EECS192 Lecture 8
Mar. 6, 2018

Notes:

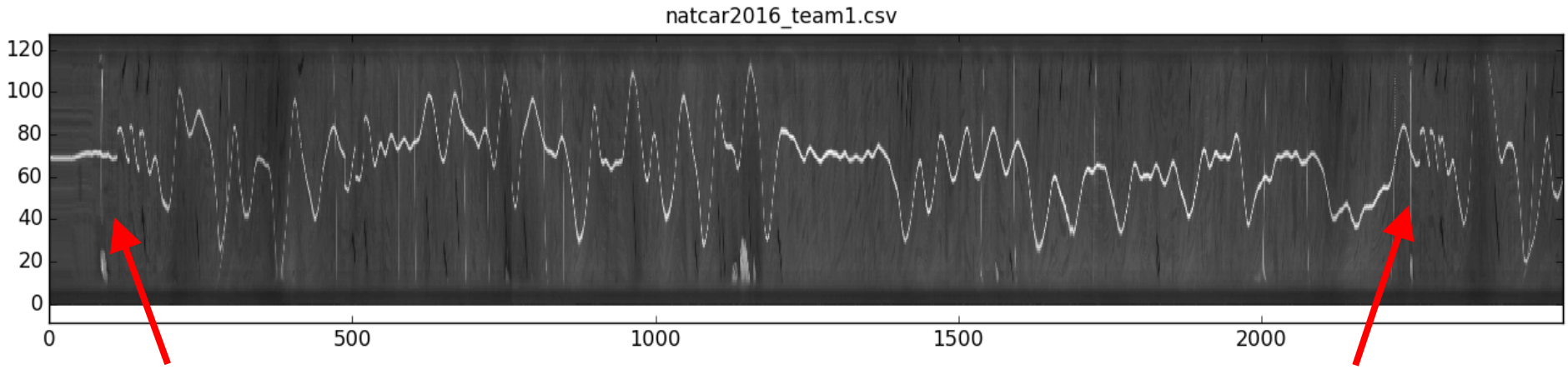
1. Check off-
    1. 3/9/2018: C6 Benchtop line tracking
    2. 3/16/2018 C7 everything on car, drop-and-run, (velocity control), open loop fig 8
    3. 3/23/2018 C8: Closed loop figure 8 drop and run
2. Quiz 4 line sensor 3/13
3. Community Spirit: PCB peer review, Piazza, helping fellow students
4. CalDay Sat. April 21 @ UCB,
5. Line sensor processing HW1 due 3/13 – upload Python code to bcourses. (Will run on other data.)
6. Waterfall plot for line data
7. Lab safety/hygiene

