EECS192 Lecture 8
Mar. 12, 2019

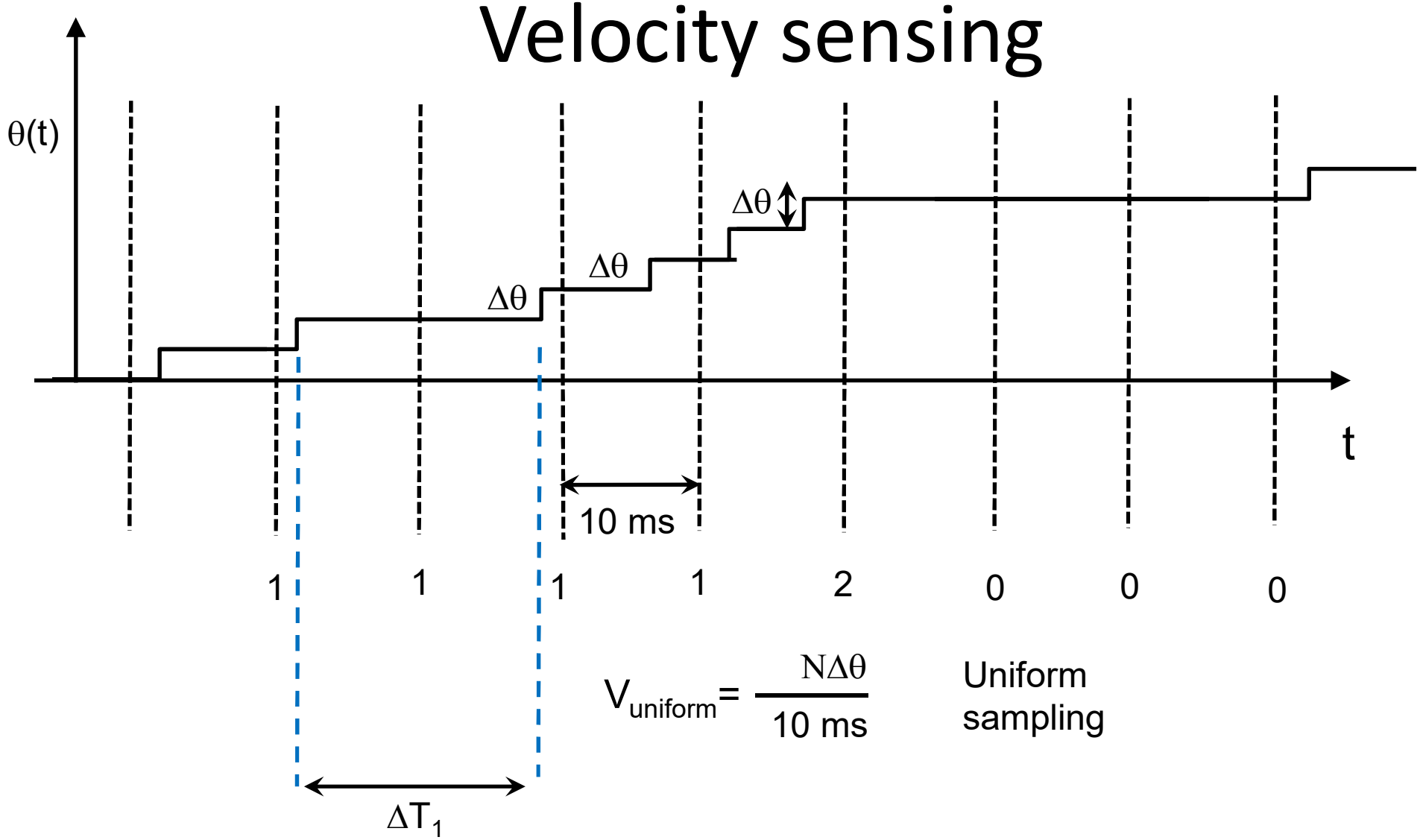
Notes:

1. Check off-
    1. 3/15/2019: C6 Benchtop line tracking
    2. 3/22/2019 C7 everything on car, drop-and-run, (velocity control), open loop fig 8
    3. 3/23/2018 C8: Closed loop figure 8 drop and run
2. Quiz 4 line sensor 3/19
3. Community Spirit: PCB peer review, Piazza, helping fellow students
4. CalDay Sat. April 13 @ UCB,
5. Line sensor processing HW1 due 3/19 – upload Python code to bcourses. (Will run on other data.)
6. Waterfall plot for line data
7. Lab safety/hygiene

Topics

- Speed sensing/velocity control review
- Velocity control detail
- Line sensor conclusion
- Software: MPU interrupt + threads
- Feedback control overview
- Bicycle steering model

