BAB 2
TEORI DASAR

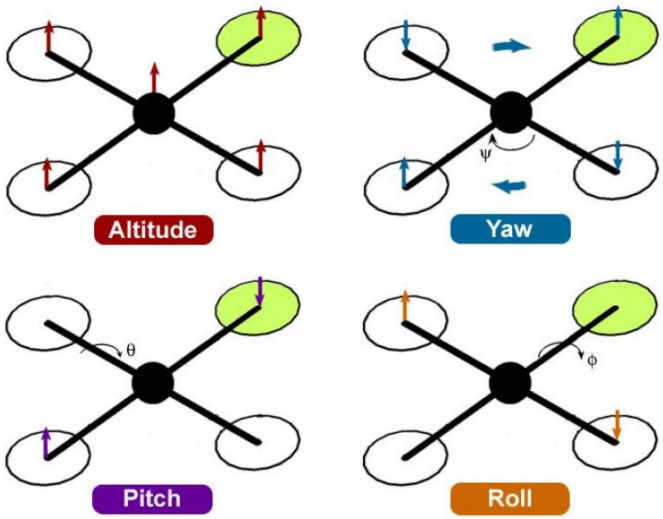
2.1. Quadrotor

Quadrotor merupakan pesawat tanpa awak yang dikendalikan menggunakan empat motor dengan empat baling-baling yang mana dua di antaranya berputar searah jarum jam dan dua lainnya berputar berlawanan arah jarum jam. Orientasi dan posisi *quadrotor* dapat ditentukan dengan menggunakan 2 sistem koordinat yaitu sistem koordinat absolut dan sistem koordinat relatif yang ditunjukkan melalui ilustrasi pada **Gambar 2.1**.

Gambar 2.1 Sistem Koordinat Absolut (hitam) dan Sistem Koordinat relatif (merah)

Perbedaan orientasi pada kedua sistem koordinat menciptakan 3 buah sudut rotasi (ϕ, θ, ψ) yang merupakan rotasi terhadap sumbu-x, sumbu-y dan sumbu-z secara berurutan. Pengukuran sudut tersebut diperoleh melalui bacaan sensor *accelerometer* dan *gyroscope* terhadap sistem koordinat relatif, sedangkan variabel posisi dari *quadrotor* dapat ditentukan oleh sistem koordinat absolut secara langsung menggunakan GPS.

Quadrotor memiliki 4 jenis gerakan utama yaitu gaya mengangkat (*thrust*), gerakan berputar (*roll*), gerakan menukik (*pitch*), dan gerakan membelok (*yaw*). Masing-masing gerakan bergantung pada besarnya kecepatan rotasi keempat rotor. Keempat jenis gerakan tersebut diilustrasikan pada **Gambar 2.2**.



Gambar 2.2 Gaya Mengangkat (*Altitude*), Gaya Membelok (*Yaw*), Gaya Menukik (*Pitch*), dan Gaya Berputar (*Roll*)

Pada tugas akhir ini, digunakan *quadrotor* jenis *AR Drone 2.0* dari perusahaan Parrot. *AR Drone 2.0* memiliki fitur penerbangan otomatis sehingga dapat melakukan lepas landas dan pendaratan dengan mudah. Unit ini dilengkapi dengan *wi-fi* sehingga *drone* dapat dihubungkan dan dikontrol dengan *smartphone* ataupun *tablet*. Selain itu, unit tersebut juga dilengkapi dengan kamera sehingga video hasil rekaman *drone* saat penerbangan berlangsung dapat dilihat secara langsung melalui *smartphone* atau *tablet* yang digunakan. Lama waktu terbang *Parrot AR. Drone 2.0* adalah sekitar 12 menit.

2.2. ROS (*Robot Operating System*)

Robot Operating System (ROS) dikembangkan pada tahun 2007 oleh *Stanford Artificial Intelligence Laboratory* (SAIL) dalam proyek *Stanford AI Robot* (STAIR) [5]. Pada tahun 2008, pengembangan ROS dilanjutkan oleh Willow Garage, suatu lembaga penelitian robotik, dengan dukungan lebih dari 20 lembaga penelitian [6].

Robot Operating System (ROS) adalah *framework* yang fleksibel untuk menulis perangkat lunak robotik. ROS merupakan kumpulan dari *tools*, *libraries*, dan konvensi-konvensi yang bertujuan untuk menyederhanakan pembuatan perilaku robot yang kompleks dan robas di berbagai macam *platform* robotik [7]. ROS

sangatlah kompleks dan memiliki banyak *tools* dan fungsi. Pada bagian ini akan dijelaskan komponen-komponen penting ROS yang berhubungan dengan tugas akhir ini.

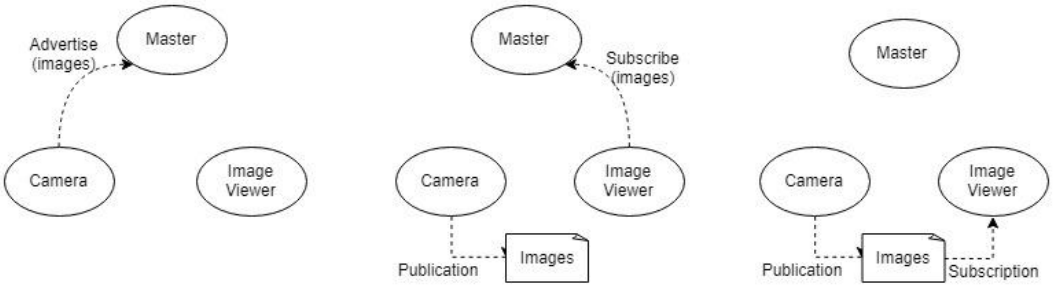
2.2.1. *Tools* dan konsep dasar

Tools dan komponen-komponen dasar pada ROS adalah sebagai berikut:

- ***Master***: ROS *Master* menyediakan servis penamaan dan registrasi untuk keseluruhan *nodes* di sistem ROS. *Master* ini akan melacak *publishers* dan *subscribers* dari topik serta servis di ROS. Peran dari *Master* adalah untuk memungkinkan *nodes* ROS mencari satu sama lain. Setelah *nodes* tersebut menemukan satu sama lain, mereka akan berkomunikasi dengan cara *peer-to-peer*.
- ***Node***: *Node* adalah proses pada ROS yang melakukan komputasi. Beberapa n*ode akan* bergabung menjadi suatu grafik dan berkomunikasi satu sama lain menggunakan topik *streaming*, servis RPC, dan *parameter server*.
  - ***Publisher node***: suatu *node* "pembicara" yang akan menyiarkan pesan secara terus menerus.
  - ***Subscriber node****:* suatu *node* "pendengar" yang akan menerima pesan secara terus menerus.
- ***Topic***: *Topic* adalah bus bernama tempat *node* bertukar pesan. *Topic* memiliki semantik *publish*/*subscribe* anonim, yang memisahkan pembuatan informasi dengan penggunaannya.
- ***Message***: *Node* berkomunikasi satu sama lain dengan mengirimkan *message*. *Message* adalah suatu struktur data yang terdiri dari *field* yang dikelompokkan.
- ***Bags***: *Bags* adalah format untuk menyimpan dan memainkan kembali data *message* ROS. *Bags* merupakan mekanisme yang penting untuk menyimpan data, seperti data sensor, yang sulit untuk disimpan tapi diperlukan untuk mengembangkan dan menguji algoritma.
- ***Package***: Perangkat lunak di ROS diorganisir dalam *packages*. Sebuah *package* dapat berisi *nodes* ROS, ROS-*independent library*, *dataset*, *file*

konfigurasi, perangkat lunak pihak ketiga, atau hal lainnya yang menyusun modul yang berguna.

Misalkan terdapat dua *node* yang akan digunakan; *node Camera* dan *node Image_viewer.* Mekanisme komunikasi ROS akan dimulai dengan *Node Camera* mendaftarkan diri ke ROS *master* bahwa ia ingin mem-*publish* citra ke topik *images* (**Gambar 2.3. (a)**). Setelah itu, *Node Camera* akan mengirimkan citra ke topik *images*, namun tidak ada yang men-*subscribe* topik tersebut sehingga citra hanya tersimpan di topik tersebut. Selanjutnya, *Node Image Viewer* yang ingin men-*subscribe* topik *images* akan mendaftarkan diri ke ROS *master* (**Gambar 2.3. (b)**). Dengan begitu, topik *images* memiliki *publisher* dan *subscriber*, sehingga *node Master* menginformasikan *Node Camera* dan *Image_viewer* mengenai keberadaan satu sama lain agar *node* tersebut dapat mengirimkan citra ke satu sama lain (**Gambar 2.3. (c)**).

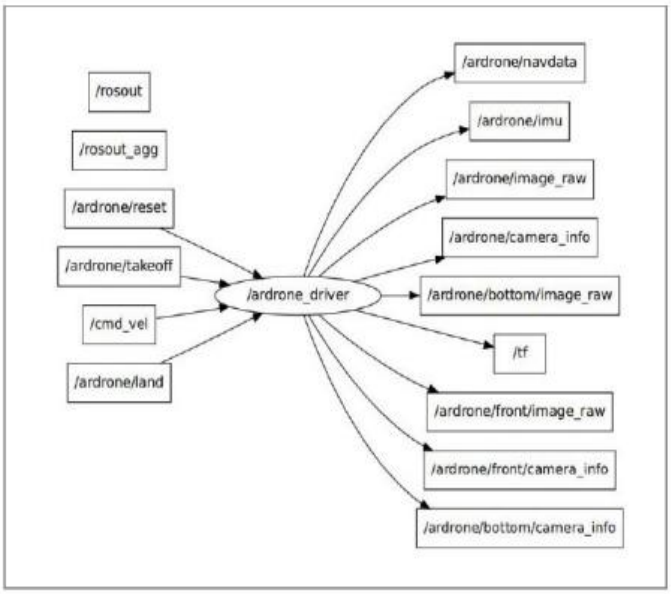


(a) (b) (c)

Gambar 2.3 Mekanisme Komunikasi *Node*: (a) *Node Camera* mendaftarkan diri ke ROS *Master*, (b) *Node Image Viewer* mendaftarkan diri ke ROS *Master*, (c) ROS *Master* menghubungkan *Node Camera* dengan *Image Viewer*

2.2.2. *Package Ardrone_autonomy*

Ardrone_autonomy adalah *package* ROS yang merupakan *driver* untuk *quadrotor Parrot AR-Drone 1.0 & 2.0*. *Driver* ini didasarkan pada SDK resmi *AR-Drone* versi 2.0.1 [8]. *Node ardrone_driver* dari *package* ini berinteraksi dengan topik yang berbeda-beda seperti yang diilustrasikan pada **Gambar 2.4.**



Gambar 2.4 Grafik Komputasional *Node ardrone_driver*

Topik-topik dengan panah yang menuju ke *ardrone_driver* (*/rosout*, */rosout_agg*, */ardrone/reset*, */ardrone/takeoff*, */cmd_vel*, dan */ardrone/land*) adalah topik-topik yang di-*subscribe* oleh *ardrone_driver*. Agar *drone* dapat melakukan tiga aksi pertama (*take off, land* atau *reset*), sebuah pesan kosong perlu dikirimkan ke topik yang berkaitan. Tipe dari pesan ini adalah *std_msgs::Empty*.