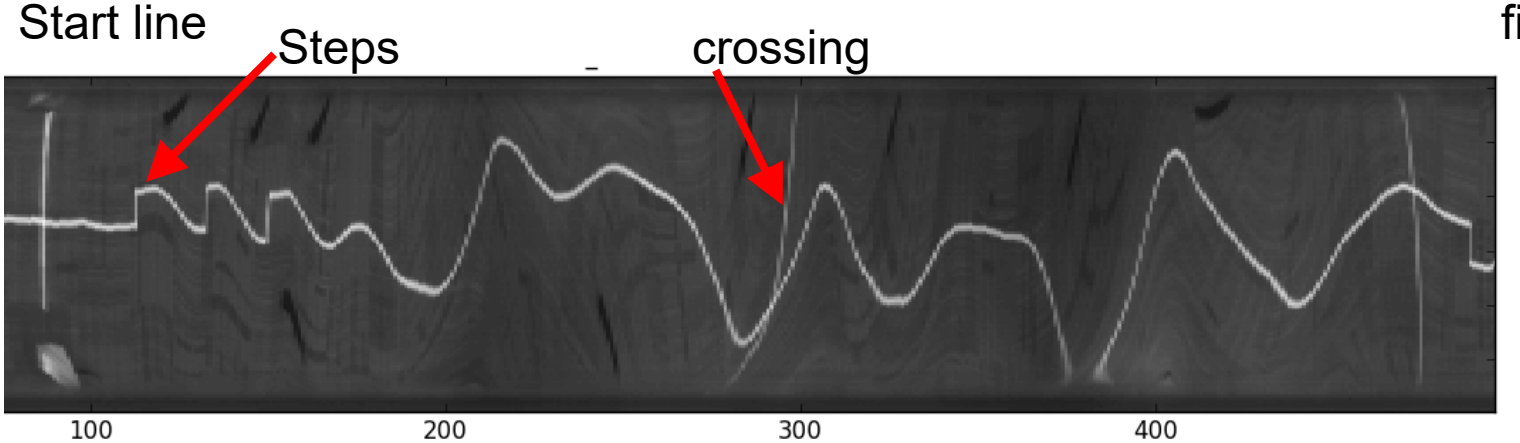
Topics

- Line sensor conclusion
- Software: non blocking printf, timing, debugger
- Speed sensing/velocity control
- Velocity control detail
- Feedback control overview
- Bicycle steering model

Python Template

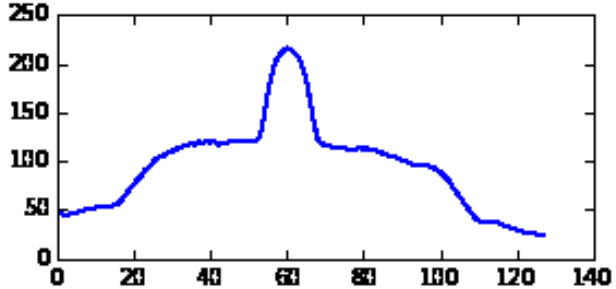
track_center_list - A length n array of integers from 0 to 127.
Represents the predicted center of the line in each frame.
track_found_list - A length n array of booleans.
Represents whether or not each frame contains a detected line.
cross_found_list - A length n array of booleans.
Represents whether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



