relatif wahana terhadap kecepatan udara.

1.2.3 Payload

Merupakan komponen yang diangkut oleh drone dan tidak esensial untuk keperluan penerbangan. Komponennya seperti kamera fpv, kamera tracking, LiDar, kamera depth, dan sensor partikel.

1.2.4 Sistem Komunikasi Radio

Komunikasi antara UAV dengan GCS adalah gelombang radio dalam frekuensi tertentu. Jalur komunikasi fungsional yang terbentuk dari pertukaran data lewat frekuensi disebut dengan *link*. Terdapat dua *link* utama dalam sistem UAV, yaitu:

- **Non-Payload Link**: Saluran utama yang bertanggunang jawab atas pengendalian UAV. Sistem ini terjalin antara *radio tranceiver* GCS dengan UAV.
- **Payload Link**: Saluran untuk mengirim data *payload* antara GCS dengan UAV.

1.2.5 Companion Computer (CC)

CC digunakan untuk menangani masalah komputasi tingkat tinggi yang harus dilakukan secara real-time dan melengkapi FC sebagai sistem saraf motorik dari UAV.

1.3 Object Oriented Programming

Di OOP sendiri aku kembali recall ke hal-hal dasar seperti apa itu class dan object. Class itu simpelnya seperti sebuah cetakan *blueprint* dan objek adalah barang hasil cetakan tersebut. Selain itu, di dalam OOP terdapat empat pilar dasar, yaitu *Encapsulation*, *Abstraction*, *Inheritance*, dan *Polymorphism*.

1.4 Tantangan

Materi *multithreading* lumayan sulit (menurutku) karena disini kita bener-bener diminta untuk buat code seaman mungkin dalam ngakses suatu *shared resource* dan disini rawan banget buat terjadi masalah semacam *deadlock*, *racing condition*, dll.

2 Jumat, 30 Januari 2026

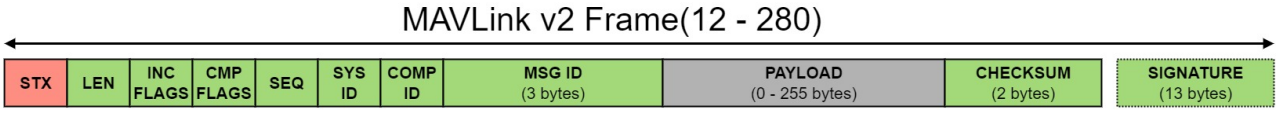
Di hari ini aku mendapat materi mengenai MAVLink dan SITL. MAVLink itu adalah sebuah protokol untuk proses komunikasi antar drone, GSC, dan FC. Salah satu keunggulan mavlink itu protokolnya ringan dan efisien. MAVLink sendiri punya struktur data (format) sendiri dan yang menarik perhatianku adalah pada bagian *signature*, ternyata protokol MAVLink memiliki langkah preventif untuk mencegah spoofing dan modifikasi paket data. Dan diakhir meeting ofkors selalu ada oleh-oleh (aku mw lima). Oleh-olehnya cukup simpel dan seru yaitu melakukan simulasi drone di SITL dan Gazebo dengan drone bergerak mengikuti *pattern* angka delapan. Jadi ada dua metode, pertama dengan membuat manual *waypoint* dan kedua dengan menggunakan *script*. Buat metode pertama itu bisa dibilang *easy* banget, ya karena cuma ngeklik-klik doang. Tapi, buat metode kedua itu lumayan menantang. Pertama aku harus cari tahu dulu fungsi yang bisa membentuk pola angka 8 apa dan nama fungsi itu adalah **Lemniscate of Gerono**.

2.1 MAVLink

MAVLink atau *Mictro Air Vehicle Link* adalah sebuah protokol akomunikasi yang digunakan untuk pertukuran informasi antar drone, GCS, *flight controller*, dan perangkat lainnya. Informasi yang dikirim dibungkus dalam bentuk *message*. MAVLink sendiri memiliki beberapa keunggulan, seprti protokolnya yang ringan dan efisien dikarenanan *size overhead*-nya kecil, tidak memerlukan *framing* tambahan, *reliable* untuk komunikasi antar berbagai jenis wahana, GSC, dan node lain pada kondisi yang beragam dan menantang. Dan terakhir MAVLink *support* banyak bahasa pemrograman dan dapat berjalan di berbagai mikrokontoler/OS.