Velocity sensing



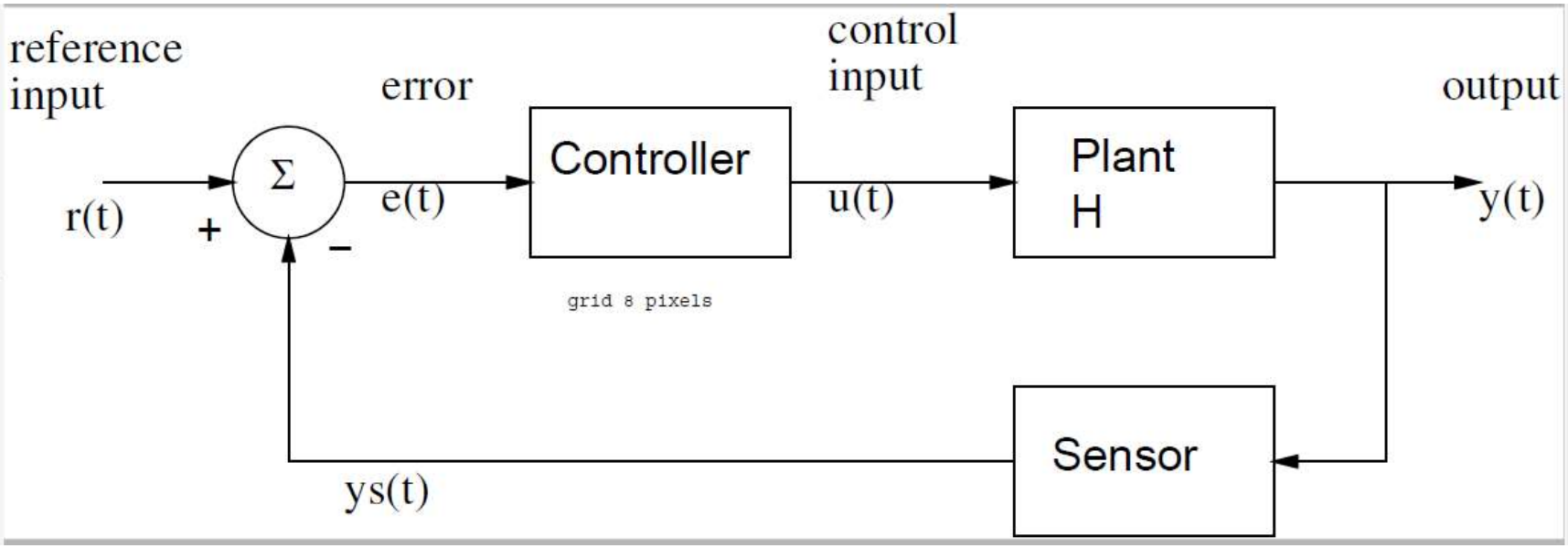
$$V_{uniform} = \frac{N\Delta\theta}{10\ \text{ms}}$$

Uniform sampling

$$V_{edge} = \frac{\Delta\theta}{\Delta T_1}$$

Edge timing

Velocity control overview



On board…

Proportional control:

U = kp*e = kp* (r-y);

Proportional + integral control

```
U = kp*e + ki * e_sum;
e_sum = e_sum + e;
```

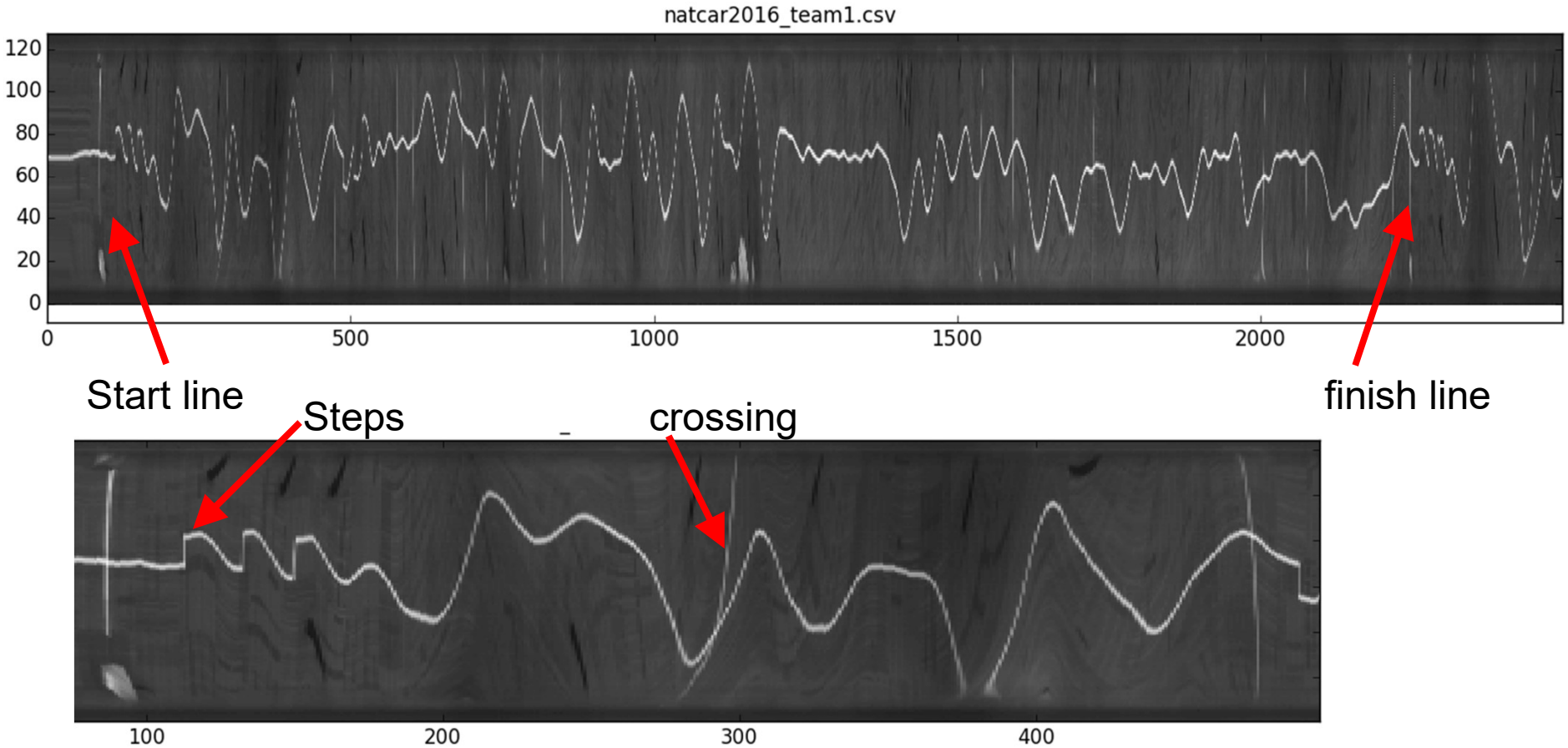
Velocity sensing (recap)