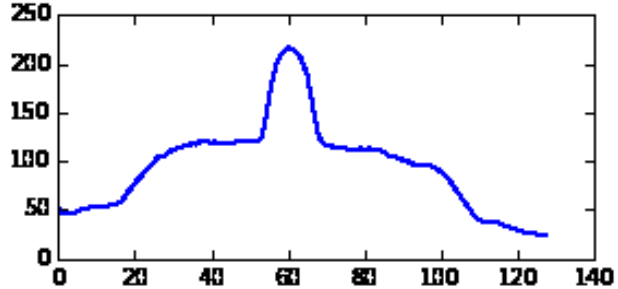


TSL 1401 line sensor NATCAR 8 bit

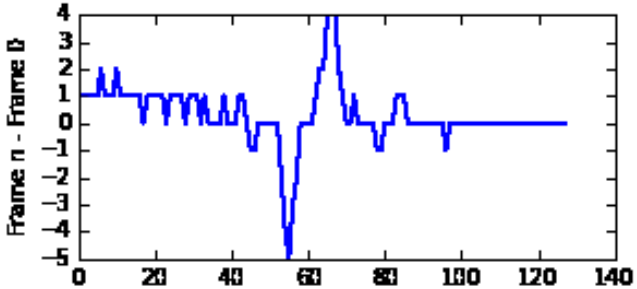
Frame 0



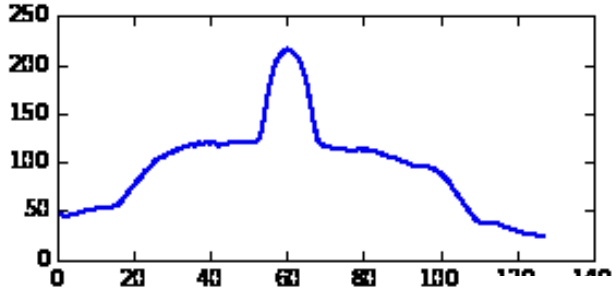
Frame 1



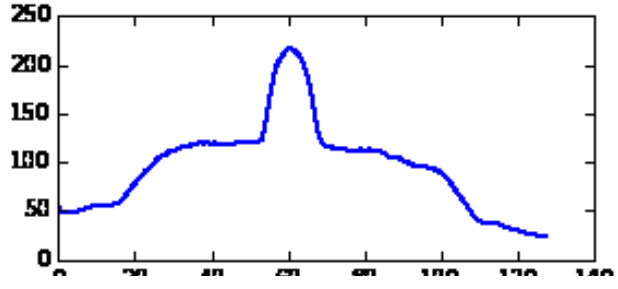
Frame 1-Frame 0



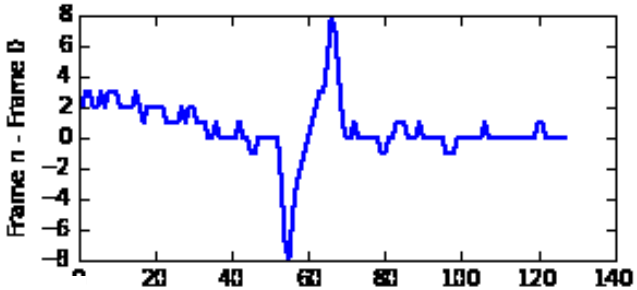
Frame 0

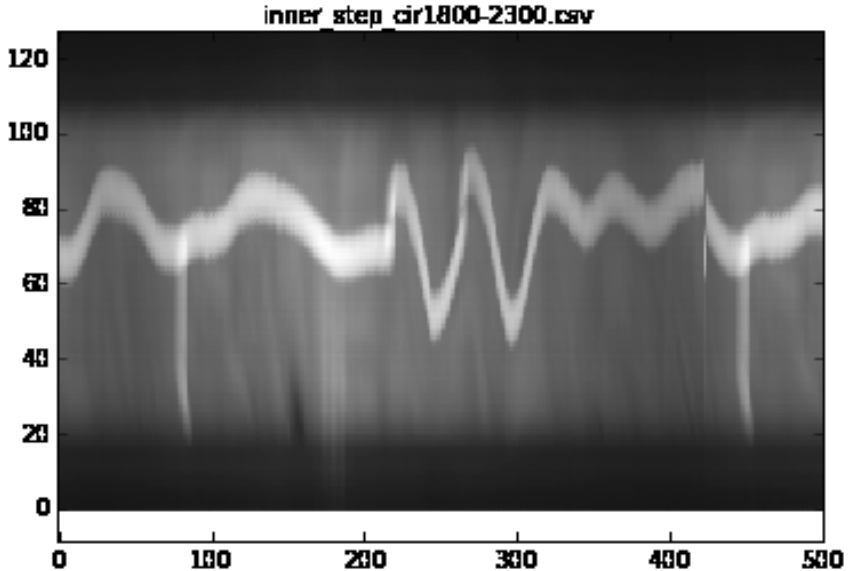


Frame 2



Frame 2-Frame 0



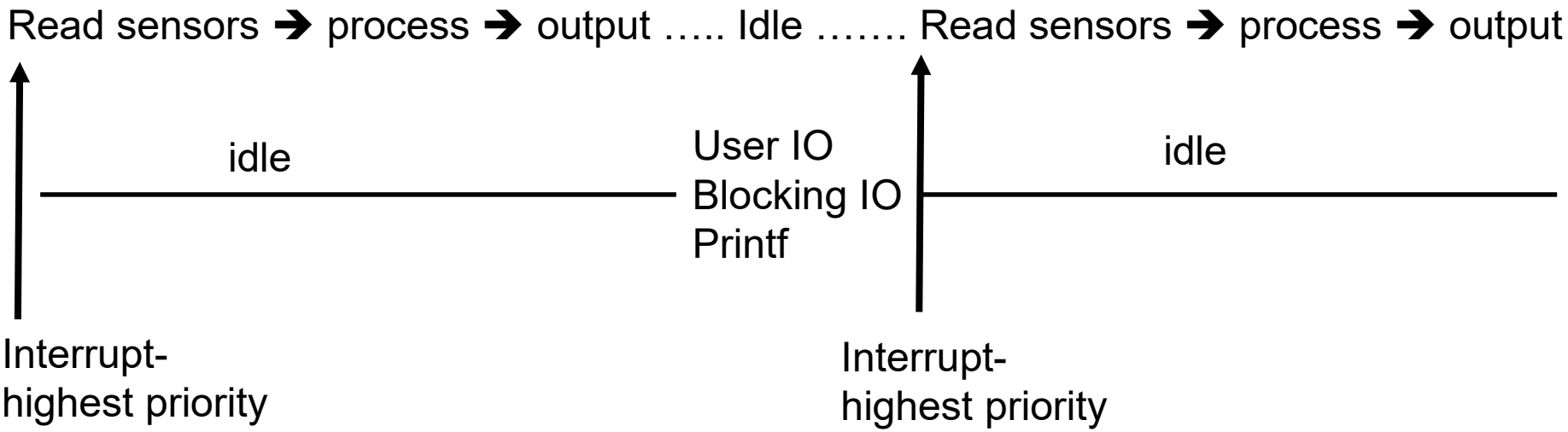


Possible algorithms for line detection

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



Non-blocking print

https://github.com/ucb-ee192/SkeletonMCUX/tree/master/frdmk64f_skeleton

```
sprintf(log,"Idle. sum of cos = %d \n\r",
                                (long)ZSum);
log_add(log);

void log_add(char *log)
{ xQueueSend(log_queue, log, 0);
     // send data to back of queue,
    // non-blocking, wait=0 ==> return
immediately if the queue is already full.

}
static void log_task(void *pvParameters)
{ …
xQueueReceive(log_queue, log, portMAX_DELAY);
```

Timing from MCUXpresso Tools

- Global Systime (every 100 us)
- Periodic Interrupt Timer PIT-CVAL0 (count down)
- internal module clock frequency (60 MHz)

Step over

MCUXpressoProj - Develop - frdmk64f_skeletonrtos/source/task1.c - MCUXpresso IDE

File Edit Source Refactor Navigate Search Project Run FreeRTOS Window Help