2.1.1 Struktur

Paket pesan MAVLink memeliki struktur sebagai berikut.



Gambar 1: Struktur Paket Pesan MAVLink

- STX / awal paket: berisi *magic number* dalam basis heksadesimal untuk membedakannya dengan protolol lain.
- LEN: Panjang *payload*, *range*-nya antara 1 hingga 255.
- `INC_FLAGS`/*Incompatibility Flags*: Menandai fitur mana yang wajib didukung *receiver*, jika tidak paket akan ditolak.

- `COMP_FLAGS`/*Compatibility Flags*: Menandai fitur tambahan yang sifatnya opsional. Jika *receiver* tidak mengenali flag ini, maka paket tetap boleh untuk diproses.
- `SEQ`/*Sequence Number*: Nomor urut paket yang digunakan untuk mendeteksi *packet loss*. Nilainya da apada range 0 hingga 255 dan akan kembali menjadi 0 jika telah mencapai batas.
- `SYS_ID`/*System ID*: Menunjukkan identitas sistem pengirim pesan.
- `COMP_ID`/*Component ID*: Menunjukkan identitas komponen dalam suatu sistem.
- `MSG_ID`/*Message ID*: Menunjukkan identitas jenis pesan yang dikirim. MAVLink 2 memiliki *size* 3 byte.
- Payload: Berisi data utama dari pesa sesuai dengan `MSG_ID`. Panjang payload ditentukan oleh nilai LEN dan data disusun dalam format *little-endian*.
- Checksum: Digunakan untuk memverifikasi integritas data. Memiliki *size* 2 btye, checksum dihitung dari head (kecuali STX), payload, dan CRC extra.
- Signature: Digunakan jika fitur *message signing* diaktifkan. Fungsinya untuk meningkatkan keamanan komunikasi dengan mencegah *spoofing* dan modifikasi paket.

2.1.2 Messages

MAVLink memiliki berbagai jenis *messages* yang digunakan untuk pertukuran informasi. Setiap *messages* memiliki ID unik dan meiliki struktur *payload*-nya sendiir. Beberapa Message ID yang umum digunakan dalam MAVLink ialah sebagai berikut:

Message Name	Message ID	Fungsi
HEARTBEAT	0	Menandakan bahwa sistem masih aktif dan mengirimkan status dasar sistem
SYS_STATUS	1	Menyediakan informasi status sistem seperti kesehatan sensor dan daya
SYSTEM_TIME	2	Menyinkronkan waktu sistem
PING	4	Menguji konektivitas dan latensi komunikasi
GPS_RAW_INT	24	Data mentah GPS

2.1.3 MAVProxy

Merupakan tool berbasis CLI berfungsi sebagai router, relay, dan controller untuk komunikasi MAVLink antara FC dan berbagai aplikasi lain. MAVProxy memiliki fungsi yang mirip dengan GCS pada umumnya. MAVProxy dapat melakukan *arm*, *disarm*, mengubah *flight mode*, *upload*, dan eksekusi misi. Tapi MAVLink tidak dapat melakukan fungsi-fungsi pada GCS yang berkaitan dengan GUI misalnya menampilkan peta interaktif. Beberapa keuntungan dari MAVProxy, yaitu dapat berjalan di Linux, *lightweight*, dan stabil untuk *long running*.

2.2 SITL

SITL atay *Software In The Loop* yang digunakan adalah ArduPilot sebagai *flight stack* utama dan ArduPilot merupakan *flight stack* yang secara *de facto* digunakan di Aksantara ITB. Fungsi dari SITL adalah untuk menjalankan dan menguju sistem autopilot drone tanpa menggunakan *hardware* fisik. ArduPilot berperan sebagai otak drone, yang menjalankan logika penerbangan.