Setelah lepas landas dan dalam keadaan melayang, *drone* dapat dikendalikan dengan mengirimkan pesan (*publish*) ke topik *cmd_vel*. Tipe pesan untuk mengendalikan *drone* adalah *geometry_msgs::Twist*. Batasan nilai dari komponen *Twist* tersebut adalah di antara -1.0 dan 1.0.

Pesan-pesan perintah yang dapat dikirimkan ke topik *cmd_vel* antara lain:

- **+linear.x**: gerak maju
- **-linear.x**: gerak mundur
- **+linear.y**: gerak ke kiri
- **-linear.y**: gerak ke kanan
- **+linear.z**: naik
- **-linear.z**: turun
- **+angular.z**: berputar searah jarum jam

- **-angular.z**: berputar berlawanan jarum jam

Sebaliknya, topik-topik yang ditunjuk oleh panah dari *ardrone_driver* adalah topik-topik yang dikirimkan pesan (*publish*) oleh *ardrone_driver*. *Node* dapat mengirimkan pesan melalui topik-topik tersebut dan *node* lain yang men-*subscribe* topik tersebut akan menerima pesan tersebut.

Pada tugas akhir ini, topik */ardrone/image_raw* yang merupakan topik streaming kamera akan di-*subscribe*. Jenis pesan dari topik ini adalah *sensor_msgs/Image*.

2.2.3. *Package Tum_ardrone*

Tum_ardrone adalah *package* ROS yang berisi implementasi yang berkaitan dengan publikasi [9, 10, 11]. *Package* ini merupakan implementasi PTAM (*Parallel Tracking and Mapping*) yang dikembangkan oleh *Technische Universität München* [12].

Package ini terdiri dari tiga komponen: sistem SLAM (*Simultaneous Localization and Mapping*) monokuler, EKF (*Extended Kalman Filter*) dan pengontrol PID. *Tum_ardrone* memungkinkan *quadrotor* kualitas rendah yang dilengkapi dengan laptop untuk dinavigasi secara otonomos di lingkungan yang tidak diketahui dan tidak terjangkau oleh GPS.

Tum_ardrone disusun atas tiga *node*:

- ***drone_stateestimation***: untuk mengestimasi keadaan *drone*, meliputi PTAM dan visualisasi.
- ***drone_autopilot***: pengontrol *drone*, memerlukan *drone_stateestimation.*
- ***drone_gui****:* GUI (*Graphical User Interface*) untuk mengendalikan *drone* (dengan keyboard) dan untuk mengendalikan *node drone_autopilot* dan *drone_stateestimation*.

2.2.3.1. SLAM (*Simultaneous Localization and Mapping*)

SLAM adalah suatu teknik komputasi untuk membangun atau memperbaharui suatu peta dari lingkungan yang belum diketahui dan secara bersamaan memantau

pergerakan dan lokasi agen di peta tersebut. SLAM merupakan permasalahan yang umum dalam bidang navigasi robotik.

Dengan serangkaian perintah kontrol u_t dan pengamatan sensor o_t sepanjang waktu diskrit dengan *step* t, permasalahan SLAM adalah dalam mengkomputasikan suatu estimasi dari lokasi agen x_t dan peta lingkungan m_t. Semua variabel tersebut umumnya bersifat probabilistik, sehingga sasarannya adalah untuk mengkomputasikan $P(m_{t+1}, x_{t+1}|o_{t+1}u_{t+1})$.

Menggunakan aturan Bayes akan memberikan kerangka kerja untuk memperbaharui lokasi posterior secara sekuensial, dengan syarat fungsi peta dan transisi adalah $P(x_t|x_{t-1})$,

$$P(x_t|o_{1:t}, u_{1:t}, m_t)$$

$$= \sum_{m_{t-1}} P(o_t|x_t, m_t, u_{1:t}) \sum_{x_{t-1}} P(x_t, x_{t-1}) P(x_{t-1}|m_t, o_{1:t-1}, u_{1:t})/Z \qquad (2.1)$$

Dengan cara yang sama, peta dapat diperbaharui dengan

$$P(m_t|x_t, o_{1:t}, u_{1:t})$$

$$= \sum_{x_t} \sum_{m_t} P\left(m_t\middle|x_t, m_{t-1}, o_{t,} u_{1:t}\right) P(m_{t-1}, x_t|o_{1:t-1}, m_{t-1}, u_{1:t}) \qquad (2.2)$$

2.2.3.2. EKF (*Extended Kalman Filter*)

Extended Kalman Filter (EKF) merupakan salah satu algoritma optimal Bayesian untuk mengestimasi variabel keadaan (*state*). Berbeda dengan Kalman Filter yang digunakan pada sistem linear, EKF biasanya digunakan pada sistem nonlinear. Algoritma ini terdiri atas 2 tahap yaitu tahap prediksi dan tahap koreksi. Tahap prediksi digunakan untuk mengestimasi variabel keadaan berdasarkan sistem dinamik *quadrotor* sedangkan tahap koreksi digunakan untuk melakukan koreksi dari hasil estimasi berdasarkan hasil pengukuran IMU (*Inertial Measuring Unit*) *quadrotor*.

Pada *tum_ardrone*, EKF digunakan untuk menggabungkan seluruh data pengukuran yang ada untuk menghasilkan estimasi keadaan *quadrotor*. Selain itu,

EKF juga digunakan untuk mengkompensasi waktu tunda yang berbeda-beda pada sistem, yang disebabkan oleh komunikasi *wi-fi* dan komputasi pelacakan visual yang kompleks.

2.2.3.3. Pengontrol PID

Pengontrol PID (*Proportional-Integral-Derivative*) merupakan pengontrol mekanisme umpan balik yang sangat umum digunakan pada sistem kontrol. Sebuah pengontrol PID secara kontinu menghitung nilai kesalahan sebagai beda antara *setpoint* yang ditentukan dan variabel proses yang terukur. Nilai kesalahan ini akan dikomputasi oleh pengontrol PID untuk menghasilkan perintah kontrol baru dengan persamaan:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (2.3)$$

Dengan K_p, K_i, dan K_d secara berturut-turut adalah koefisien proporsional, integral, dan derivatif.

