Kinematics

Kinematics algoritma merupakan kerangka matematika yang digunakan untuk menentukan gerakan, seperti sendi robot tanpa berdasarkan input seperti gaya, posisi, atau kecepatan. Kinematics terbagi menjadi dua yaitu forward kinematics algorithm dan inverse kinematics algorithm. Forward Kinematics Algorithm melakukan kalkulasi terhadap sudut sendi bagian ujung robot. Sedangkan, Inverse Kinematics Algorithm melakukan kalkulasi terhadap sudut sendi yang diperlukan agar bagian sendi paling ujung robot terletak pada posisi yang benar.

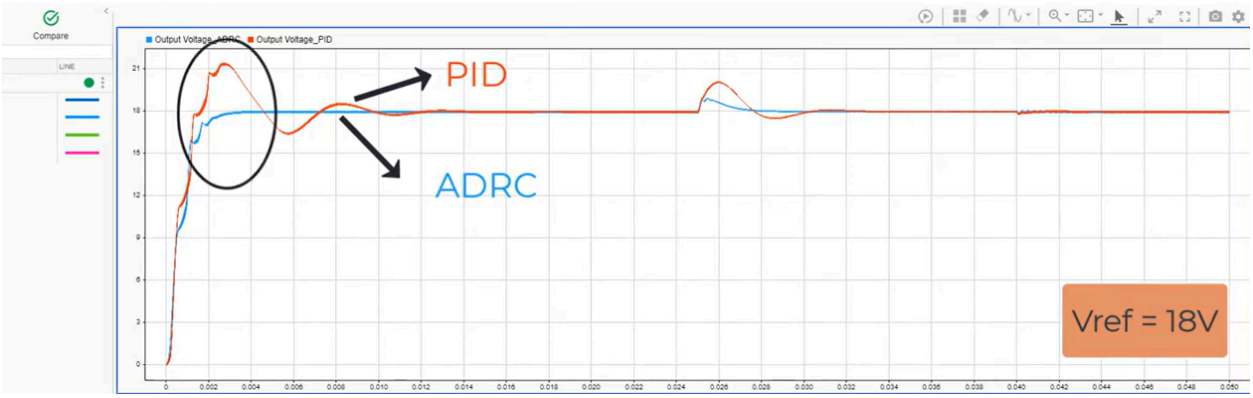
Object Detection merupakan identifikasi dan menentukan lokasi sebuah objek di dalam sebuah gambar atau video. Object Detection dan Kinematics tidak berhubungan langsung, namun masih berhubungan. Robot yang memiliki tujuan mengambil sebuah objek perlu melakukan kalkulasi terhadap gambar atau video yang diberikan agar robot dapat menentukan posisi benda tersebut dalam lingkungan tiga dimensi. Setelah melakukan kalkulasi terhadap posisi benda, kinematics baru akan memainkan peran dalam menentukan gerakan sendi robot agar robot dapat memegang dan mengangkat objek yang telah diidentifikasi

Pose Estimation menentukan posisi dan sikap sebuah benda di dalam lingkungan tiga dimensi. Kinematics memainkan peran penting dalam pose estimation. Saat ingin menentukan sikap robot, terutama sikap tubuh yang seperti manusia, robot memerlukan kinematics untuk menentukan derajat tekuk sendi seluruh robot. Untuk sikap tubuh seperti manusia, biasanya dinamakan kinematics chain karena tubuh yang kaku dihubungkan dengan banyak sendi.

Camera Calibration merupakan proses dimana data 2D, tanpa bagian ini Object Detection dan Pose Estimation tidak dapat dilakukan. Camera Calibration perlu ditentukan parameter intrinsik dan ekstrinsik kamera. Parameter intrinsik terdiri panjang fokus kamera, pusat optik, dan koefisien distorsi yang membantu mengubah lingkungan 3D menjadi lingkungan 2D. Sedangkan, parameter ekstrinsik terdiri dari posisi dan orientasi kamera relatif terhadap koordinat robot, dimana parameter dapat mengubah koordinat lingkungan 2D menjadi koordinat 3D. Kinematics berhubungan dengan Camera Calibration, sebab Camera Calibration setelah melakukan perhitungan akan memberi matriks transformasi yang dapat digunakan oleh kinematics untuk kalkulasi gerakan robot terhadap benda/objek yang teridentifikasikan.

ADRC (Active Disturbance Rejection Center)

ADRC (Active Disturbance Rejection Center) merupakan tipe dari metode kontrol kuat nonlinier yang menambahkan sistem model dengan variabel tambahan dan fiktif, mewakili segala sesuatu yang tidak dimasukkan oleh pengguna dalam model matematika robot. ADRC bekerja baik dalam kondisi dimana sistemt dengan dinamika yang tidak diketahui serta gangguan internal dan eksternal. ADRC adalah penerus dari teknologi PID, ADRC memiliki performa yang lebih baik dari pada PIP. Sebab, ADRC dapat mengatasi gangguan baik internal maupun eksternal sehingga tegangan keluar yang diharapkan lebih sesuai meskipun terdapat perubahan tiba-tiba tegangan masuk.