2.3 Tantangan

Karena aku pengguna **Jendela** dan belum punya niat untuk *dual boot* jadi aku hanya menggunakan vm untuk menjalankan sistem operasi Ubuntu. Tapi waktu aku ngejalanin simulasi di Gazebo, aduhai *lag* parah cuyyy... Ini gak tau emang laptop aku yang kentang apa gimana. Alhasil aku terpaksa untuk *dual boot*, tapi eits.... gak semudah itu, waktu pengen partisi *storage* ternyata cuma bisa ngepartisi 20GB padahal *free space* di SSD ku masih ada 170 GB (emang si jendela ini ringkih ygy). Setelah ngotak-ngatik *settingan* dfrag, *cache*, dll. *Storage* yang bisa akhirnya nambah... 5GB doang (NGUWAWOR CIK) dan *in the end* terpaksa make *third party software*.

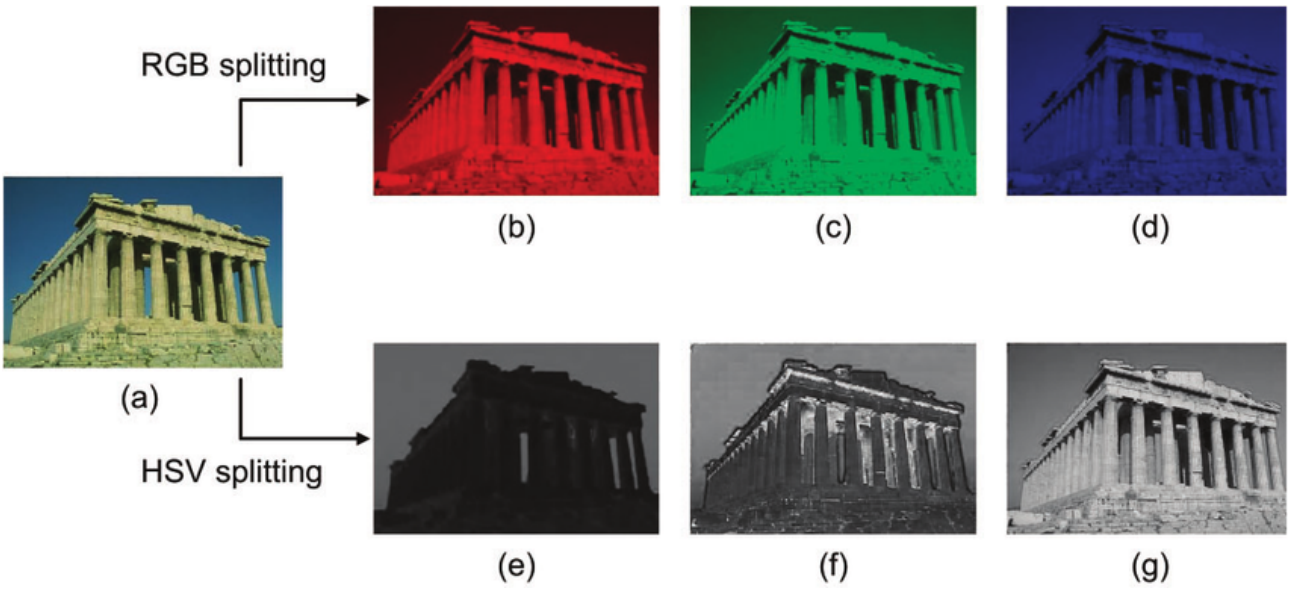
3 Minggu, 1 Februari 2026

Materi hari ini mantep banget, aku baru tau kalo ternyata selama ini RGB, HSV, CMYK, dll itu merupakan representasi ruang citra dalam bentuk matriks. Terus yang gak kalah menarik adalah operasi *Neighborhood* seperti *convulution* dan morfologi. *connvulution* sendiri ialah operasi yang dilakukan antara matriks kecil (*kernel*) dengan setiap piksel dan nilai tetannga dari suatu citra, salah satu penggunaanya adalah *edge detection*. Lalu ada morfologi yang biasa digunakan pada pengolahan citra biner (hitam putih) untuk memanipulasi bentuk objek dalam citra.

3.1 Computer Vision

3.1.1 Representasi Citra

Citra atau gambar tersusun atas piksel-piksel. Sebuah citra dapat direpresentasikan sebagai sebuah matriks berukuran tinggi(H) x lebar(W), yang mana setiap elemen dalam matriks akan mewakili intensitas cahaya dalam piksel tertenu. Contoh pada citra *grayscale*, setiap piksel memiliki nilai intensitas dengan *ranage* 0 (hitam) hingga 255 (putih). Sedangkan untuk citra berwarna seperti RGB, tiap pikselnya direpresentasikan oleh tiga nilai intensitas dari warna dasar (merah, biru, hijau). Citra dapat dianggap sebagai tumpukan dari 3 matriks, tiap matriks mewakili saluran (*channel*) warna.