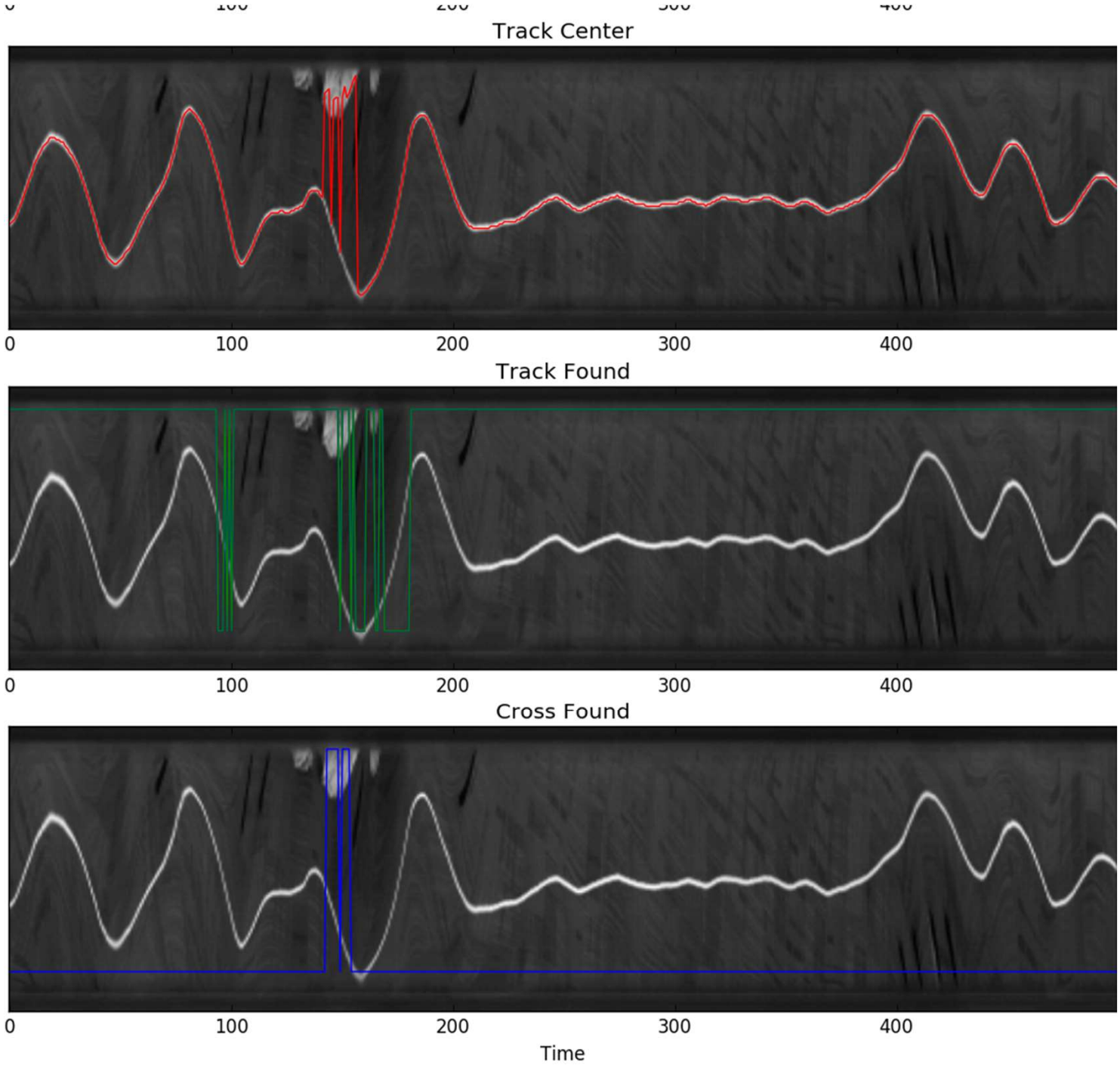
V~ (change in angle)/(change in time)