Quick Access

Qui... Glo... Var... Bre... Out...

Variable	Type	Value
log_queue	QueueHandle_t	0x200011c8
sqrt_array	float [1000]	0x20000094
systime	volatile uint32_t	30076
ulIdleCycleCount	uint32_t	95938
xTickCount	volatile TickType_t	3000
ZSum	double	95789.89952

Debug

write_task_10 at task1.c:75 0x15ea

startup_mk6... main.c idletask.c task1.c

```
70      tick_start = xTaskGetTickCount();
71      for (i = 0; i < 9; i++)
72      {
73          systime_start = systime;
74          for (j=0; j < 1000; j++)
75              sqrt_array[j]= sqrtf((float)j);  // sqrt ~ 100 u
76          sprintf(log, "Tsk1#%d, 1000 sqrt time (us) %d\n\r",
77                  (int)i, 100*(long)(systime-systime_start));
```

Monitors

PIT [MK64FN1M0x

PIT: 0x40037000 [MK64FN1M0xxx12] New Renderings...

Register	Address	Value
PIT-CVAL0	0x40037104	0x16d3
TVL	[31:0]	0x16d3
PIT-CVAL1	0x40037114	0x0
PIT-CVAL2	0x40037124	0x0
PIT-CVAL3	0x40037134	0x0

NXP MK64FN1M0xxx12 (frdmk64f...etonrtos)

Timing from MCUXpresso Tools

~38 clocks for sqrt (hardware floating point) ~0.6 us
~1689 clocks for cos() ~28 us

MCUXpressoProj - Develop - frdmk64f_skeletonrtos/source/task1.c - MCUXpresso IDE

File Edit Source Refactor Navigate Search Project Run FreeRTOS Window Help

Quick Access

Qui... Glo... Var... Bre... Out...