Gambar 2: Representasi Citra Dalam RGB dan HSV

Dan contoh lainnya adalah citra HSV (Hue, Saturation, Value) yang memisahkan informasi warna (hue), saturasi, dan kecerahan (value).

3.1.2 Operasi Neighborhood

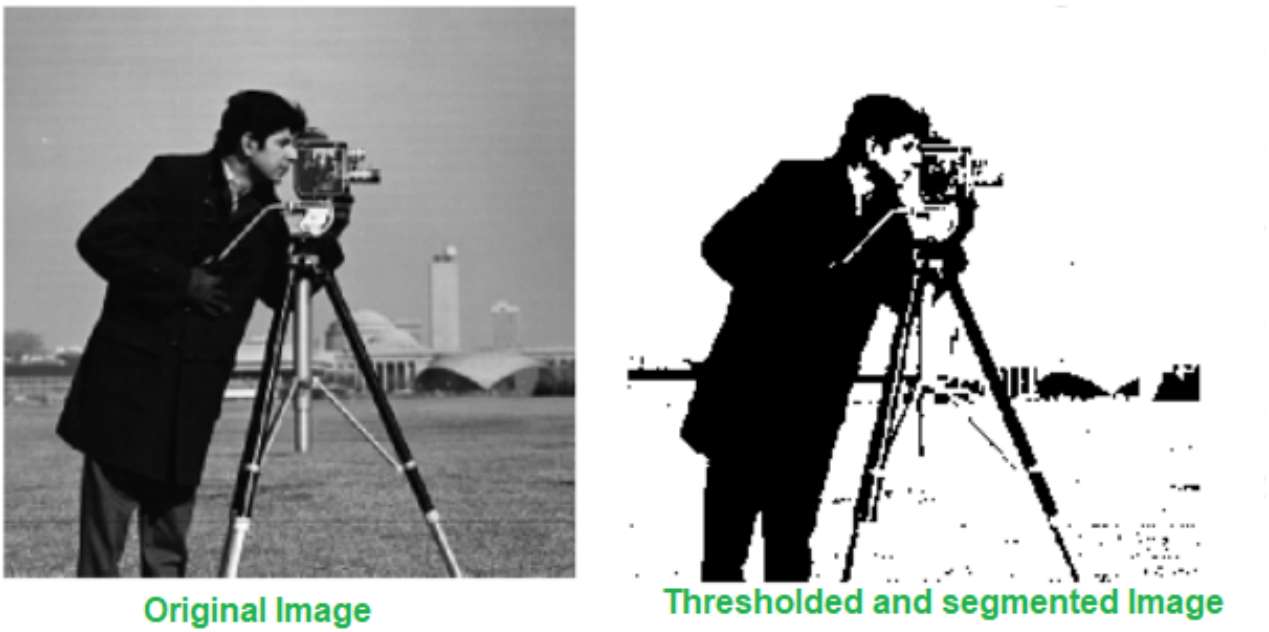
Merupakan teknis dasar dalam pengolahan citra digital yang melibatkan manipulasi piksel berdasarkan nilai piksel disekitarnya. Karena banayak informasi penting dalam citra, seperti *edge*, teksture, dan pola dapat diidentifikasi dengan melihat hubungan antar piksel yang berdekatan.

Convolution merupakan bentuk operasi *neighborhood* dengan menggunakan kernel. Kernel adalah matriks dengan ukuran kecil yang diterapkan pada setiap piksel citra untuk menghasilkan nilai baru berdasarkan nilai piksel tetangganya. Kernel akan digeser melintasi citra dan pada setiap posisi, nilai baru dihitung dengan menjumlhkan hasil kali elemen kernel dengan nilai piksel yang sesuai dibawahnya. Salah satu aplikasi *convolution* adalah dalam *edge detection*, yang mana konsepnya kernel melihat perubahan drastis dalam intensitas piksel yang menandakan tepi objek dalam citra. Algortima Sobel merupaak contoh dari *convulution* untuk mendeteksi tepi secaar horizontal dan vertikel dengan menggunakan dua kernel berbeda. Selain untuk *edge detection*, *convolution* juga bisa diterapkan untuk pengaburan, penajaman, dan peningkatan kontras.

