URO TEST ITB

SOAL NOMOR 4B

Muhammad Faiz Alfada Dharma - 19624244

Kinematics

1. **Objection detection**

   Objection detection digunakan untuk mendeteksi objek visual / *visual objects* dari kelas tertentu (seperti manusia, hewan, mobil, atau bangunan) dalam gambar digital, seperti foto. *Object detection* memanfaatkan sifat-sifat khusus dan unik untuk mengidentifikasi objek yang diperlukan. Berikut adalah beberapa algoritma untuk *object detection*:

   - Region-based Convolutional Neural Networks (R-CNN)

     Dalam model R-CNN akan mengekstraksi fitur / *extract features* yang paling penting (biasanya sekitar 2000 fitur) dengan menggunakan fitur selektif / *selective features*. Proses pemilihan ekstraksi dapat dihitung dengan bantuan *selective search algorithm* untuk mendeteksi objek.
   - YOLO (You Only Look Once)

     YOLO menggunakan tiga terminologi utama untuk mencapai tujuan *object detection*. Pertama yaitu *residual blocks*, yang digunakan untuk membuat sebuah *grid* pada gambar. Setiap *grid* berperan sebagai titik pusat untuk prediksi. Kedua yaitu pembuatan *bounding boxes* berdasarkan titik pusat prediksi. Ketiga yaitu penggunaan *Intersection over Union* (IOU) untuk menghitung *bounding box* untuk *object detection.*

2. **Pose estimation**

   Pose estimation adalah proses yang melibatkan pendeteksian dan pelacakan posisi serta orientasi bagian tubuh manusia dalam sebuah gambar atau video. *Pose estimation* memanfaatkan *pose* dan orientasi untuk memprediksi dan melacak lokasi seseorang atau objek. Umumnya, *pose estimation* menggunakan dua langkah, yaitu mendeteksi *human bounding boxes* dan memperkirakan *pose*. *Pose estimation* bekerja dengan menemukan *key points*, seperti siku, lutut, pergelangan tangan, dan lainnya. *Pose estimation* terdiri dari dua jenis, yaitu *single-pose* untuk mendeteksi objek tunggal dan *multi-pose* untuk mendeteksi *pose* beberapa objek.

3. **Camera calibration**

Camera Calibration adalah proses memperkirakan parameter kamera / *camera parameters* dengan menggunakan gambar yang berisi *calibration pattern*. Parameter tersebut terdiri dari *camera intrinsics, distortion coefficients,* dan *camera extrinsics*. Parameter ini digunakan untuk menghilangkan *lens distortion effects* dari gambar, mengukur objek planar / *planar objects*, rekonstruksi 3D *scenes* dari beberapa kamera, dan lainnya. Untuk memperkirakan parameter kamera / *camera parameters*, diperlukan *3D world points* dan *2D image points* yang sesuai. Ini dapat diperoleh dengan menggunakan beberapa gambar pola kalibrasi / *calibration pattern*, seperti papan catur. Setelah mengkalibrasi kamera, untuk mengevaluasi akurasi parameter yang diperkirakan, dapat melakukan hal-hal berikut:

- Plot *relative location* antara kamera dan pola kalibrasi / *calibration pattern*.
- Menghitung *reprojection errors*.
- Menghitung *parameter estimation errors*.

ADRC (Active Disturbance Rejection Control)

Active Disturbance Rejection Control (ADRC) adalah *model free control technique* yang berguna untuk merancang *controller* untuk pembangkit dinamik / *plant dynamics,* gangguan internal, gangguan eksternal yang tidak diketahui. Algoritma ini hanya memerlukan perkiraan dari *plant dynamics* untuk merancang *controllers* yang menyediakan *disturbance rejection* yang tidak melampaui batas.

Kita dapat menggunakan *Active Disturbance rejection Control* (ADRC) *block* untuk mengimplementasikan ADRC. Blok tersebut menggunakan perkiraan model orde pertama / *first order* atau orde kedua / *second order* dari sistem dinamik / *dynamic system* yang diketahui dan gangguan yang tidak diketahui yang dimodelkan sebagai *extended state of the plant.*

- *First order approximation* / perkiraan model orde pertama:

$$\dot{y}(t) = b_0 u(t) + f(t)$$

- *Second order approximation* / perkiraan model orde kedua:

$$\ddot{y}(t) = b_0 u(t) + f(t)$$

y(t) adalah *plant output*

u(t) adalah *input signal*

bo adalah *critical gain,* yaitu perkiraan penguatan / *estimated gain* yang menggambarkan respon pembangkit / *plant response* terhadap suatu input **u(t)**

f(t) adalah total gangguan / *total disturbance,* yang termasuk dinamik / *plant dynamics,* gangguan internal, gangguan eksternal yang tidak diketahui

Blok tersebut menggunakan *Extended State Observer (*ESO) untuk memperkirakan f(t) dan mengimplementasikan *disturbance rejection* dengan mengurangi efek perkiraan gangguan pada bagian perkiraan model yang diketahui.