Berdasarkan data estimasi posisi dan kecepatan dari EKF pada $t + \Delta t_{control}$, dengan $\Delta t_{control}$ adalah waktu yang dibutuhkan agar suatu perintah kontrol baru diterima oleh *quadrotor*, akan digunakan kontrol PID untuk mengendalikan *quadrotor* menuju tujuan $\boldsymbol{p} = (\hat{x}, \hat{y}, \hat{z}, \hat{\Psi})^T \in \mathbb{R}^4$ pada sistem koordinat global. Berdasarkan estimasi keadaan, perintah kontrol yang dihasilkan ditransformasi ke sistem koordinat robot dan dikirimkan ke *quadrotor*. Untuk masing-masing derajat kebebasan *quadrotor*, digunakan pengontrol PID dengan *gain* yang berbeda seusai dengan yang dibutuhkan.

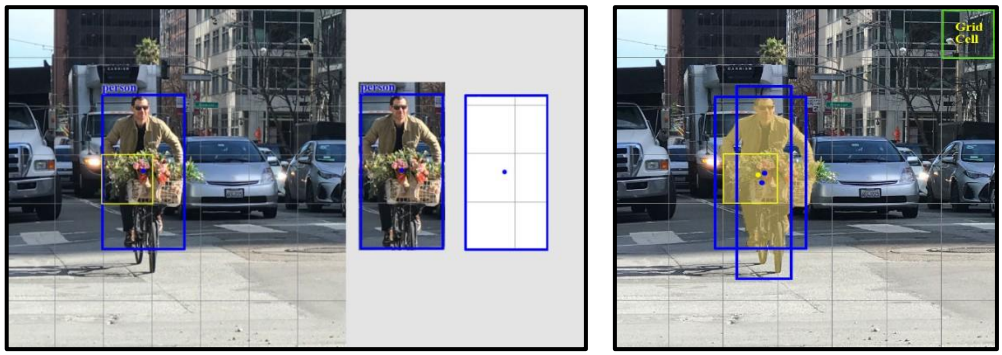
2.3. YOLO (*You Only Look Once*)

YOLO merupakan salah satu metode pendeteksian objek yang memanfaatkan sistem jaringan syaraf (*neural networks*) dalam mengintegrasikan ekstraksi kotak kandidat, ekstraksi fitur dan metode klasifikasi objek ke dalam jaringan saraf. YOLO secara langsung mengekstraksi kotak kandidat (*candidate boxes*) dari citra dan objek yang terdeteksi melalui seluruh fitur gambar. Dalam jaringan YOLO, citra dibagi menjadi kisi-kisi (*grid*) S × S. Kotak kandidat didistribusikan secara

merata pada sumbu X dan sumbu Y. Setiap kotak kandidat memiliki pendeteksian objek untuk memprediksi tingkat keyakinan (*confidence level*) adanya objek di setiap kotak kandidat. Tingkat keyakinan mencerminkan pengklasifikasian serta keakuratan posisi objek.

2.3.1. Pendeteksian Objek dengan YOLO

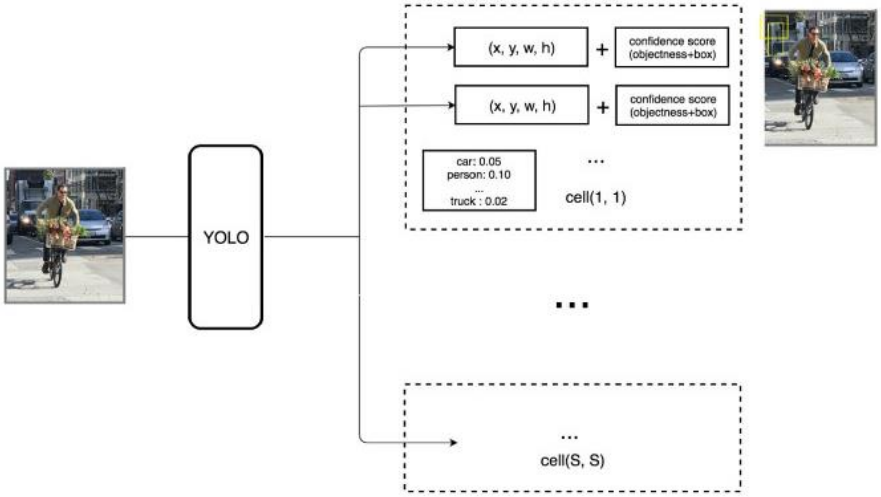
YOLO membagi gambar input menjadi kisi S × S. Setiap sel kotak memprediksi hanya satu objek. Misalnya, sel kotak kuning pada **Gambar 2.5 (a)** mencoba memprediksi objek "orang" yang pusatnya (titik biru) berada di dalam sel kotak. Setiap sel *grid* memprediksi sejumlah kotak batas. Pada **Gambar 2.5 (b)**, sel kotak kuning membuat dua prediksi kotak batas (kotak biru) untuk menemukan lokasi "orang" yang terkait.



(a) **(b)**

Gambar 2.5 Pendeteksian Objek dengan YOLO: (a) YOLO Mencoba Memprediksi Orang dengan Titik Biru sebagai Pusat, (b) Terbentuk dua prediksi Kotak Batas pada Lokasi Orang yang Dideteksi

Untuk setiap sel kisi akan diprediksi beberapa kotak pembatas B dan setiap kotak memiliki suatu nilai tingkat kepercayaan. Sistem akan mendeteksinya sebagai objek apabila *grid* tersebut memiliki lebih dari satu kotak pembatas. Probabilitas kelas bersyarat C (satu per kelas untuk kemungkinan kelas objek) akan ditampilkan dalam bentuk persentase.



