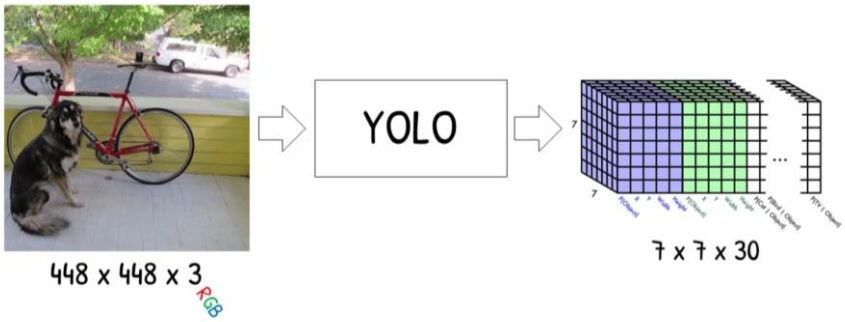
Kinematics

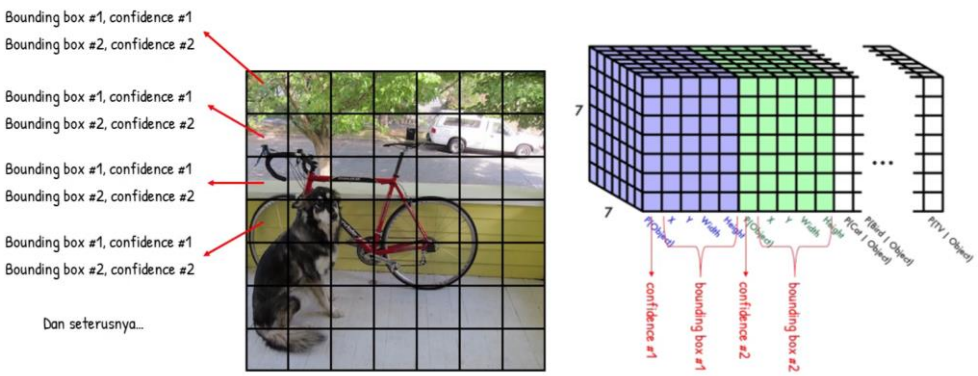
- Object Detection

Versi YOLO

Pertama kita inputkan gambar lalu gambar itu akan di rezise menjadi 448x448. Cara kerja object detectionnya sebagai berikut.



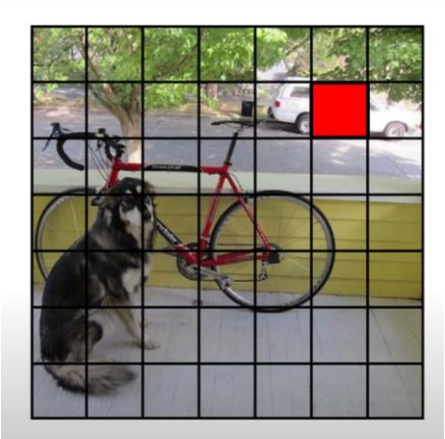
YOLO menginput gambar 448x448x3 karena berwarna. Lalu neural network. Lalu outputnya akan berukuran 7x7x30. 7x7 berarti YOLO akan membagi gambar menjadi grid 7x7 dan x30 karena kategorinya ada 20.



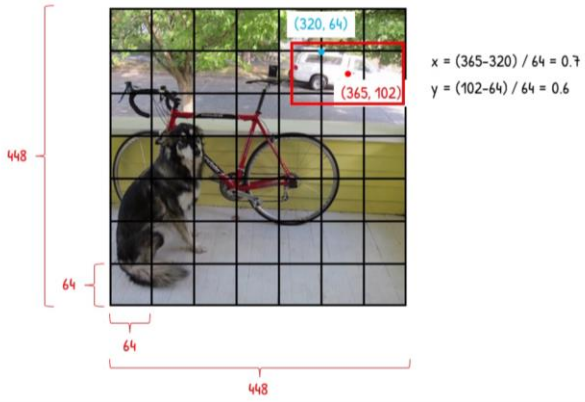
Total prediksi bounding box = 7 x 7 x 2 = 98

Bounding Box yang tebal memiliki nilai confident yang tinggi

Setiap cell pada grid memiliki prediksi bounding boxnya masing masing. Misal kita perhatikan satu cell.