Name	Type	Value
pvParameters	void *	<optimized out>
log	char [50]	0x20002374 <uch
tick_start	TickType_t	0
tick_end	TickType_t	<optimized out>
systime_start	uint32_t	30075
i	int	3
j	int	6

```
Name : log
    Details:"Tsk1#2, 1000 sqrt time (us) 
    Default:0x20002374 <ucHeap+4536>
    Decimal:536879988
    Hex:0x20002374
    Binary:1000000000000000100011011110100
```

Debug

write_task_10 at task1.c:74 0x15ee

startup_mk6... main.c idletask.c task1.c

```
70      tick_start = xTaskGetTickCount();
71      for (i = 0; i < 9; i++)
72      {
73          systime_start = systime;
74          for (j=0; j < 1000; j++)
75              sqrt_array[j]= sqrtf((float)j);  // sqrt ~ 100 u
76          sprintf(log, "Tsk1#%d, 1000 sqrt time (us) %d\n\r",
77              (int)i, 100*(long)(systime-systime_start));
```

Monitors

PIT [MK64FN1M0x

PIT: 0x40037000 [MK64FN1M0xxx12] New Renderings...

Register	Address	Value
PIT-CVAL0	0x40037104	0x16d0
TVL	[31:0]	0x16d0
PIT-CVAL1	0x40037114	0x0
PIT-CVAL2	0x40037124	0x0
PIT-CVAL3	0x40037134	0x0

NXP MK64FN1M0xxx12 (frdmk64f...etonrtos)

Timing from MCUXpresso Tools

```
tick_start = xTaskGetTickCount();
```

```
/* Set timer period for channel 0 */
PIT_SetTimerPeriod(PIT, kPIT_Chnl_0, USEC_TO_COUNT(100U, PIT_SOURCE_CLOCK));
// 100 us timing
   /* Enable timer interrupts for channel 0 */
   PIT_EnableInterrupts(PIT, kPIT_Chnl_0, kPIT_TimerInterruptEnable);
   /* Enable at the NVIC */
   EnableIRQ(PIT_IRQ_ID);
```

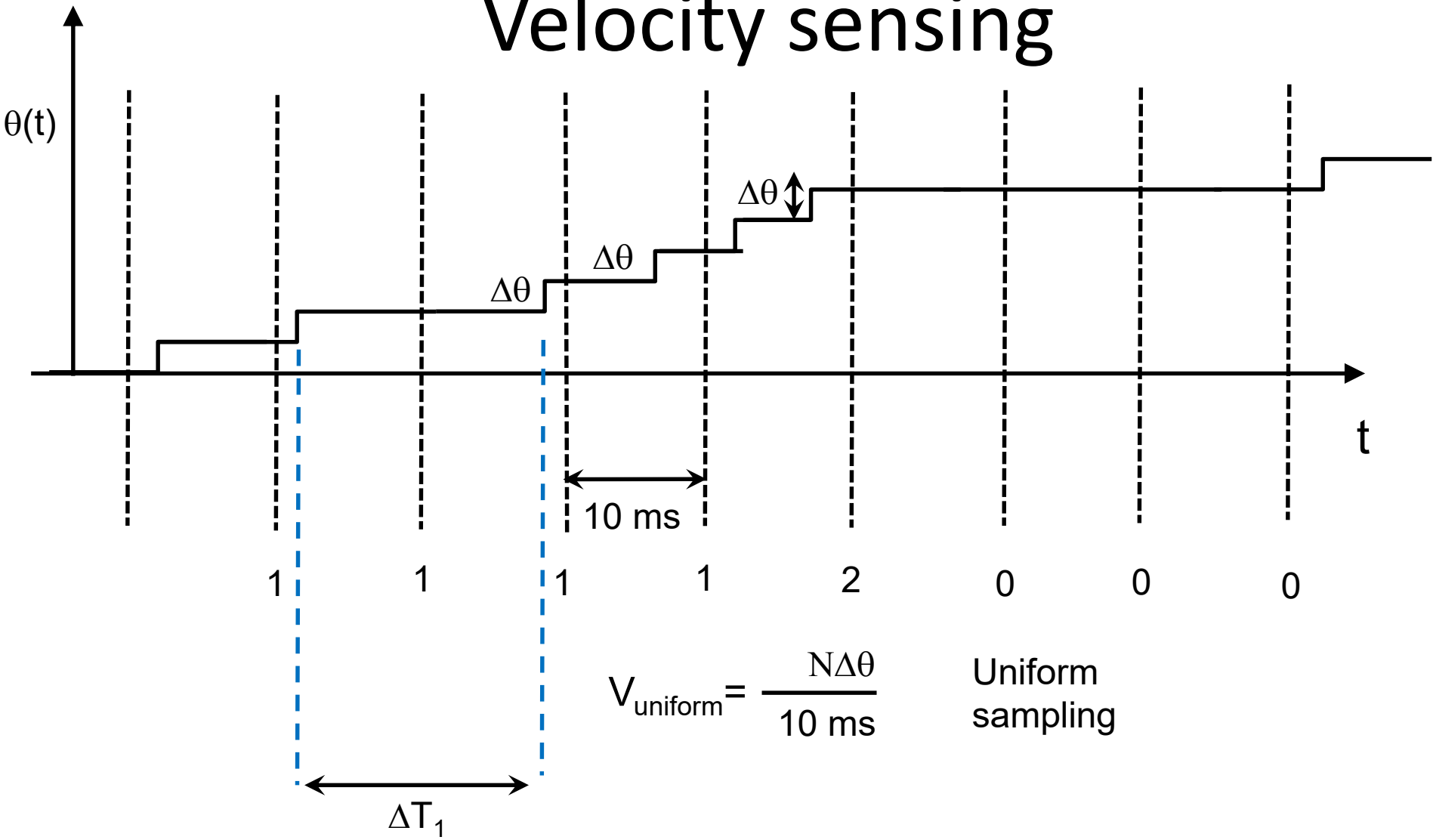
```
void PIT0_IRQHandler(void) \\ 100 us
{ systime++; /* hopefully atomic operation */
    PIT_ClearStatusFlags(PIT, kPIT_Chnl_0,
kPIT_TimerFlag);
/* Clear interrupt flag.*/
    pitIsrFlag = true;
}
```