Gambar 2.6 Algoritma Pembagian *Grid*

Setiap kotak pembatas berisi 5 elemen: (x, y, w, h) dan nilai kepercayaan kotak. Nilai kepercayaan mencerminkan seberapa besar kemungkinan kotak tersebut berisi objek (objektivitas) dan seberapa akurat kotak pembatas. Normalisasi dapat dilakukan untuk menormalkan lebar kotak pembatas w dan tinggi h dengan lebar dan tinggi gambar. x dan y adalah offset dari sel tersebut. Oleh karena itu, x, y, w dan h semuanya antara 0 dan 1. Setiap sel memiliki 20 probabilitas kelas bergantung pada *database* yang digunakan. Probabilitas kelas bersyarat adalah probabilitas bahwa objek yang terdeteksi milik kelas tertentu (satu probabilitas per kategori untuk setiap sel). Jadi, prediksi YOLO memiliki bentuk (S, S, B × 5 + C) = (7, 7, 2 × 5 + 20) = (7, 7, 30). Konsep utama YOLO adalah membangun jaringan CNN untuk memprediksi tensor (7, 7, 30). YOLO melakukan regresi linear menggunakan dua lapisan *fully connected* (FC) untuk membuat prediksi kotak batas 7 × 7 × 2 (**Gambar 2.6**). Untuk membuat prediksi akhir, kotak dengan nilai kepercayaan tertinggi (lebih besar dari ambang batas) akan disimpan sebagai prediksi akhir. Nilai kepercayaan kelas untuk setiap kotak prediksi dihitung sebagai:

$$\text{class confidence score} = \text{box confidence score} \times \text{conditional class probability} \tag{2.4}$$

Persamaan (2.5) sampai (2.6) digunakan untuk mengukur tingkat keyakinan (*confidence level*) pada klasifikasi dan lokalisasi tempat objek berada.

$$nilai\ keyakinan\ kotak \equiv P_r(object) \cdot IoU \tag{2.5}$$

$$probabilitas\ kelas\ kondisional \equiv P_r(class_i|object) \quad (2.6)$$

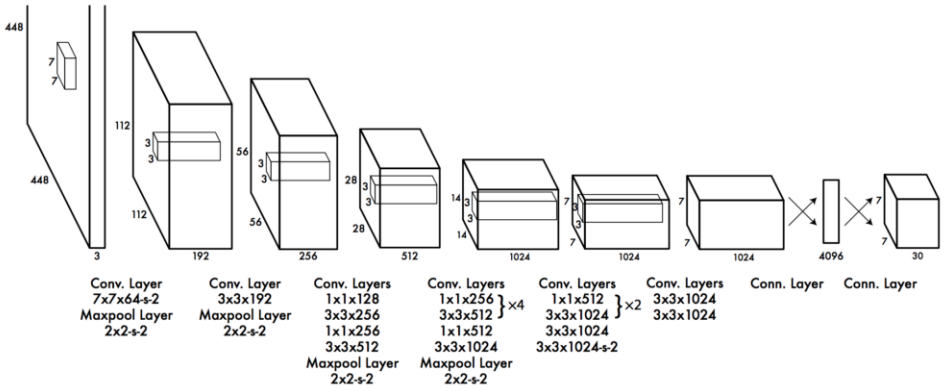
$$nilai\ keyakinan\ kelas \equiv P_r(class_i) \cdot IoU \quad (2.7)$$

2.3.2. Arsitektur YOLO

YOLO memiliki 24 lapisan konvolusional diikuti oleh 2 lapisan *fully connected* (FC). Beberapa lapisan konvolusi menggunakan lapisan reduksi 1 × 1 sebagai alternatif untuk mengurangi kedalaman peta fitur. Untuk lapisan konvolusi terakhir, keluarannya menghasilkan tensor dengan bentuk (7, 7, 1024) dan kemudian diratakan. Menggunakan 2 lapisan FC sebagai bentuk regresi linear, keluaran lapisan akan menghasilkan output dengan 7 × 7 × 30 parameter dan kemudian dibentuk kembali menjadi (7, 7, 30) untuk melakukan klasifikasi berdasarkan kelasnya. YOLO menggunakan lapisan konvolusional 1 * 1 (untuk mengintegrasikan informasi lintas saluran) dan lapisan konvolusional 3 * 3 untuk melakukan penggantian sederhana.

Algoritma YOLO adalah sebagai berikut:

1. Citra masukan dibagi menjadi kisi S * S;
2. Untuk setiap kisi, diperkirakan akan ada kotak pembatas (*boundary boxes*) B yang mencakup pembatas yang mewakili kepercayaan target dan probabilitas bahwa setiap wilayah perbatasan memiliki beberapa kelas;
3. Jendela target S * S * B dapat diprediksi dan kemudian jendela target dapat dihapus jika mereka memiliki probabilitas lebih rendah daripada ambang batas (pada umumnya 50%);
4. Jendela redundan hasil prediksi dapat dihapus dengan *Non-Maximum Suppression* (NMS). NMS digunakan untuk memastikan bahwa dalam deteksi objek, objek tertentu diidentifikasi hanya sekali.



Gambar 2.7 Lapisan Konvolusional YOLO

Cara kerja NMS adalah sebagai berikut:

1. Semua sel dengan probabilitas kehadiran objek (dihitung dalam lapisan *softmax* akhir) yang kurang dari ambang batas yang ditentukan akan dibuang.
2. Sel dengan probabilitas terbesar dijadikan kandidat di antara kandidat lain untuk dinyatakan sebagai objek prediksi yang tepat;
3. Sel yang memiliki nilai *Intersection over Union* (IoU) ≥ 0,5 di dalam sel prediksi dibuang, menyisakan jendela hasil prediksi yang tunggal untuk tiap objeknya.