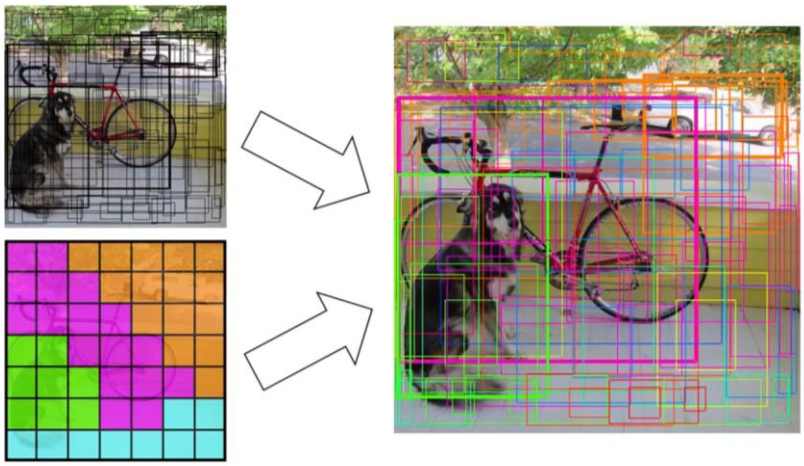


Di cell ini, dia memprediksi satu bounding box yang memiliki nilai confident yang tinggi. Bounding box ini memiliki titik tengah. Cara kita menghitung seberapa Panjang bounding box itu dan di koordinat mana titik tengah bounding box itu adalah dengan

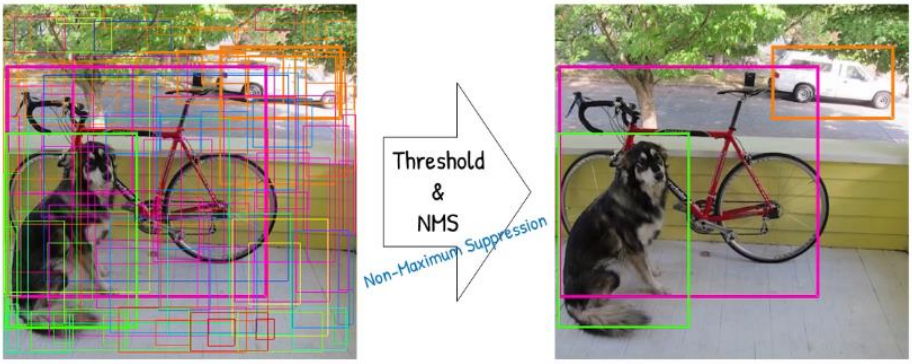


Selain bounding box, YOLO juga mengkategorikan objek di setiap selnya. Misal di setiap selnya, YOLO memprediksikannya seperti ini

Kita sudah punya prediksi bounding box dan kita juga sudah punya kategorinya. Jika digabungkan maka setiap bounding box akan memiliki kategorinya sendiri.



Lalu prediksi yang banyak di treshold, artinya bounding box yang memiliki confident yang rendah akan tidak di anggap. Lalu NMS, jadi hanya bounding box yang memiliki confident tertinggi yang diperhitungkan



- Pose Estimation

Pose Estimation adalah algoritma untuk memprediksi dan melacak lokasi seseorang atau objek

Ada dua cara untuk mencapai hal ini :

Top-Down Approach :