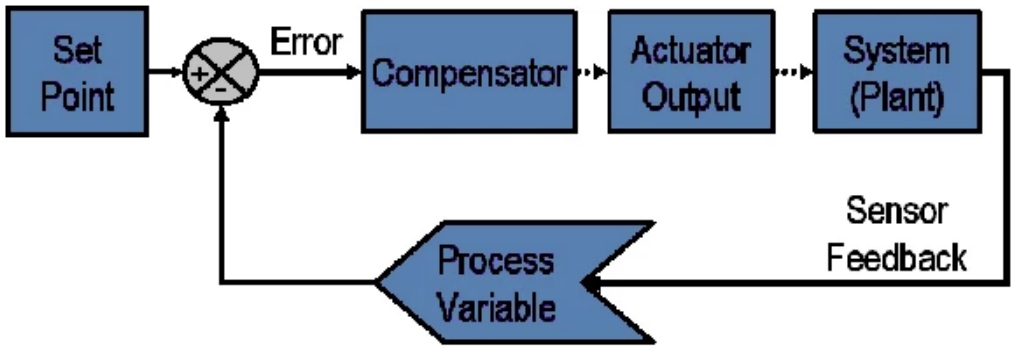
Sumber : https://www.youtube.com/watch?v=snH8-fpuNJA

ADRC terbagi menjadi berbaga bagian, yaitu :

1. Tracking Differentiator(TD); TD digunakan untuk mengubah masukan referensi yang tiba-tiba atau berisik menjadi lintasan yang mulus.
2. Extended State Observer(ESO); Tujuan dari ESO adalah mengamati gangguan dan kemudian menghasilkan koreksi berdasarkan hasil pengamatan.
3. Nonlinear State Error Feedback(NLSEF); NLSEF menstabilkan sistem menggunakan umpan balik berdasarkan kesalahan antara keadaan yang diinginkan dan perkiraan keadaan dari ESO.

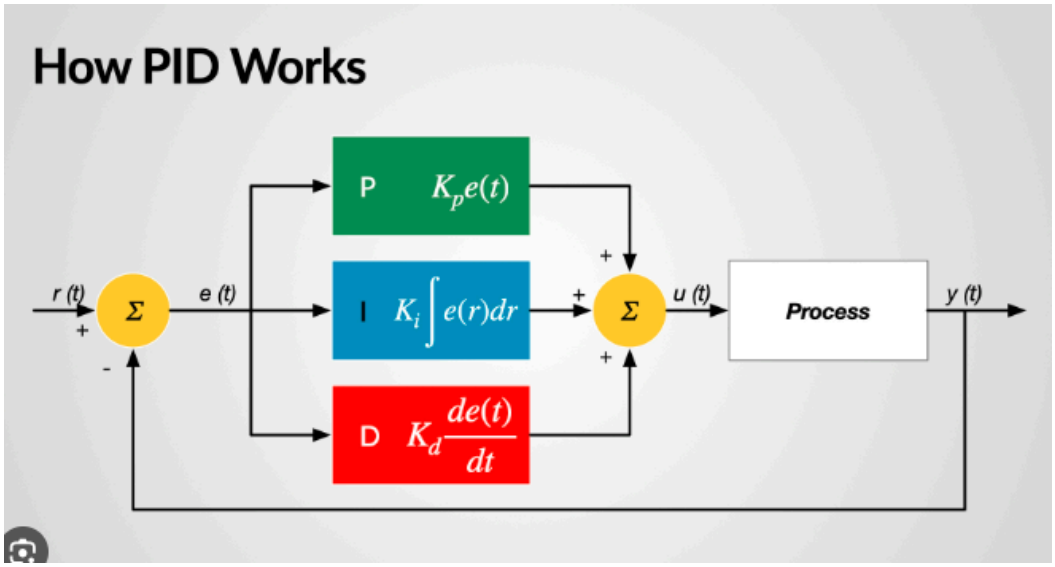
PID (Proportional-Integral-Derivative) control algorithms

PID (Proportional Integral Derivative) control algoritma sangat sering digunakan dalam sektor industri. Hal ini karena begitu kokohnya algoritma terhadap kondisi yang bervariasi dan fungsionalitas yang mudah dipahami sehingga dapat digunakan oleh insinyur. PID kontroler merupakan instrumen yang menerima data dari sensor, lalu menghitung selisih antara nilai aktual dan nilai atau hasil yang diinginkan, serta menyesuaikan keluaran agar sesuai dengan nilai atau hasil yang diinginkan. PID apabila digambarkan dalam bentuk diagram balok akan berupa mirip dengan sistem kontrol tertutup pada no 4.a.