2.3.3. Loss Function

YOLO memprediksi beberapa kotak pembatas dalam satu sel *grid*. Agar dapat menghitung besar rugi-rugi (*loss*) untuk probabilitas, setiap *grid* dipastikan hanya memiliki satu objek sebagai prediksinya. Untuk itu, dalam pengolahan dipilih prediksi dengan nilai IoU terbesar dengan nilai kebenaran objek dasar (*ground truth*) yang bersumber dari *pretrained database*. Strategi ini mengarahkan prediksi pada objek yang spesifik pada setiap kotak pembatas.

YOLO menggunakan *sum-squared error* (SSE) antara prediksi setiap kotak pembatas dan kebenaran objek dasar untuk menghitung rugi-rugi. *Loss function* terdiri dari:

a. *Classification loss.*

b. *Localization loss* (galat yang terbentuk antara kotak pembatas hasil prediksi dengan kebenaran objek dasar).
c. *Confidence loss* (tingkat keyakinan dari kotak pembatas).

2.3.3.1. *Classification Loss*

Jika sebuah objek terdeteksi, galat dari setiap sel yang diklasifikasi dibandingkan dengan probabilitas kondisional setiap kelas dengan melihat nilai SSE untuk setiap kelasnya.

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \quad (2.8)$$

Dengan:

$\mathbb{1}_i^{obj} = 1$ jika objek hadir di sel i, selain dari itu bernilai 0
$\hat{p}_i(c)$ menunjukkan probabilitas kelas kondisional untuk kelas c di sel i

2.3.3.2. *Localization Loss*

Localization loss mengukur nilai galat dari kotak pembatas hasil prediksi dan ukurannya. *Loss* ini hanya diterapkan pada kotak yang memiliki objek yang sesusai dengan kelas, sehingga kotak pembatas yang tidak berisikan objek yang bersesuaian dengan kelas tidak memiliki nilai tersebut.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2.9)$$

Dengan:

$\mathbb{1}_{ij}^{obj}$=1 jika kotak pembatas ke-j di sel i mendeteksi objek, selain itu dari bernilai 0.
λ_{coord} meningkatkan bobot dari *loss* pada koordiat kotak pembatas.

2.3.3.3. *Confidence Loss*

Jika sebuah objek terdeteksi, tingkat keyakinan dari kotak pembatas dapat ditentukan sebagai:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 \tag{2.10}$$

Dengan:

$\hat{C}_i$ adalah nilai keyakinan kotak *j* di sel *i*.

$\mathbb{1}_{ij}^{obj}$=1 jika kotak pembatas ke-*j* di sel *i* mendeteksi objek, selain dari itu bernilai 0.

Jika sebuah objek tidak terdeteksi, tingkat keyakinan dari kotak pembatas dapat ditentukan sebagai:

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 \tag{2.11}$$

Dengan:

$\mathbb{1}_{ij}^{noobj}$ adalah komplemen dari $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ adalah nilai keyakinan kotak *j* di sel *i*.

λ_{noobj} memberi bobot *loss* ketika mendeteksi latar belakang.

2.3.3.4. *Total Loss*

Total Loss merupakan penjumlahan dari keseluruhan *loss*, yaitu *localization*, *confidence* dan *classification loss*.

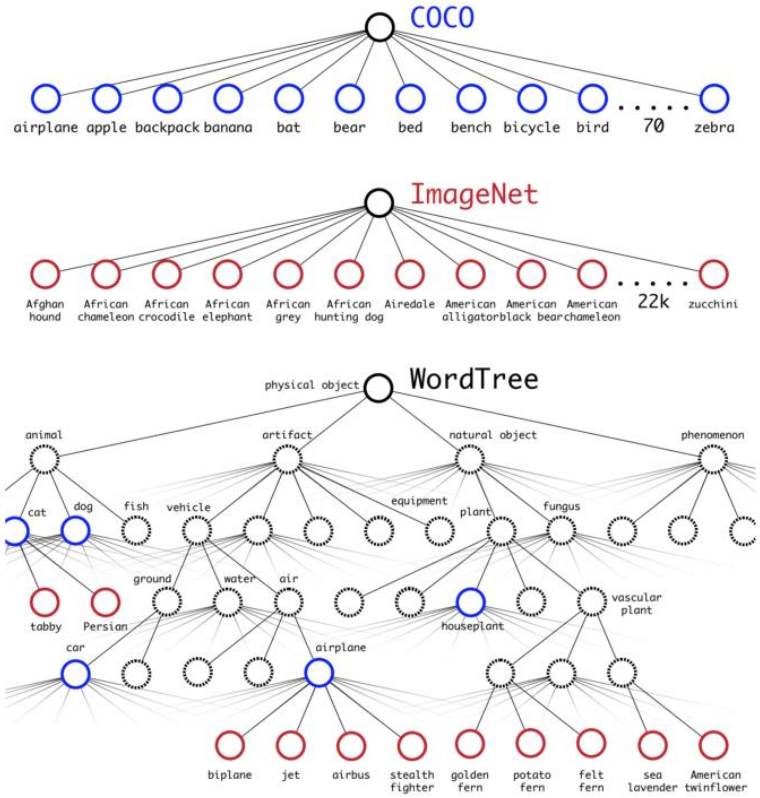
$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

(2.12)

2.3.4. *COCO Database*

COCO *dataset* merupakan kumpulan objek yang telah dikategorikan dalam bentuk kelas yang dikembangkan oleh Tsung-Yi Lin dkk. [14] dengan mengklasifikasikan foto dari 91 jenis objek yang mudah dikenali oleh anak berusia 4 tahun. COCO Dataset menyajikan analisis prediksi objek yang terperinci dari dataset dibandingkan dengan PASCAL, ImageNet, dan SUN. Penggunaan database yang mudah dan menggunakan keterkaitan konteks sebagai acuannya membuat COCO dataset tepat untuk digunakan pada tugas akhir ini untuk pengklasifikasian yang memerlukan kecepatan tinggi dengan tingkat akurasi yang memadai.



