a.) Kinematics (object detection, pose estimation, camera calibration)

Secara umum, algoritma kinematics adalah algoritma yang menggunakan pengaplikasian geometri (umumnya geometri euclid). Algoritma kinematics menggambarkan pergerakan mekanika secara matematis dengan konsep transformasi baik itu 2D maupun 3D.

- Object detection

Deteksi objek adalah algoritma yang memberi kemampuan pada robot untuk mendeteksi, mengklasifikasikan, dan melokalisasi objek di lingkungannya menggunakan sensor atau kamera.

Pada umumnya deteksi objek dilakukan dengan menggunakan big data berupa sampel objek yang akan diolah dengan machine learning, data tersebut berbentuk kumpulan pixel yang diubah menjadi matriks angka dan label tertentu sehingga dapat diolah di dalam neural network machine learning.

Jika suatu model machine learning sudah mampu mendeteksi objek dengan range tertentu, maka dapat diimplementasikan pada software robotika dan diintegrasikan dengan transformasi matematis untuk membuat fitur tambahan (misal, melakukan tracking sehingga robot mengikuti suatu objek).

- Pose estimation

Pose estimation adalah algoritma yang digunakan untuk melacak pose tubuh manusia menggunakan keypoint-keypoint yang dapat dilihat komputer melalui implementasi computer vision (menggunakan sensor, kamera, dan lain-lain).

Pose estimation dapat diimplementasikan menggunakan Direct Linear Transfrom algorithm dan Perspective-n-Point algorithm yang mengambil gambar titik-titik 2D dari kamera yang di korespondensi kan dengan lokasi titik-titik 3D di dunia nyata, korespondensi tersebut dapat dijadikan persamaan matematis sehingga dapat digunakan untuk mengetahui lokasi keypoint titik 3D melalui gambar 2D.

Selain menggunakan algoritma secara langsung, pose estimation juga dapat diterapkan menggunakan metode deep learning, menggunakan big data yang diproses dengan neural network multilapis.

- Camera calibration

Kalibrasi kamera adalah algoritma yang dapat menggunakan parameter-parameter kamera untuk melakukan berbagai aplikasi seperti mengurangi distorsi lensa, mengukur kedalaman objek di gambar, mengukur ukuran fisik objek melalui gambar, rekonstruksi 3D dari gambar 2D, dan lain-lain.

Parameter kamera yang dimaksud terdiri dari parameter eksternal dan internal. Parameter internal kamera diantaranya yaitu panjang focal, pusat optis kamera, dan koefisien skew (yang menggambarkan seberapa miring nya suatu pixel). Parameter eksternal diantaranya rotasi dan translasi kamera, serta posisi kamera relatif terhadap posisi objek 3D.

b.) ADRC (Active Disturbance Rejection Control)

ADRC adalah salah satu jenis sistem kontrol yang dapat diterapkan pada sistem dengan dinamika yang bervariasi dan memiliki gangguan internal serta eksternal. ADRC akan mengambil output dari sistem dan mengembalikannya dalam bentuk feedback loop ke sebuah controller.

Selain itu terdapat juga sebuah pengamat tambahan yang akan membantu controller mengatur gangguan, yaitu Extended State Observer (ESO). Dalam hal ini ESO akan menerima 2 hal, yaitu gangguan eksternal dan internal sebelum di proses sistem dan juga output dari sistem. Dari 2 hal tersebut, ESO akan menghasilkan sebuah “state” yang akan dikirim ke controller.

Sehingga controller akan menerima state dari extended state observer dan juga output dari sistem, kedua nilai itu akan dibandingkan dengan sebuah variabel referensi yang akan menjadi sebuah nilai error. Nilai error tersebut akan dikontrol oleh controller sehingga menjadi pertimbangan input selanjutnya.

Model perhitungannya yaitu sebagai berikut:

$$y(t) = b0 * u(t) + f(t)$$

- y(t) adalah output sistem

- u(t) adalah input sistem

- f(t) total gangguan yang diukur oleh extended state observer

- b0 adalah konstanta keuntungan yang menggambarkan respon sistem terhadap sebuah nilai input u(t). Nilai b0 dapat diaproksimasi dengan menjalankan sistem pada suatu jangkauan operasi tertentu terlebih dahulu.

Sistem kontrol ADRC memiliki performa yang lebih baik dibanding algoritma sistem kontrol lainnya seperti PID. Keunggulan ADRC diantaranya yaitu kemungkinan overshoot dalam menangani error sangat kecil dan pergerakan pengurangannya sangat stabil. Sistem juga cenderung lebih resistan terhadap gangguan eksternal.

c.) PID (Proportional-Integral-Derivative) control algorithms

PID adalah salah satu jenis feedback control algorithms yang digunakan untuk mengontrol input suatu sistem berdasarkan error yang dihasilkan output sistem tersebut. Pada sebuah loop sistem akan terdapat input yang diproses pada sebuah sistem menjadi output, kemudian output tersebut akan dibandingkan dengan sebuah variabel referensi sehingga menghasilkan sebuah nilai error (seberapa menyimpangnya output dari output yang diinginkan).