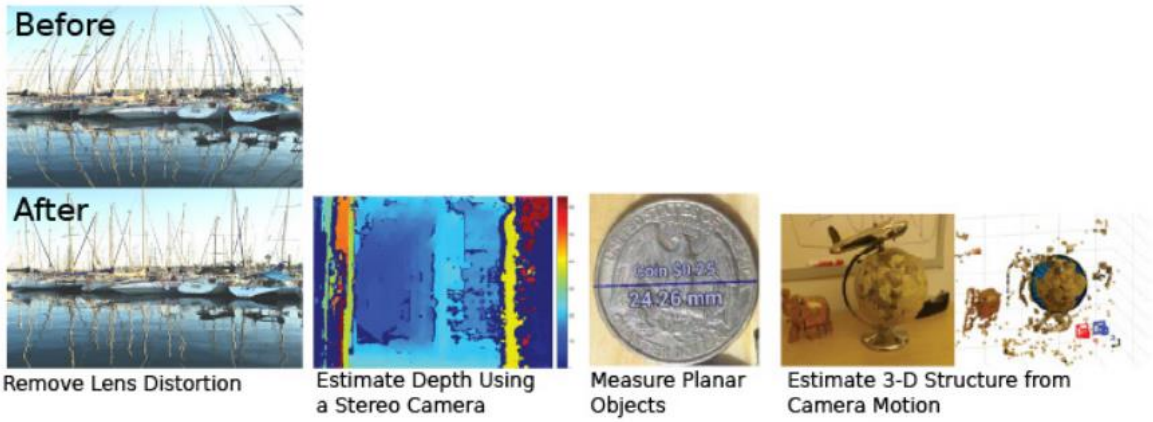
Pertama, sebuah gambar atau video akan diidentifikasi apakah ada manusia. Lalu gambar itu di crop untuk membuat single person image. Lalu gunakan Single Person Pose Estimation (SPPE) di setiap gambar yang sudah dipotong untuk mencari estimated pose. Lalu gabungkan semua estimated pose ke gambar yang asli

Bottom-Up Approach :

Kalau ini, computer langsung mendeteksi keypoints setiap tubuh manusia yang ada di dalam gambar

- Camera Callibration

Camera Callibration adalah algoritma untuk mengoreksi distorsi lensa, mengukur ukuran objek dalam dunia asli, atau menentukan lokasi kamera.



Pertama, kita harus mengambil sejumlah gambar. Kedua, algoritma akan mendeteksi titik-titik pada gambar dari sudut posisi yang berbeda. Titik ini digunakan sebagai referensi dalam proses camera callibration. Ketiga, algoritma akan memetakan titik-titik 3D dari pola yang sudah ditentukan menjadi titik-titik 2D. Hal ini menggunakan **proyeksi pinhole**. Keempat, algoritma akan mencari corner dari benda tersebut. Keempat, algoritma akan mencari lagi corner dari benda tersebut. Algoritma akan mencari lagi karena calibration yang bagus adalah presisi. Untuk menghasilkan hasil yang bagus kita harus mencari titik cornernya sampai sub-pixel level. Terakhir algoritma akan mengkalibrasi kamera.

ADRC (Active Disturbance Rejection Control)

ADRC adalah algoritma yang paling optimis. Algoritma ADRC adalah sebuah modern-based untuk memecahkan suatu masalah. Masalah apa yang ditangani? Masalah seperti gangguan eksternal dan ketidakpastian dalam model sistem. ADRC sangat berguna pada saat model matematika dari sistem tidak diketahui atau terlalu rumit. Contoh kita ingin membuat robot yang bisa terbang. Pasti banyak gangguan eksternal yang dapat terjadi, seperti kuatnya arus angin, berat beban robotnya, dan panasnya matahari. ADRC algorithm ini dapat memperkirakan gangguan eksternal tersebut dan menyesuaikan sinyal robot supaya robot tetap stabil di segala medan.

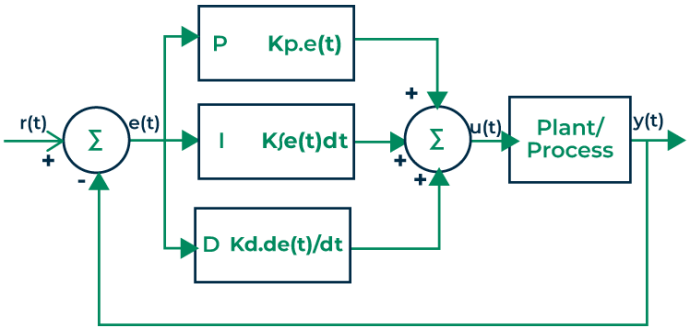
PID (Proportional-Integral-Derivative) Control Algorithms

PID merupakan gabungan dari beberapa aksi yang dapat dilakukan untuk mengatur operasi sebuah sistem dengan mengatur variabel-variabel yang ada.