Sumber : https://www.ni.com/en/shop/labview/pid-theory-explained.html?srsltid=AfmBOorQN_8HaT68L5jx8cEXIiHdgwqq36ISFYFX232Mp42vqGTRtTV8

Contoh penggunaan PID adalah saat mengatur AC, saya memberi nilai yang diinginkan yaitu 25 derajat celsius. Saat AC menjalankan tugas yang diberikan, AC akan mengukur udara yang terdapat di dalam ruangan. Apabila suhu di ruangan berbeda dengan suhu yang diinginkan, yaitu 25 derajat celsius, sinyal output aktuator yang dikeluarkan oleh algoritma kontroler akan memerintah sistem AC untuk menurunkan atau meningkatkan suhu sehingga suhu yang diinginkan akan tercapai.



Sumber : https://www.kevsrobots.com/resources/how_it_works/pid-controllers

Saat nilai yang dikeluarkan tidak sesuai dengan nilai yang diharapkan, maka eror akan dikirimkan ke awal loop dan diproses menjadi tiga bagian, yaitu Proportional, Integral, dan Derivative. Pada balok proportional, balok proportional akan mengirimkan output sinyal yang proporsional dengan sinyal eror. Rumus untuk di balok proportional adalah P output = Kp x eror. Balok proportional akan menyebabkan magnitude eror mendekati nilai yang diharapkan, namun hanya mendekatkan tidak akan pas pada nilai tersebut. Besarnya nilai Kp yang diberi perlu diperhatikan, apabila terlalu besar maka dapat menyebabkan nilai yang berlebih dan berosilasi, lalu kebalikannya jika terlalu kecil. Pada balok integral, output sinyal akan proporsional dengan magnitude dan durasi eror. Rumus balok integral adalah I output = Ki x integral eror. Pada balok derivative, output sinyal akan proporsional dengan tingkat perubahan dalam sinyal eror. Balok derivative berguna untuk memprediksi sinyal eror, dengan rumus D Output=Kd×dt/d(Error). Ketiga output sinyal tersebut akan digunakan untuk melakukan koreksi dalam sistem sehingga dapat mengubah nilai menjadi yang diinginkan.

A*(A Star) Algorithm

A star algoritma adalah Algoritma yang dianggap algoritma paling baik dalam pencarian jalur dan traversal grafik. A star biasanya digunakan untuk sistem navigasi seperti google maps

dan NPC(Non Playable Character) dalam gim 3D. Misalnya, saya berada di titik A dan saya ingin ke titik Z, lalu saya ingin menempuh ke titik Z dengan melalui jalur terpendek.

Menurut algoritma A star, kita definisikan n, lalu fungsi f, g, dan h. Dimana fungsi f melambangkan Total perkiraan biaya jalur melalui node n,, fungsi g melambangkan jarak antara node ,dan fungsi h melambangkan heuristic. Heuristic sendiri adalah biaya pergerakan untuk berpindah dari node tertentu peta ke tujuan akhir. Lambang n melambangkan noda berikut yang akan dilalui. Sehingga kita akan mendapatkan rumus :

$$f(n) = g(n) + h(n)$$

Tergantung layout dari peta, fungsi heuristik akan bernilai berbeda-beda, hal ini akan mempengaruhi efisiensi A star.Berikut adalah beberapa heuristik yang umum digunakan termasuk:

- Jarak Manhattan: Untuk grid dengan gerakan horizontal dan vertikal.
- Jarak Euclidean: Untuk ruang kontinu atau gerakan diagonal.
- Jarak Octile: Jika gerakan diagonal memiliki biaya berbeda

Berikut adalah langkah-langkah untuk mencari jarak terdekat agar mencapai tujuan dalam bentuk komputerisasi :

1. Inisialisasi:
    - Buat open list (daftar node yang akan dievaluasi) dan closed list (daftar node yang sudah dievaluasi).
    - Tempatkan node awal dalam open list dengan g(0)=0 dan f(0)=h(0).
2. Loop Utama:
    - Selama open list tidak kosong:
        1. Pilih Node Terbaik: Pilih node n dengan nilai f(n) terendah dalam open list dan sebut node tersebut adalah b.
        2. Hapus b dari daftar terbuka
        3. Buat 8 Node Suksesor dari bdan atur orang tua (parent) dari setiap suksesornya menjadi b
        4. Untuk setiap suksesor.
            - Jika suksesor adalah tujuan, hentikan pencarian..
            - Jika tidak, hitung nilai g dan h untuk suksesor.
                - Suksesor dari g = q.g + jarak antara suksesor dan q
                - Suksesor dari h = jarak dari tujuan ke suksesor (dapat dilakukan dengan beberapa cara: Manhattan, Diagonal, dan Euclidean
                - Suksesor dari f = suksesor.g + suksesor.h
            - Jika ada node dengan posisi yang sama dengan suksesor di daftar Terbuka yang memiliki f lebih rendah daripada suksesor, lewati suksesor ini.
            - Jika ada node dengan posisi yang sama dengan suksesor di daftar Tertutup yang memiliki f lebih rendah daripada suksesor, lewati suksesor ini. Jika tidak, tambahkan node ke daftar terbuka.

- Akhiri loop