Project 8: Lateral Velocity Control
So far, we have focused on orientation control
  • Proportional error: relative to a goal
  • Damping: prefers zero rotational velocity

Next step:
  • Estimate actual lateral velocity from the cameras
  • High level controller specifies a desired lateral velocity
  • Use lateral acceleration to “close the gap” between desired and sensed lateral velocity
Estimating Lateral Velocity

You already have implemented:

```c
void accumulate_slip(int32_t adx[3], int32_t ady[3])
```

- Update adx/ady with slip information from each of the cameras
- Note: now, we will only accumulate slip over 5ms

```c
void compute_chassis_motion(int32_t adx[3], int32_t ady[3], float[3] motion)
```

- Translate slip into movement of the chassis
Smoothing Velocities

• From one 5ms step to the next, the number of pixels slipped can vary a lot (especially when velocity is low)
• In order to address this sampling noise, we will filter our velocity estimates

• New global variable:

```plaintext
float velocity_filtered[3]; // x_dot, y_dot, theta_dot
```
Instantaneous Velocity

Your function `compute_chassis_motion()` gives us movement of the chassis within the last 5ms: call this $dx$.

- Our instantaneous estimate of velocity is: ??
Smoothing the Velocity Estimate

“Low pass filter”: remove the high frequency components of some signal

• In our case, we assume that the true velocity is slowly changing and that sampling noise manifests itself as high-frequency changes
Velocity Control

• High-level specifies desired velocity
• Controller chooses acceleration to close the distance between desired and actual velocity
Low-Pass Filter in Code

\[ fv = fv \times (1 - dt / \tau) + dx / \tau; \]