On board…

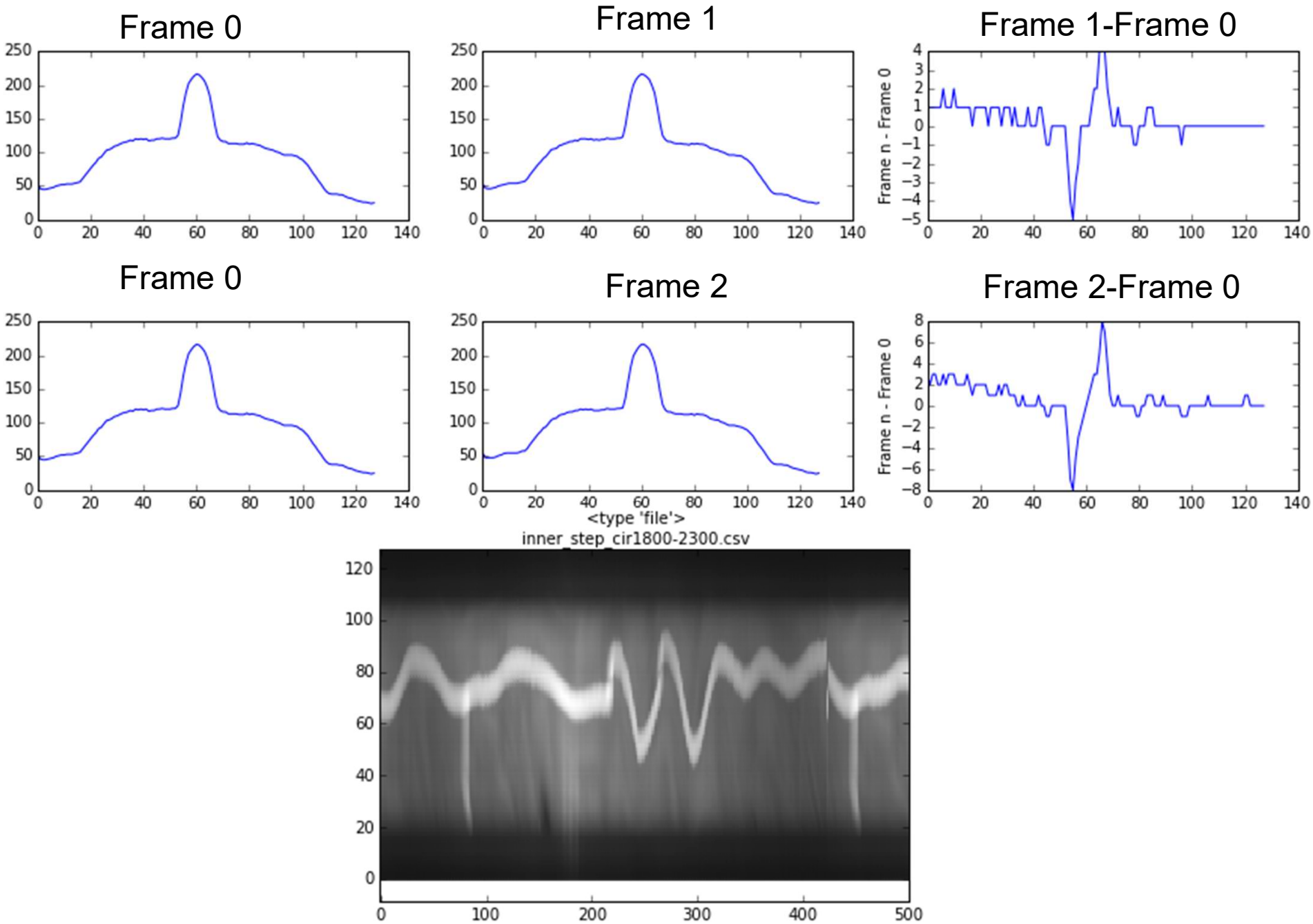
Python Template

track_center_list - A length n array of integers from 0 to 127.
Represents the predicted center of the line in each frame.
track_found_list - A length n array of booleans.
Represents whether or not each frame contains a detected line.
cross_found_list - A length n array of booleans.
Represents whether or not each frame contains a crossing.