Gambar 2.8 *Database* yang Tersedia

2.4. Fotogrametri

Fotogrametri adalah ilmu, teknologi, dan rekayasa yang bersumber dari cara pengolahan data hasil rekaman dan informasi, baik dari citra fotografik maupun dari non-fotografik, untuk tujuan pemetaan rupa bumi serta pembentukan basis data bagi keperluan rekayasa tertentu. Selain untuk pemetaan rupa bumi (pemetaan

topografi), fotogrametri dapat dimanfaatkan untuk memperoleh berbagai informasi lahan atau biasa disebut sebagai GIS (*Geographic Information System*). Secara umum, fotogrametri dapat dipisahkan atas dua kelompok besar, yaitu:

1. Fotogrametri metrik adalah penentuan geometri dan posisi obyek melalui pengukuran/pengamatan baik dari jarak, sudut, luas, dan volume hasil proses fotogrametri.
2. Fotogrametri *interpretative* adalah pengolahan citra fotografik maupun non-fotografik (*radar*, *satellite imagery*, dan lain-lain) guna pembentukan basis data bagi keperluan rekayasa tertentu.

Kegiatan pemetaan terdiri atas tiga langkah pokok, yaitu:

1. *Data capture,* pengumpulan data/ pemotretan/ perekaman data.
2. *Data processing* atau pengolahan *data capture* menjadi hasil peta.
3. Penyajian dan penyimpanan data (termasuk pemeliharaan basis data).

2.4.1. Interpretasi Citra

Interpretasi citra merupakan kajian foto udara atau citra dengan tujuan mengidentifikasi obyek dan melihat arti pentingnya obyek tersebut (Estes dan Simonett, 1975 dalam Sutanto 1986). Interpretasi dapat dilakukan secara digital berdasarkan nilai pantulan spektral obyek yang terekam pada citra digital. Data dalam bidang fotogrametri dapat dibedakan menjadi:

1. Data metrik (kuantitatif) adalah data yang bersifat kuantitatif dan ditunjukkan dengan nilai angka. Sebagai contoh data-data ukuran yang diambil dari pengukuran di atas bidang foto seperti ukuran jarak, sudut, ketinggian, bentuk dan ukuran suatu obyek. Data-data ini dapat dijadikan bentuk peta dengan skala tertentu melalui proses reduksi dan tranformasi.
2. Data non metric (kualitatif) adalah data yang bersifat kualitatif dan menunjukkan mutu atau perbandingan dari unsur-unsur obyek yang ada diatas bidang foto. Untuk tujuan tertentu data ini sangat menunjang dalam pembuatan peta dalam tema tertentu.

2.4.2. Geometri Dalam Fotogrametri Udara

Derivasi skala foto (sp) dengan Persamaan (2.12) adalah rasio yang bertujuan untuk menentukan skala negatif berdasarkan lebar negatif terkait jarak tanah yang dicakup oleh bingkai paparan.

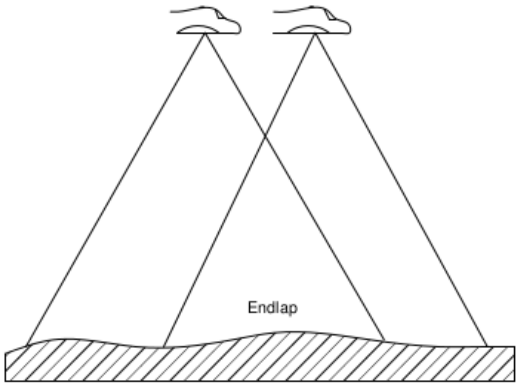
$$s_p = \frac{n}{g} : \frac{1\ in}{1\ ft} \tag{2.13}$$

Rumusan yang telah disederhanakan untuk menentukan skala foto diturunkan dari persamaan (2.12), yaitu:

$$s_p = H/f \tag{2.14}$$

2.4.2.1. *Endlap*

Endlap, juga dikenal sebagai *forward overlap*, adalah area citra yang beririsan secara berurutan di antara foto-foto di sepanjang jalur penerbangan. Bagian yang tumpang tindih dari dua citra udara yang berurutan akan menciptakan efek tiga dimensi yang diperlukan untuk pemetaan, yang disebut sebagai *stereomodel* atau lebih umum sebagai "model". **Gambar 2.9** menunjukkan area *endlap* pada satu pasangan foto berturut-turut dalam satu baris penerbangan.



Gambar 2.9 *Endlap* Pada Dua Foto Berurutan

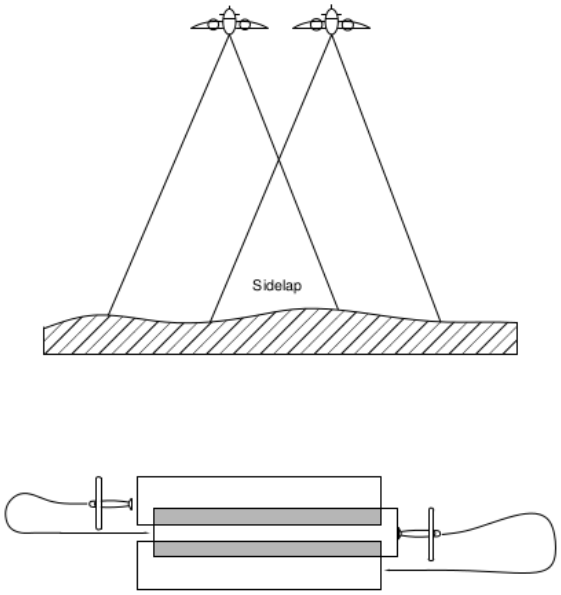
Pada umumnya, pesawat akan mengikuti jalur penerbangan yang telah ditentukan (*path planning*) saat kamera mengekspos berturut-turut gambar yang tumpang tindih. *Endlap* standar berkisar antara 55 dan 65% dari panjang foto, dengan nominal rata-rata 60% untuk sebagian besar proyek pemetaan.