Morfologi digunakan dalam pengolahan citra biner (hitam putih) untuk memanipulasi bentuk objek dalam citra. Operasi dasar dalam morfologi adalah *erosion* dan *dilation*. *Erosion* akan menguragi ukuran objek dengan cara menghapus bagian piksel di tepi objek, sedangkan *dilation* merupakan kebalikannya yaitu memperbesar ukurang objek dengan menambah piksel di tepi objek.

3.1.3 Thresholding dan Segmentasi

Thresholding merupakan teknik sederhana tetapi efektif untuk memisahkan objek dari latar belakang dalam citra. Dengan menetapkan *threshold* tertentu, kita dapat mengubah citra *grayscale* menajdi citra biner, tiap piksel dengan nilai diatas batas akan berubah menjadi putih dan yang dibwah akan menjadi hytam.



Gambar 3: Contoh Dari Thresholding dan Segmentasi

Citra hasil *thresholding* akan digunakan untuk segmentasi objek. Beberapa teknik segmentasi lainnya melibatkan beberapa metode, seperti *k-means clustering*, *region growing*, dan *graph-based segmentation*.

3.2 AI Dalam Computer Vision

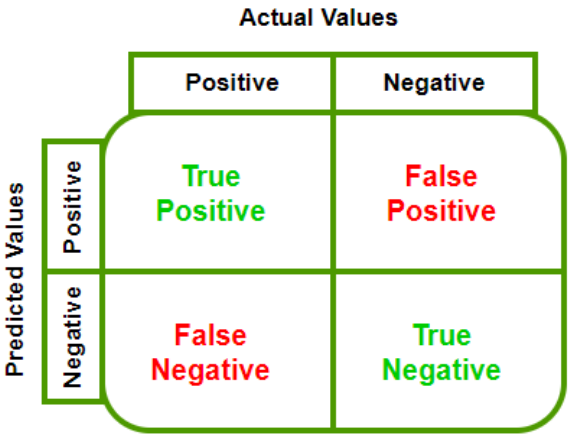
3.2.1 Machine Learning

Merupakan cabang dari *artificial intelligence* yang berfokus pada pengembangan algoritma dan model yang memungkinkan komputer untuk belajar dari data dan membuat prediksi tanpa perlu diprogram secara eksplisit. Dalam *computer vision*, *machine learning* digunakan untuk mengajarkan komputer dalam mengali *pattern*, objek, dan fitur dalam gambar atau video. Dalam *machine learning*, model dilatih menggunakan dataset yang beri contoh input (*feature*) dan ouput(*target).* Proses pelatihan melibatkan penyesuaian parameter model agar dapat meminimalkan *loss function* pada hasil prediksi model. Dalam *machine learning* terdapat beberapa pendekatan utama, yaitu:

- *Supervised Learning*: Pada pendekatan ini, model dilatih menggunakan data yang sudah diberi label. Jadi model akan diberi tahu input dan ouput dari masing-masing data. Contoh penggunaanya ada dalam klasifikasi gambar, di mana model belajar untuk mengenali suatu objek berdasar label yang telah diberi saat proses *training.*
- *Unsupervised Learning*: Pada pendekatan ini, model dilatih dengan menggunakan data yang tidak diberi label, sehingga model harus menemukan *pattern* dari tiap-tiap data. Contoh aplikasinya adalah *clustering*, model akan mengelompokkan data/gambar berdasar kesamaan fitur tanpa mengetahui gambar apa itu sebenarnya.
- *Reinforcement Learning*: Model akan belajar untuk membuat suatu keputusan dengan berinteraksi dengan lingkungan dan menerima *feedback* berupa *prize* jikas berhasil atau *pusnishment* jika gagal. Contohnya adalah *agent* dalam video game, di mana *agent* mempelajari strategi terbaik untuk memenangkan berdasarkan hasil dari tindakannya.

Model Evaluation Digunakan untuk mengetahui seberapa bagus model yang dibuat. Terdapat beberapa metrik yang umum digunakan untuk menilai model *machine learning* dalam *task computer vision*, seperti klasifikasi dan deteksi objek.