Track Center
0
100
200
300
400
Track Found
0
100
200
300
400
Cross Found
0
100
200
300
400
Time

TSL 1401 line sensor NATCAR 8 bit



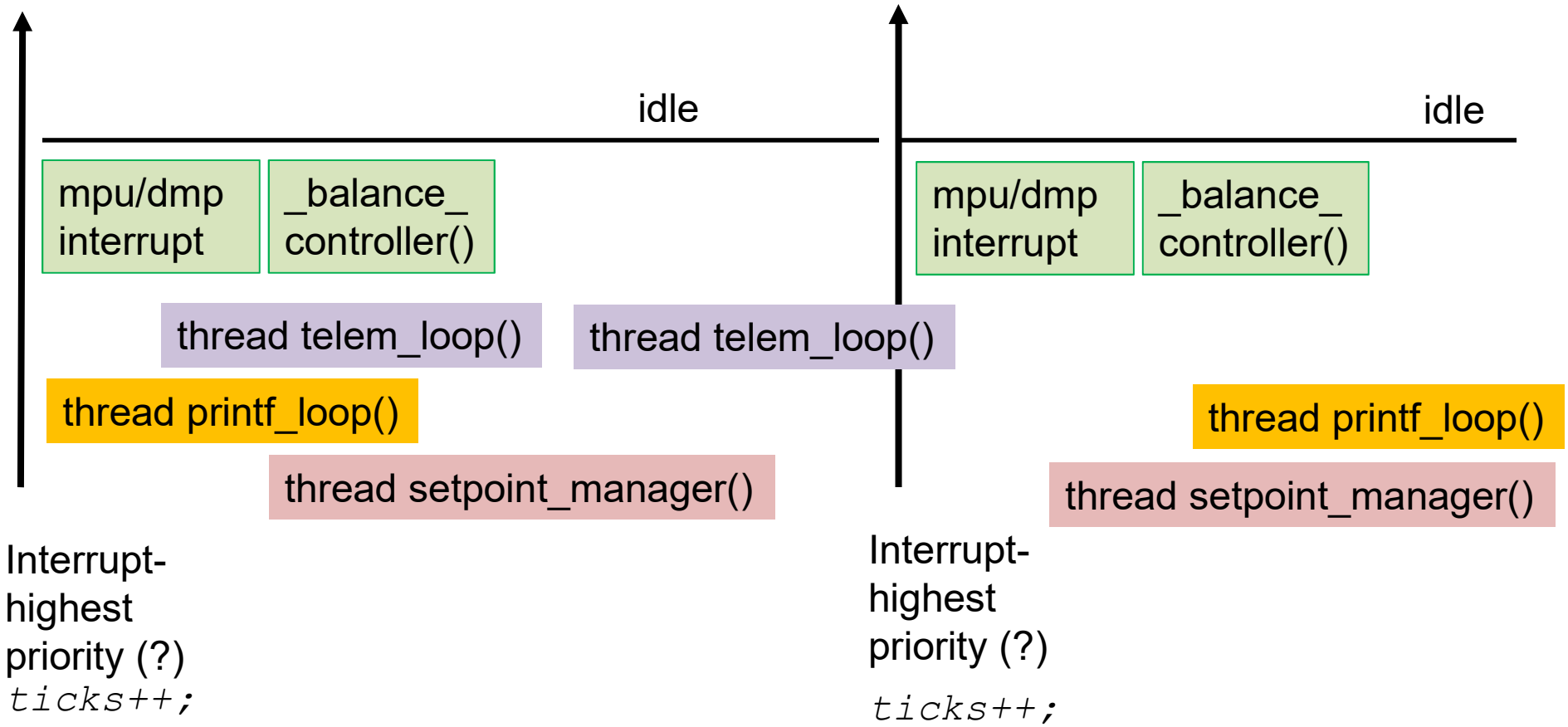
Possible algorithms for line detection

e.g. scipy.signal.filter

- Subtraction- to find left and right edge of line (ok if not noisy, somewhat lighting invariant)
- Difference of gaussians (idea is to smooth then differentiate)
- Correlation (best match position for known features)
  - scipy.signal.correlate

Software Notes

Read sensors ➔ process ➔ output ….. Idle ……. Read sensors ➔ process ➔ output



Threads are asynchronous wrt interrupt!

`rc_pthread_set_process_niceness()` ?

rc_balance2.c using gyro/MPU

When new data is ready in the buffer, the IMU sends an interrupt to the BeagleBone triggering the buffer read followed by the execution of a function of your choosing set with the rc_mpu_set_dmp_callback() function.

```
// set up mpu configuration
rc_mpu_config_t mpu_config = rc_mpu_default_config();
mpu_config.dmp_sample_rate = SAMPLE_RATE_HZ;

// start mpu
if(rc_mpu_initialize_dmp(&mpu_data, mpu_config))

// this should be the last step in initialization
// to make sure other setup functions don't interfere
rc_mpu_set_dmp_callback(&__balance_controller);

// idle while sensing and control done elsewhere
while(rc_get_state()!=EXITING){
       rc_usleep(200000);  }
```