Gain dari *endlap*, yang merupakan jarak antara pusat foto berturut-turut di sepanjang jalur penerbangan, dapat dihitung dengan persamaan (2.14).

$$g_{end} = s_p * w * [(100 - o_{end}/100)] \quad (2.15)$$

2.4.2.2. *Sidelap*

Sidelap, juga disebut *side overlap*, mencakup area yang tumpang tindih di antara foto-foto pada dua jalur penerbangan yang berdekatan. Sidelap diperlukan agar tidak ada celah kosong dalam cakupan tiga dimensi pada pemetaan *multiline* (lebih dari satu jalur). **Gambar 2.10** (atas) menunjukkan posisi relatif *head-on* pesawat dalam jalur penerbangan yang berdekatan dan area yang dihasilkan dari cakupan paparan. Biasanya, *sidelap* berkisar antara 20 dan 40% dari lebar foto, dengan nominal rata-rata 30%. **Gambar 2.10** (bawah) menggambarkan pola *sidelap* dalam pemetaan yang membutuhkan tiga jalur penerbangan.



Gambar 2.10 *Sidelap* pada Dua Foto yang Bersampingan (atas) dan *Sidelap* pada Jalur Penerbangan yang Bersebelahan (bawah)

Gain sidelap, jarak antara pusat-pusat jalur penerbangan yang berdekatan, bisa dihitung dengan menggunakan persamaan (2.15).

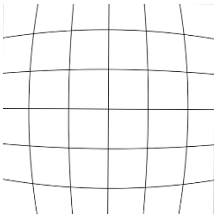
$$g_{side} = s_p * w * [(100 - o_{side})/100] \quad (2.16)$$

2.4.2.3. Distorsi dan Perpindahan

Distorsi adalah fenomena fisik yang dalam situasi tertentu dapat memengaruhi geometri gambar tanpa mengganggu kualitas atau mengurangi informasi yang ada dalam gambar. Fenomena ini dapat dimodelkan dengan distorsi radial, komponen yang paling menonjol dalam distorsi. Distorsi radial disebabkan oleh bentuk bola lensa, mulai dari kecembungan dan kualitas permukaan lensa, sedangkan distorsi tangensial disebabkan oleh *decentering* dan non-ortogonalitas komponen lensa sehubungan dengan sumbu optik.

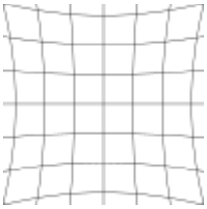
Distorsi radial simetris yang sering terjadi dapat digolongkan menjadi tiga jenis, yaitu:

1. Distorsi *Barrel*: perbesaran gambar berkurang seiring bertambahnya jarak dari sumbu optik. Efek yang tampak adalah gambar yang nampak seperti telah dipetakan pada permukaan bola. Lensa *fisheye*, yang mengambil tampilan setengah bola, memanfaatkan jenis distorsi ini sebagai cara untuk memetakan bidang objek yang sangat lebar ke area citra yang terbatas.



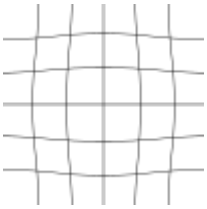
Gambar 2.11 Distorsi *Barrel*

2. Distorsi *Pincushion*: perbesaran gambar meningkat seiring dengan bertambahnya jarak dari sumbu optik. Efek yang jelas terlihat adalah garis yang tidak melalui sumbu optik akan terlihat ditarik ke tengah, seperti *pincushion* (bantal peniti).



Gambar 2.12 Distorsi *Pincushion*

3. Distorsi *Mustache*: gabungan dari kedua jenis distorsi, disebut sebagai distorsi kumis atau distorsi kompleks.



Gambar 2.13 Distorsi *Mustache*

2.4.3. *Agisoft Photoscan*

Agisoft merupakan salah satu perangkat lunak yang paling sering digunakan dalam meproses citra udara. Langkah-langkah dasar dalam penggunaan *Photoscan* adalah sebagai berikut:

1. *Align photo*: Poin utama diekstraksi dari citra melalui analisis tekstur gambar. Kemudian, posisi relatif dan orientasi gambar diestimasi menggunakan algoritma pengaturan bundel berdasarkan titik pencocokan yang diidentifikasi.
2. *Build dense point clouds*: Langkah utama proses. Berdasarkan perkiraan posisi relatif dan orientasi citra, informasi kedalaman untuk setiap kamera dihitung dan kemudian digabungkan menjadi satu titik awan padat. Parameter penting dalam langkah ini adalah derajat penyaringan. Tiga opsi disediakan untuk menentukan derajat penyaringan, yaitu *mild*, *agressive* dan *moderate*. Pengaturan *aggresive* tepat untuk area yang tidak berisi detail kecil yang bermakna. Sebaliknya, pengaturan *mild* cocok untuk area kompleks dengan banyak detail kecil. Pengaturan *moderate* memberikan hasil yang berada di antara *mild* dan *aggresive*.
3. *Add GCPs*: Awan titik yang dihasilkan dijelaskan dalam sistem koordinat relatif lokal. GCP diperlukan untuk mengubah awan titik dari koordinat relatif menjadi koordinat yang di-georeferensi.