Metrik Evaluasi Klasifikasi Ketika model melakukan prediksi hasilnya tentu bisa benar atau salah. Dapat dibuat *confusion matrix* yang menujukkan jumlah benar atau salah untuk tiap kelas.



Gambar 4: Confusion Matrix

Dari *confusion matrix* bisa dihitung metrik-metrik penting, seperti accuracy, precision, recall, dan f1-score.

1. *Accuracy*: Mengukur seberapa akurat model dalam melakukan prediksi. Rumusnya:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. *Precision*: Mengukur tingkat ketepatn prediksi postif objek. Rumusnya:

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. *Recall*: Mengukur tingkat keberhasilan model dalam menemukan dala positif. Rumusnya:

$$\text{Recall} = \frac{TP}{TP + FN}$$

4. *F1-score*: Harmonic mean dari precision dan recall. Rumusnya:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metrik Evaluasi Deteksi dan Segmentasi Objek Metrik yabngn sering digunakan yaitu *Intersection Over Union* (IoU) dan *Mean Average Precision* (mAP) untuk deteksi dan *Dice Coefficient* untuk segmentasi.

1. *Intesection over Union*: Mengukur seberapa baik prediksi *bounding box* dari model tumpang tindih dengan *ground truth bounding box*. Rumusnya

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

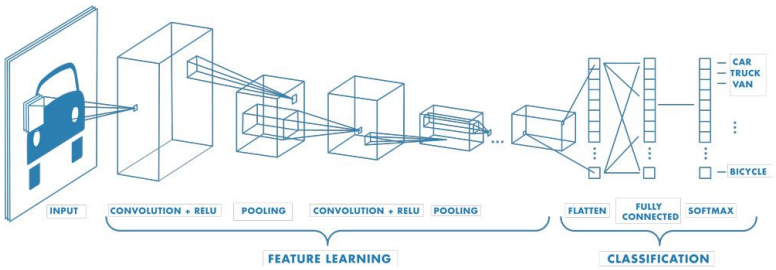
2. *Mean Average Precision*: Metrik penggabungan antara recall dan precision dalam berbagai *threshold* IoU. mAP dihtitung dengan mengambil rata-rata dari *Average Precision* untuk setiap kelas objek.

3. *Dice Coefficient*: Metrik yang digunakan untuk mengukur kesamaan antara dua set, sering digunakan dalam segmentasi citra. Rumusnya

$$\text{Dice} = \frac{2 \times |A \cap B|}{|A| + |B|}$$

3.3 Deep Learning

Deep Learning adalah subset dari *Machine Learning* yang menggunakan jaringan saraf tiruan dengan banyak lapisan untuk mempelajari representasi data yang kompleks. Salah satu arsitekturnya yang populer untuk tugas *computer vision* adalah *Convulational Neural Network* (CNN). CNN dirancang untuk memproses data yang memiliki strukutur *grid*, seperti gambar dan sangat efektif dalam mengenali *pattern* visual. CNN memahami hubungan spasial antar piksel melalui *convolution matrix*.



Gambar 5: Visualisasi Dari Convolutional Neural Network

Konsep utama dalam CNN adalah *convolutional layer* yang menerapakan filter (kernel) pada gambar input untuk mengekstrak fitur penting seperti tepi, sudut, dan tekstur. *Pooling layer* kemudian digunakan untuk mereduksi dimensi data tapi tetap mempertahankan informasi penting, sehinga beban komputasi berkurang dan mencegah *overfitting*. Kernel digeser melewati seluruh gambar untuk menghasilkan *feature maps* yang meng-*highlight* are-area penting. Setelah beberapa pengulagan *convulational dan pooling layees*, data biasanya akan diratakan (*flatten*) dan diteruskan ke *dense later* yang melakukan klasifikasi berdasar fitur yang telah di ekstraksi

3.4 Tantangan

Materinya menarik banget, tapi lumayan *overwhelmed* dengan materi-materi yang harus dipelajari (banyak buanget). Tapi gapapa seru soalnya :D

4 Senin, 2 Februari 2026

4.1 Hands-On 1

Di hari ini aku udah mulai nyicil Hands-On 1 dan jujur ini susah... Ini bukan tipe Hands-On yang sekali baca solusi atau bentukan *code*-nya udah bisa kepikiran. Tapi Hands-On ini perlu dibaca, dipahami, ditelaah berulang-ulang biar bisa mahamin soal. Ditambah dengan aku yang belum pernah belajar mengenai *design pattern*, jadi lumayan bingung harus mulai tugas ini dari mana. Setelah mahamin *design pattern*-nya (*Observer dan Strategy pattern*), aku mulai ngerti gimana *data flow* antara drone dengan si *dashboard*. Dan yak ini buat satu tugas ini aku nge-grind dari bangun tidur sampe malam.

