1. **Kinematics (Object Detection, Pose Estimation, Camera Calibration):** Kinematics adalah ilmu yang mempelajari gerak benda tanpa memperhatikan penyebab gerakan. Dalam konteks pemrograman robot dan computer vision, kinematics sering kali digunakan bersama teknik object detection, pose estimation, dan camera calibration untuk memahami posisi dan orientasi objek di ruang 3D.
    - **Object Detection:** Sebuah proses mengidentifikasi dan menentukan lokasi objek di dalam gambar atau video. Algoritma ini melibatkan teknik seperti convolutional neural networks (CNNs) atau model YOLO (You Only Look Once) yang dapat mendeteksi objek secara realtime.
    - **Pose Estimation:** Pose Estimation adalah proses menentukan posisi dan orientasi objek dalam ruang 3D berdasarkan citra 2D. Pose estimation dapat digunakan untuk melacak objek seperti tangan manusia, wajah, atau tubuh secara keseluruhan. Algoritma umum yang digunakan termasuk **PNP (Perspective-N-Point)**, yang mengambil serangkaian titik 3D yang sesuai dengan titik 2D dalam gambar untuk menghitung transformasi objek di dunia nyata.
    - **Camera Calibration:** Camera Calibration adalah proses untuk menentukan parameter intrinsik (fokus, distorsi lensa, dsb.) dan ekstrinsik (posisi dan orientasi kamera di ruang 3D) kamera. Ini penting untuk menerjemahkan koordinat gambar 2D ke dunia nyata. Algoritma **Zhang's Method** sering digunakan untuk kalibrasi kamera, di mana gambar pola kalibrasi (misalnya, checkerboard) digunakan untuk mengestimasi parameter kamera.
2. **ADRC (Active Disturbance Rejection Control):** Active Disturbance Rejection Control (ADRC) adalah metode kontrol yang digunakan untuk mengatasi gangguan eksternal dan internal dalam sistem tanpa memerlukan model matematis yang sangat akurat dari sistem tersebut. Ini membuat **ADRC** lebih fleksibel dibandingkan dengan kontrol berbasis model lainnya seperti **PID**. Komponen Utama ADRC:
    - Extended State Observer (ESO)
    - Tracking Differentiator (TD)
    - Nonlinear State Error Feedback (NLSEF)

    Keuntungan ADRC:
    - Tidak memerlukan model yang sangat akurat dari sistem.
    - Mampu menolak gangguan tanpa memerlukan tuning berlebihan.

- Lebih robust terhadap ketidakpastian sistem dan perubahan parameter.

3. **PID (Proportional-Integral-Derivative) Control Algorithms:** PID control adalah algoritma kontrol feedback yang banyak digunakan dalam industri untuk mempertahankan nilai keluaran proses sesuai dengan setpoint yang diinginkan dengan meminimalkan error secara dinamis.
    - **Komponen PID**:
        - **Proportional (P)**: Menghasilkan output yang proporsional terhadap error saat ini. Semakin besar error, semakin besar koreksi yang diberikan.
        - **Integral (I):** Menghitung integral dari error dari waktu ke waktu, memperbaiki akumulasi error yang belum teratasi oleh komponen P. Ini berguna untuk menghilangkan steady-state error.
        - **Derivative (D)**: Mengukur laju perubahan error, membantu mengantisipasi perubahan di masa depan. Ini membantu mencegah overshoot dan meningkatkan stabilitas sistem.
    - **Kelebihan PID**:
        - Dapat mengontrol berbagai jenis sistem.
        - Parameter tuning sederhana tetapi sangat efektif.
        - Sering digunakan dalam sistem real-time seperti sistem suhu, motor servo, dan pengontrol tekanan.

4. **A* Algorithm (A star):** A* Algorithm adalah algoritma pencarian jalur yang banyak digunakan dalam kecerdasan buatan dan robotika untuk menemukan jalur terpendek dari titik awal ke tujuan pada graf atau peta grid. Ini adalah salah satu algoritma pencarian jalur yang optimal dan lengkap.
    - **Cara Kerja:**
        - A* menggunakan dua nilai **g(n)** dan **h(n)** untuk menilai setiap node:

        - **g(n)**: jarak yang telah ditempuh (dari node awal sampai node saat ini).
        - **h(n)**: Estimasi jarak dari node saat ini ke tujuan, biasanya disebut sebagai **heuristik**.
        - **f(n)**: Jumlah dari g(n) dan h(n), yaitu **f(n)=g(n)+h(n)**
    - Algoritma A* memilih node dengan nilai **f(n)** terkecil, mengembang ke node node lain, lalu memperbarui nilai **g(n)** dan **h(n)** untuk node-node yang di laluinya. Proses ini berlanjut hingga mencapai node tujuan.
- **Kelebihan A*:**
    - Optimal dan efisien untuk menemukan jalur terpendek.
    - Heuristik membuatnya lebih cepat dibandingkan pencarian algoritma seperti Dijkstra dalam kasus tertentu.
- **Kekurangan A*:**
    - Kinerja algoritma sangat tergantung pada heuristik yang digunakan.
    - Bisa menjadi lambat untuk lingkungan yang sangat besar dan rumit, meskipun lebih efisien dibandingkan metode brute-force.