Velocity sensing (recap)

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

On board…

Velocity sensing



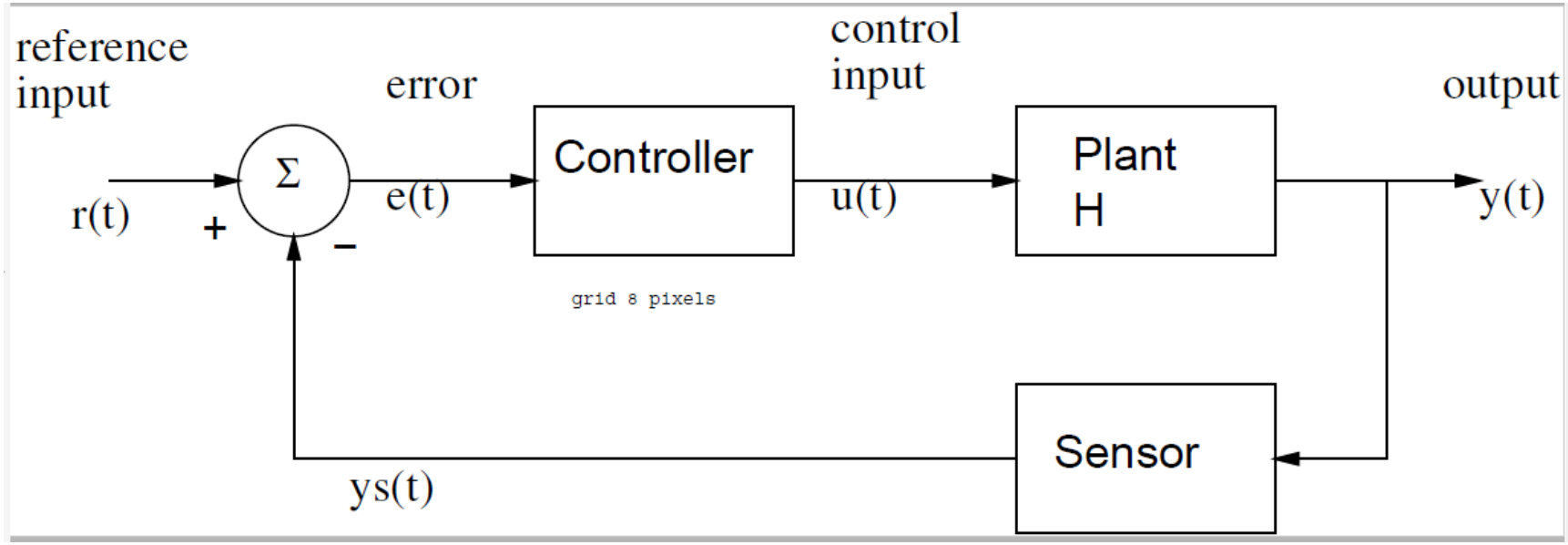
$$V_{uniform} = \frac{N\Delta\theta}{10\ \text{ms}}$$