5 Selasa, 3 Februari 2026

5.1 THT 2

Distaym kita ngerjain THT, aku ngeliat soal-soal THT 2 ini menarik banget terutama implementasi *path planning* dan *swarm drone*. Pada *swarm drone* menghindari *obstacle* adalah hal yang wajib untuk dilakukan, sesuai dengan *paper* yang diberikan untuk menghindari *obstacle* digunakan metode *adaptive artificial potential field*. Metode ini menganggap sebuah *obstacle* sebagai titik yang memberi gaya tolak ke objek/drone yang meningkat berbanding terbalik dengan kuadrat jarak antara *obstacle* dengan objek. Objek akan bergerak menujut arah dari total gaya tolak yang dirasakannya. Rumus yang digunakan untuk menghitung gaya tolak antara objek i dengan vektor posisi $\vec{p_i} = (\vec{x_i}, \vec{y_i})$ dan obstacle o dengan vektor posisi $\vec{p_o} = (\vec{x_o}, \vec{y_o})$ yaitu:

$$\overrightarrow{F_{rep,i}} = \begin{cases} \eta(\frac{1}{d_{oi}} - \frac{1}{d_T})^2 \cdot \frac{\vec{p_o} - \vec{p_i}}{|\vec{p_o} - \vec{p_i}|}, & d_o i \leq d_T \\ 0, & d_o i > d_T \end{cases}$$

dimana η adalah konstanta bilangan ril, d_{oi} adalah jarak *euclidean* antara objek dengan *obstacle* dan d_T adalah jarak maksimum dimana gaya tolak antara drone dengan *obstacle* bekerja. Total gaya tolak bisa dinotasikan dengan

$$\overrightarrow{F_{rep}} = \sum \overrightarrow{F_{rep,i}}$$

Lalu untuk drone swarm diterapkan *Boids Flocking Algorithm*, terdapat tiga aturan fundamental untuk mensimulasikan perilaku *flocking*, yaitu *separation*, *alignment*, dan *cohesion*. Aturan *separation* mencegah drone untuk bertabrakan satu sama lain. Misal drone swarm dengan n buah drone, maka aturan ini dapat diberlakukan dengan

$$\vec{v_s} = \sum_{i=1}^{n} \frac{\vec{x_0} - \vec{x_i}}{|\vec{x_o} - \vec{x_i}|^2}$$

Lalu, aturan *alignment* memastikan swarm drone untuk bergerak menuju arah yang sama. Aturan ini bisa direpresentasikan sebagai

$$\vec{v_s} = \frac{1}{n} \sum_{i=1}^{n} \vec{v_i}$$

Dan aturan *cohesion* memastikan drone untuk berdekatan satu sama lain dan drone bergerak menuju pusat swarm. Aturan ini bisa direpresentasikan sebagai berikut

$$\vec{v_c} = \frac{1}{n} \sum_{i=1}^{n} \vec{x_i}$$

Resultan kecepatan dari drone adalah

$$\vec{v} = w_s \vec{v_s} + w_a \vec{v_a} + w_c \vec{v_c}$$

Dimana w_a, w_b, dan w_c adalah konstanta yang ditentukan saat proses *tuning*. Saat implementasinya dalam programnya rasanya sulit-sulit tapi seru gitu dan sesekali aku ngintip + belajar dari repo Kak Nayaka (izin ya kak hehe). Tantangan dalam tugas swarm drone ini adalah saat ngekonfigurasi *weight* untuk setiap gaya dan aturan, aku sendiri ngelakuinnya dengan trial dan error, nyoba berkali-kali sampai swarm drone-nya lumayan cerdas untuk menuju target.