PID (Proportional Integral Derivative) Control Algorithms

PID (Proportional Integral Derivative) *controller* adalah sebuah perangkat yang menerima input data dari sensor, menghitung perbedaan antara *actual values* dan *desired setpoints*, lalu menyesuaikan outputnya untuk mengendalikan variabel seperti suhu, *flow rate*, kecepatan, tekanan, dan tegangan. Proses Ini dilakukan melalui tiga mekanisme:

1. *Proportional control,* yang merespon terhadap kesalahan saat ini
2. *Integral control,* yang menangani kesalahan masa lalu
3. *Derivative control,* yang memprediksi kesalahan di masa depan

PID *controller* menjumlahkan ketiga komponen tersebut untuk menghitung output. Hal ini memungkinkan PID *controllers* secara efisien mempertahankan *process control* dan stabilitas sistem. Output dari PID *controller* mengikuti formula berikut:

$$u(t) = ke(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt},$$

u adalah *control signal* dan **e** adalah *control error*. Sinyal kontrol adalah jumlah dari tiga variabel, yaitu P-*term* (*proportional to the error*), I-*term* (*proportional to the integral of the error*), dan D-*term* (*proportional to the derivative error*).

- K = *proportional gain*
- Ki = *integral gain*
- Kd = *derivative gain*

Proses mengatur *optimal gains* untuk P (*proportional)*, I (*integral)*, dan D (*derivative)* dengan tujuan mendapatkan respons ideal dari *system control* disebut ***tuning***. PID *controller* dapat diperoleh melalui metode *trial and error*. Dalam metode ini, I dan D diatur ke nol terlebih dahulu, kemudian *proportional gain* ditingkatkan hingga output dari *loop* berosilasi. Dengan meningkatkan *proportional gain, maka* akan mempercepat sistem, tetapi harus tetap menjaga kestabilan. Setelah P disesuaikan, *integral* ditingkatkan untuk mengurangi osilasi. Istilah *Integral* akan mengurangi *steady state error*. Setelah P dan I diatur, *derivative* akan ditingkatkan hingga *loop* menjadi cepat untuk mencapai *set point*. Meningkatkan *derivative* akan mengurangi overshoot dan menjaga stabilitas. Sistem juga akan menjadi lebih sensitif terhadap *noise.*

A* (A star) Algorithm

A* (A Star) *algorithm* adalah algoritma pencarian yang digunakan untuk menemukan jalur terpendek antara titik awal dan titik akhir. A* (A Star) *algorithm* sering digunakan untuk menemukan rute terpendek. Algoritma ini dirancang untuk masalah *graph traversal* untuk membantu membangun robot yang dapat menemukan jalannya sendiri. A* (A Star) *algorithm* akan mencari jalur terpendek terlebih dahulu sehingga menjadikannya optimal. Aspek yang membuat A* (A Star) *algorithm* sangat kuat adalah implementasinya pada *weighted graphs*. *Weighted graphs* menggunakan angka untuk mewakili *cost* dalam mengambil setiap jalur atau tindakan. Hal Ini memungkinkan untuk memilih jalur dengan *cost* terendah dan menemukan rute terbaik dalam hal jarak dan waktu.

Semua grafik memiliki *node* atau titik yang harus dilewati oleh algoritma untuk mencapai *node* akhir / *final node*. Jalur di antara *node-node* ini memiliki nilai numerik / *numerical value* yang dianggap sebagai *weight of the path*. Jumlah total semua jalur yang dilintasi memberi sebuah *cost* dari rute tersebut.

Awalnya, algoritma ini menghitung *cost* untuk semua *node* tetangga / *neighbouring nodes* terdekatnya (n) dan memilih yang memiliki *cost* yang paling rendah. Proses ini akan berulang hingga tidak ada *node* baru yang bisa dipilih dan sampai semua jalur telah dilalui. Setelah itu, algoritma ini akan mempertimbangkan jalur terbaik di antaranya. Jika f(n) mewakili *cost* akhir / *final cost*, maka dapat dinyatakan sebagai:

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

- g(n) = *cost of traversing* / perpindahan dari satu *node* ke *node* lain.
- h(n) = perkiraan heuristik / *heuristic approximation* nilai *node*, bukan nilai nyata tetapi perkiraan nilai *cost*.

A* (A Star) *algorithm* memiliki beberapa keunggulan:

1. Algoritma ini akan menjamin menemukan jalur yang optimal.
2. Algoritma ini efisien dan bisa menangani ruang pencarian yang besar / *large search spaces* dengan efektif dengan memangkas jalur.
3. Algoritma ini fleksibel dan mudah beradaptasi pada *terrain costs* yang bervariasi.

Selain adanya keunggulan, A* (A Star) *algorithm* juga memiliki beberapa kelemahan:

1. Dalam beberapa kasus, Algoritma ini bisa memiliki biaya komputasinya yang tinggi / *computationally expensive,* terutama ketika ruang pencarian / *search space* sangat luas dan jumlah jalur yang banyak.
2. Algoritma ini juga dapat memakan banyak memori dan *processing resources*.
3. Algoritma ini sangat bergantung pada kualitas fungsi heuristik / *heuristic function*. Jika heuristik tidak dirancang dengan baik atau tidak akurat dalam memperkirakan jarak ke tujuan, kinerja algoritma ini bisa terpengaruh.
4. Algoritma ini juga dapat mengalami kesulitan dengan grafik atau ruang pencarian / *search space* yang memiliki struktur tidak teratur.

Sumber Referensi:

https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm
https://www.ni.com/en/shop/labview/pid-theory-explained.html#section--2054068861
https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID
https://www.cds.caltech.edu/~murray/courses/cds101/fa04/caltech/am04_ch8-3nov04.pdf
https://neptune.ai/blog/object-detection-algorithms-and-libraries
https://www.mathworks.com/help/vision/camera-calibration.html
https://www.mathworks.com/help/vision/ug/camera-calibration.html
https://viso.ai/deep-learning/pose-estimation-ultimate-overview/
https://www.mathworks.com/help/slcontrol/ug/active-disturbance-rejection-control.html