rc_balance2.c __balance_controller()

```
static void __balance_controller(void)
{ticks++;
 end_time = rc_nanos_since_boot();
 run_time = end_time - start_time;
// time since previous interrupt

/*****************************************************************
* STATE_ESTIMATION
* read sensors and compute the state
*****************************************************************/
cstate.wheelAngleL =
(rc_encoder_eqep_read(ENCODER_CHANNEL_L) * 2.0 * M_PI) \
/(ENCODER_POLARITY_L * GEARBOX * ENCODER_RES);
/*******************************************************
* Send signal to motors
*******************************************************/
dutyL = cstate.d1_u - cstate.d3_u;
rc_motor_set(MOTOR_CHANNEL_L, MOTOR_POLARITY_L * dutyL);
}
```

rc_balance2.c: threads

// Note that using anything other than SCHED_OTHER with priority 0 is only available to root

```
int main(int argc, char *argv[])
{ int c;
   pthread_t setpoint_thread = 0;
   pthread_t printf_thread = 0;
   pthread_t telem_thread = 0;
…
// print thread to print to screen without blocking main
rc_pthread_create(&printf_thread, __printf_loop, (void*) NULL,
SCHED_OTHER, 0);
…

// start balance stack to control setpoints
rc_pthread_create(&setpoint_thread, __setpoint_manager,
(void*) NULL, SCHED_OTHER, 0);
…

// telemetry thread to log to file
rc_pthread_create(&telem_thread, telem_loop, (void*) NULL,
SCHED_OTHER, 0);
// telem loop could write to file
```

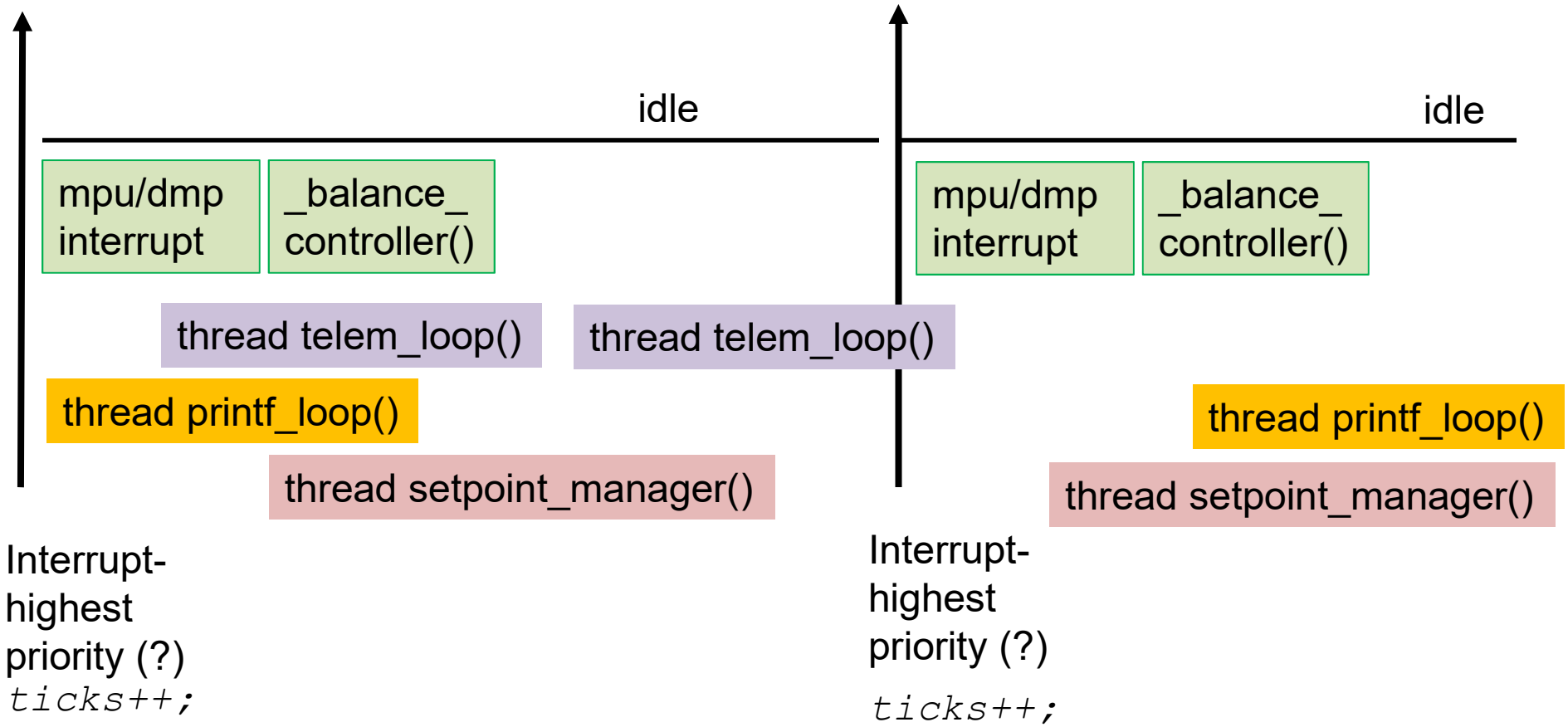
TimingTest.c: how long does fprintf take?

```
while(rc_get_state()!=EXITING)
{  // just data for csv format
        current_time = rc_nanos_since_boot() - start_time;
        old_tick = ticks;
        fprintf(logfile, "%ld, ", old_tick);  // pass value which not changing by other process
                current_time_f = ((double) current_time)/ 1e6; // milliseconds
                run_time_f = ((double) run_time)/1000.0; // us

                fprintf(logfile,"%8.3lf, %8.3lf, ", current_time_f, run_time_f);
                fprintf(logfile, "%" PRIu64 ", ",current_time);
                fprintf(logfile, "%" PRIu64 "\n",run_time);

                end_time = rc_nanos_since_boot() - start_time;
                run_time = end_time - current_time;
                while(old_tick == ticks)
                { rc_usleep(100); // sleep 100 us
                }
}
```