Uniform sampling

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

Edge timing

Velocity control overview



On board…

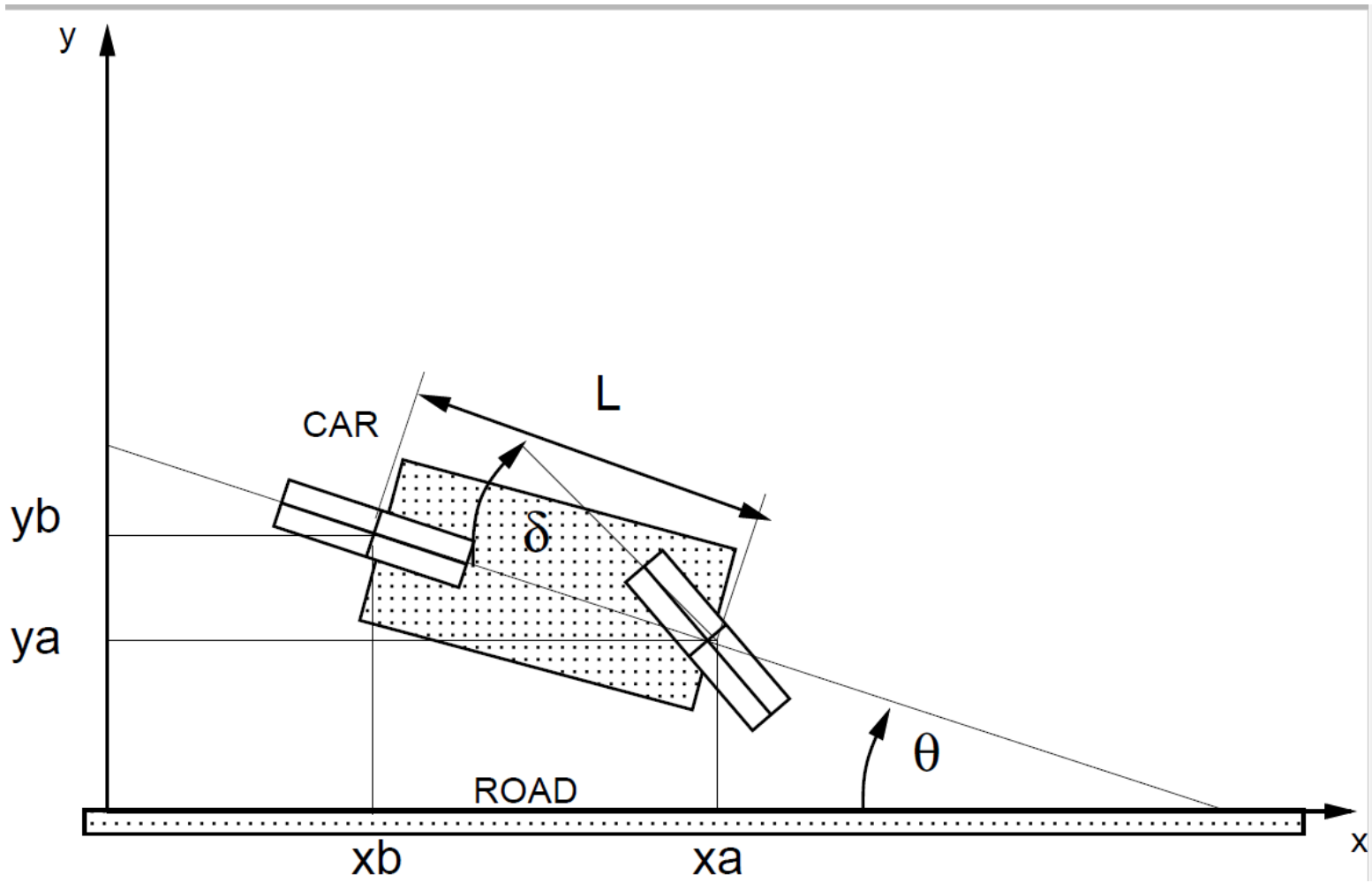
Proportional control:

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

Proportional + integral control

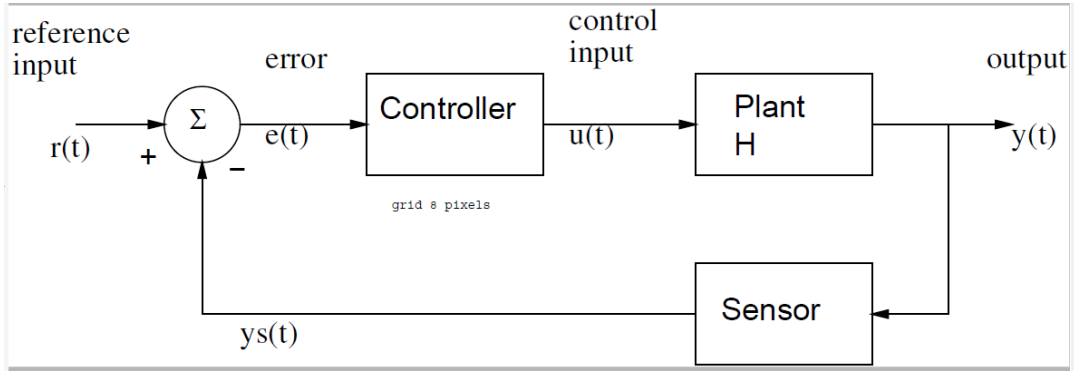
```
U = kp*e + ki * e_sum;
e_sum = e_sum + e;
```

Bicycle Steering Model

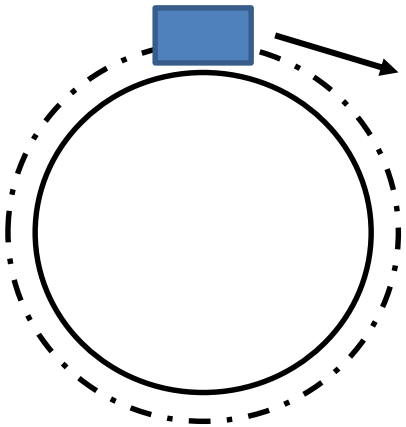


Equations On board

Bicycle Steering Control



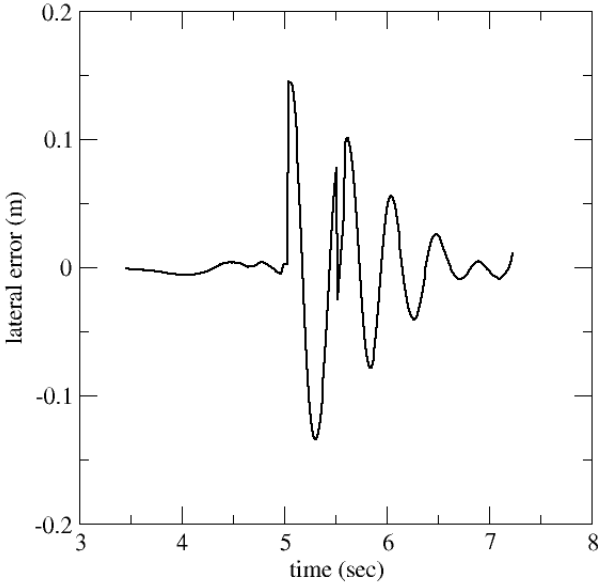
Note steady state error: car follows larger radius



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

Proportional+derivative

P+I+D



On board