Proportional merupakan variabel penyelesaian sebuah *error* saat error itu muncul. Integral merupakan variabel penyelesaian sebuah *error* dengan mencegah *error* yang telah terjadi. Derivative merupakan variabel penyelesaian sebuah *error* dengan memprediksi *error* yang akan terjadi.

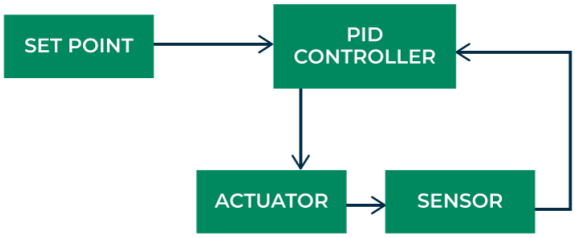
Secara matematis, PID dapat direpresentasikan dengan bentuk berikut.

$$Co(t) = Kp \cdot e(t) + Ki \cdot \int e(t)\,dt + Kd \cdot de(t)/dt$$



Sumber: https://www.geeksforgeeks.org/proportional-integral-derivative-controller-in-control-system/

PID berfungsi berdasarkan *closed system loop* yang dapat digambarkan dengan *flowchart* berikut. *Looping* ini akan berlangsung hingga hasil yang diinginkan tercapai.



Sumber: https://www.geeksforgeeks.org/proportional-integral-derivative-controller-in-control-system

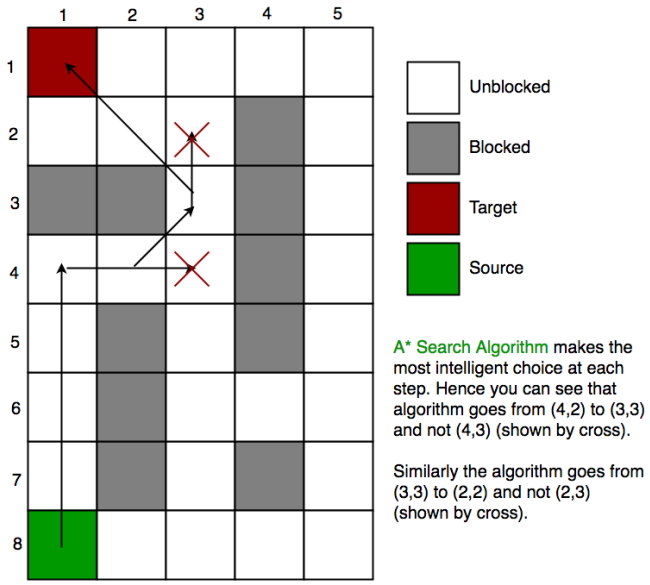
A* (A STAR ALGORITHM)

Algoritma A star merupakan algoritma untuk mengaproksimasi shortest path dari poin awal ke poin target. Algoritma A star merupakan algoritma yang umum digunakan dalam aplikasi maps untuk mencari jalan tercepat, contohnya Google Maps.

Algoritma A star menggunakan tiga variabel untuk menentukan shortest path yang dirumuskan sebagai berikut.

f = h + g

dengan g menunjukkan jarak pergerakan dari poin A ke poin B dan h menunjukkan jarak pergerakan dari poin awal ke poin target. Kita dapat memvisualisasikan pergerakan dengan diagram kotak-kotak sebagai berikut.



sumber: https://www.geeksforgeeks.org/a-search-algorithm/

Dari grafik, kita dapat menamakan tiap kotak dengan koordinat cell (x,y). Misal, kotak hijau sebagai cell (8,1) dan kotak merah sebagai cell (1,1)