run_time: min 20 us, typical 30-50 us, max 6600 us

Software Notes

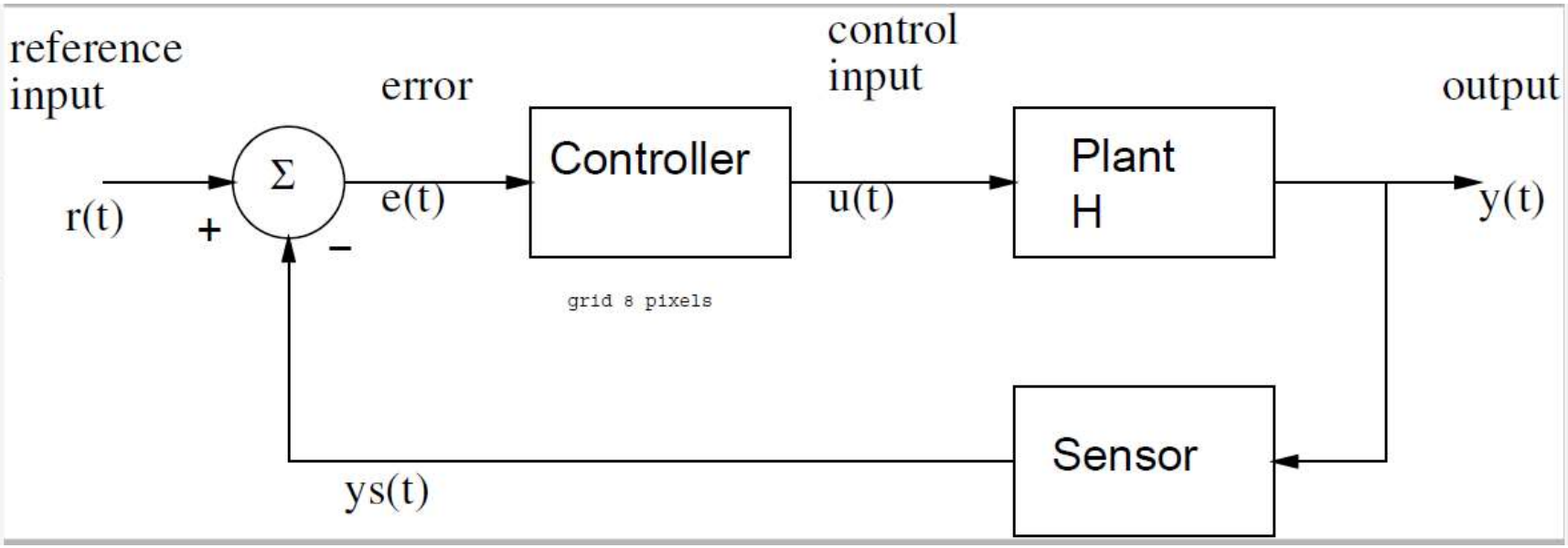
Read sensors ➔ process ➔ output ….. Idle ……. Read sensors ➔ process ➔ output



Threads are asynchronous wrt interrupt!

`rc_pthread_set_process_niceness()` ?