Nilai error tersebut akan akan diproses pada sebuah control algorithms yang bekerja berdasarkan prinsip PID (Proportional-Integral-Derivative). Hasil dari pemrosesan error tersebut akan dijadikan input dan error diproses dengan sebuah cara sehingga error tersebut akan semakin mendekati nol seiring berjalannya waktu.

Berikut merupakan cara kerja algoritma pemrosesan error tersebut:

- **Proportional Tuning:**

Proportional Tuning dilakukan dengan mengoreksi sebuah variabel secara proporsional terhadap selisih nya dengan error yang dihasilkan. Sehingga secara teoritis, variabel tersebut akan bergerak menuju target yang diinginkan dan error akan bergerak mendekati nol. Tetapi jika hanya menggunakan Proportional Tuning akan muncul steady state error, yaitu dimana error tidak akan pernah benar-benar mencapai nol sehingga dibutuhkan Integral Tuning.

- **Integral Tuning:**

Integral Tuning dilakukan dengan mengingat sebuah total dari error dan menambahkan terus total tersebut seiring berjalannya waktu. Total tersebut pun ditambah ke input selanjutnya. Ketika error masih positif, maka total error akan meningkat sehingga total integral error juga meningkat dan error bergerak semakin cepat mendekati nol. Namun hal ini dapat menyebabkan overshoot (melebihi target yang diinginkan), sehingga mungkin input akan berosilasi antara overshoot dan undershoot selama beberapa kali sebelum mencapai nilai error nol.

- Derivative Tuning:

Untuk mencegah osilasi tidak stabil tersebut, ditambahkan Derivative Tuning. Hal ini dilakukan dengan cara menambahkan ke input tingkat perubahan error terhadap waktu. Hal ini akan membuat lebih stabil dan mengurangi overshoot karena ketika semakin turun error mendekati nol, semakin dikurangi koreksi yang dilakukan.

Ketiga proses Tuning tersebut dapat disesuaikan dengan kondisi tertentu yang diinginkan (misal, nilai integral tuning bisa lebih diprioritaskan dibanding derivative tuning). Sehingga bisa mengontrol error secara otomatis.

d.) A* (A star) algorithm

A* adalah algoritma pencarian jarak terpendek pada suatu graf dengan sebuah node awal dan node akhir yang terdefinisi. Algoritma ini bekerja secara heuristic, artinya ia menggunakan fungsi heuristic yang berisi perkiraan “harga” untuk mencapai node tujuan dari node tertentu, algoritma pun akan memprioritaskan node yang lebih “menjanjikan”.

Berikut merupakan langkah-langkah implementasinya:

1.) Definisikan “himpunan terbuka” yaitu himpunan node yang belum dikunjungi, dan “himpunan tertutup” yaitu himpunan node yang sudah dikunjungi. Awalnya semua node tidak berada di himpunan terbuka atau himpunan tertutup.

2.) Definisikan g(n) sebagai jarak antara node n dengan node awal.

3.) Definisikan h(n) sebagai jarak antara node n dengan node tujuan.

4.) Masukkan node awal ke himpunan terbuka

5.) Ambil sebuah node di himpunan terbuka yang memiliki nilai g(n)+h(n) paling kecil, jika terdapat nilai yang sama ambil node dengan nilai h(n) paling kecil. Node ini dinamakan “node saat ini”. Tandai node saat ini sebagai “telah di kunjungi” dan pindahkan dari himpunan terbuka ke himpunan tertutup.

6.) Jika node saat ini adalah node tujuan, telah ditemukan shortest path yaitu node di himpunan terbuka yang menghubungkan node awal dan node tujuan. Keluar dari algoritma.

7.) Untuk setiap tetangga node saat ini, jika tetangga tidak bisa dicapai atau berada di himpunan tertutup maka hiraukan tetangga ini. Jika tidak, maka cek tetangga ini.

8.) Dalam pengecekan, jika tetangga tidak berada di himpunan manapun maka ubah nilai g(tetangga) dan nilai h(tetangga) sesuai dengan shortest path yang telah ditemukan sejauh ini. Kemudian masukkan tetangga ke himpunan terbuka.

9.) Dalam pengecekan, jika tetangga berada di himpunan terbuka dan nilai g(tetangga) ataupun nilai h(tetangga) bisa dikurangi berdasarkan shortest path sejauh ini maka ubah nilai g(tetangga) dan nilai h(tetangga) sesuai dengan shortest path yang telah ditemukan sejauh ini.

10.) Kembali ke langkah (5).