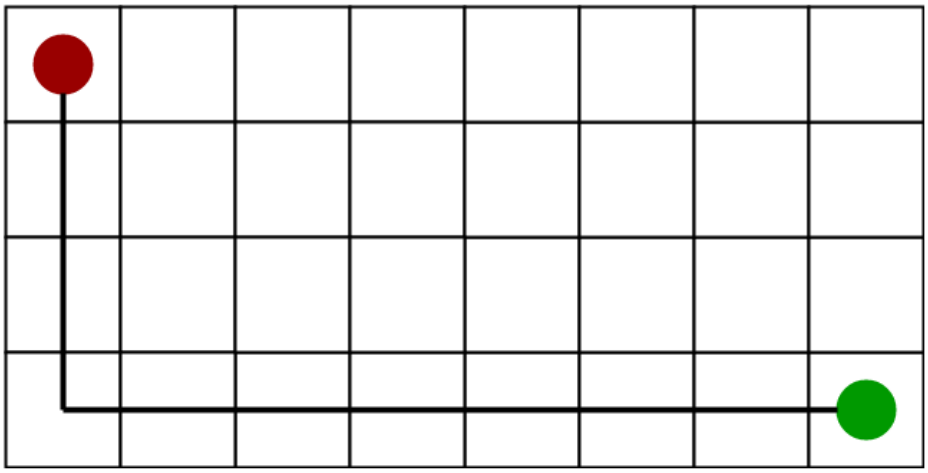
Kita sebenarnya tidak dapat menentukan variabel h tanpa benar-benar mencoba setiap path, tetapi itu akan memakan sangat banyak waktu. Jadi, kita hanya bisa menebak-nebak variabel h dengan menggunakan heuristik. Terdapat tiga heuristik yang dapat digunakan untuk menentukan, yaitu Manhattan Distance, Diagonal Distance, dan Euclidean Distance.

1. Manhattan Distance

Manhattan Distance menunjukkan selisih cell (x,y) dengan cell (x,y),

h = abs (current.x - target.x) + abs (current.y - goal.y)

perlu diperhatikan bahwa metode heuristik ini hanya dapat digunkana ketika kita hanya dapat bergerak secara vertikal dan horizontal.



2. Diagonal Distance

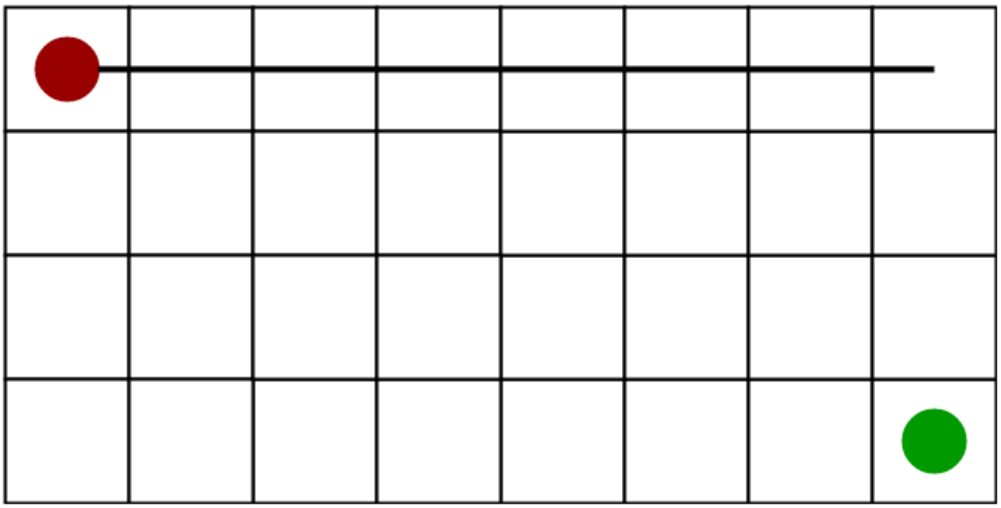
Diagonal Distance menunjukkan hal yang sama dengan Manhattan Distance, tetapi juga memperbolehkan pergerakan diagonal,

dx = abs (current.x - target.x)

dy = abs (current.y - goal.y)

h = D * (dx + dy) + (D2 - 2*D) * min(dx, dy)

D merupakan jarak pergerakan (umumnya = 1) dan D2 merupakan jarak pergerakan diagonal (umumnya sqrt(2)).



3. Euclidean Distance

Eucledian Distance menghitung seluruh pergerakan sebagai pergerakan diagonal,

h = sqrt((current.x - target.x)^2 + (current.y - goal.y)^2).