Control overview



On board…

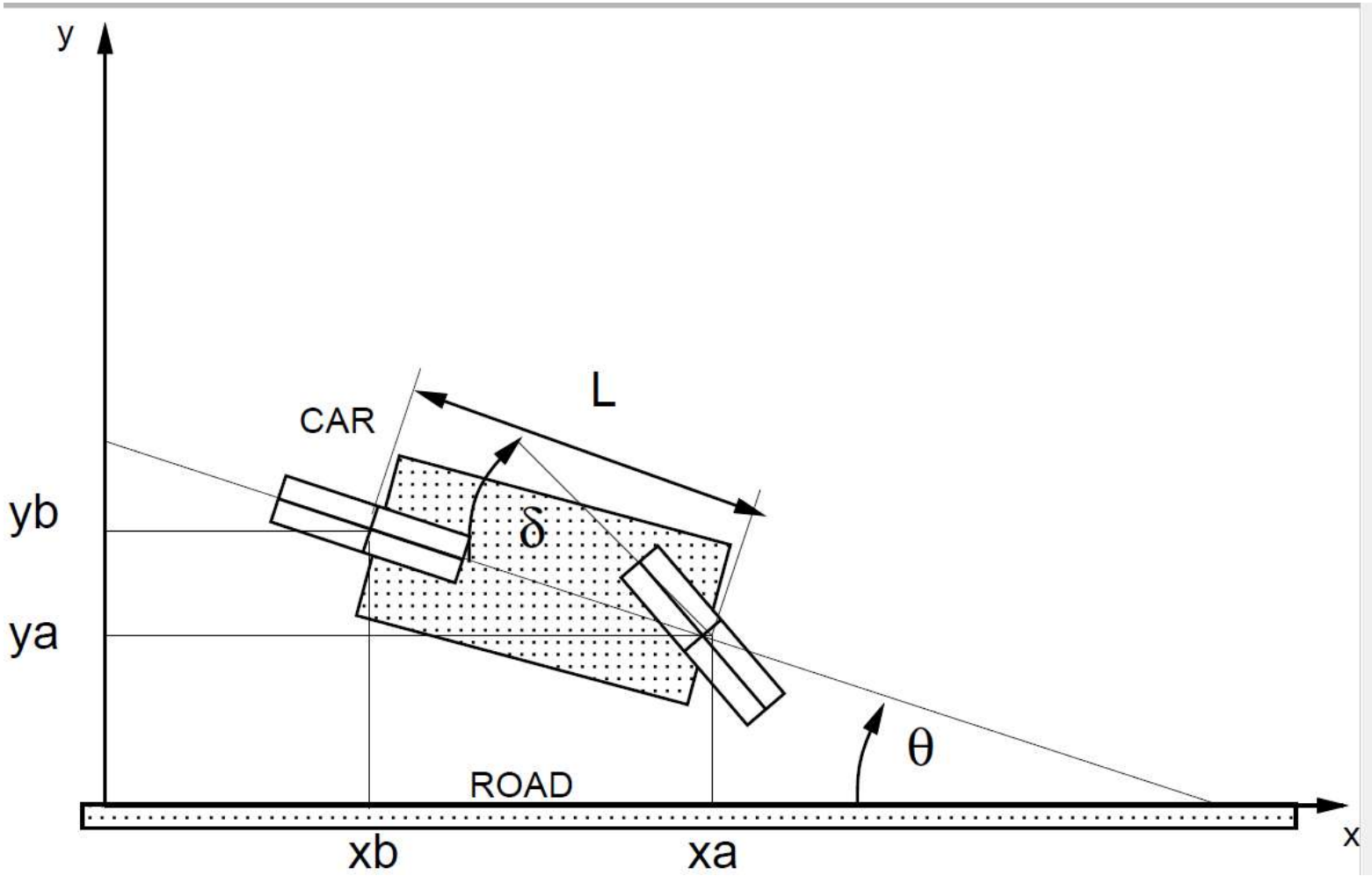
Proportional control:

U = kp*e = kp* (r-y);

Proportional + integral control

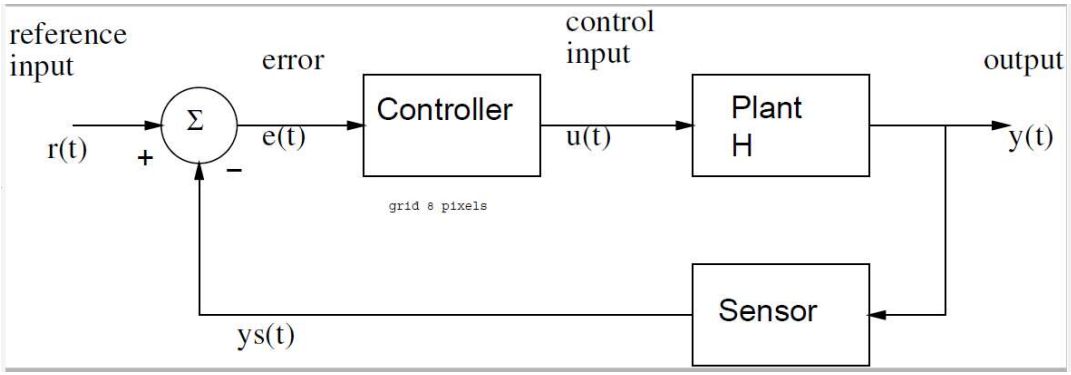
```
U = kp*e + ki * e_sum;
e_sum = e_sum + e;
```

Bicycle Steering Model

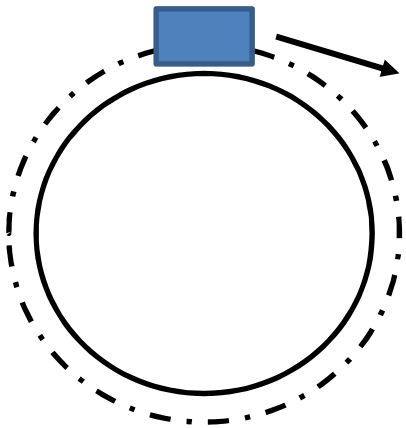


Equations On board

Bicycle Steering Control



Note steady state error: car follows larger radius



Proportional control:
r = 0 (to be on straight track)
δ=u = kp*e

Proportional+derivative

P+I+D

On board