Weekly Goals #6
due Wednesday, 11/16/22 11:59pm

The main objective of this set of goals is to make the gimbals move smoothly, especially between movements, and visualize the data.

I would like complete this programming by Wednesday 11/16, so we have a full week to play with cameras before Thanksgiving.

Problem 1 (Plotting):

Before we start complex movements, it always helps to plot the data. Please update the code to

(a) Initialize some arrays:

```
LENGTH = 1000
TIME   = np.zeros(LENGTH)
PAN    = np.zeros(LENGTH)
TILT   = np.zeros(LENGTH)
index  = 0
```

(b) Inside the loop, store the relevant data:

```
if index < LENGTH:
    TIME[index] = t
    PAN[index]  = pan
    TILT[index] = tilt
    index += 1
else:
    # Do we want to break out of the loop?
    break
```

(c) After the loop, plot.

```
plt.plot(TIME[0:index], PAN[0:index] , color='blue', linestyle='-',  label='Pan')
plt.plot(TIME[0:index], TILT[0:index], color='red',  linestyle='-',  label='Tilt')

plt.xlabel('Time␣(s)')
plt.ylabel('Angle␣(rad)')
plt.title('Robot␣Data')
plt.grid()
plt.legend()
plt.show()
```

Please show us a screen shot? Does this look as you expect?

Problem 2 (Plotting Actual vs. Desired):

Please update the code to show both the actual and commanded angles. For example, the actual as a solid line, the desired as a dashed line.

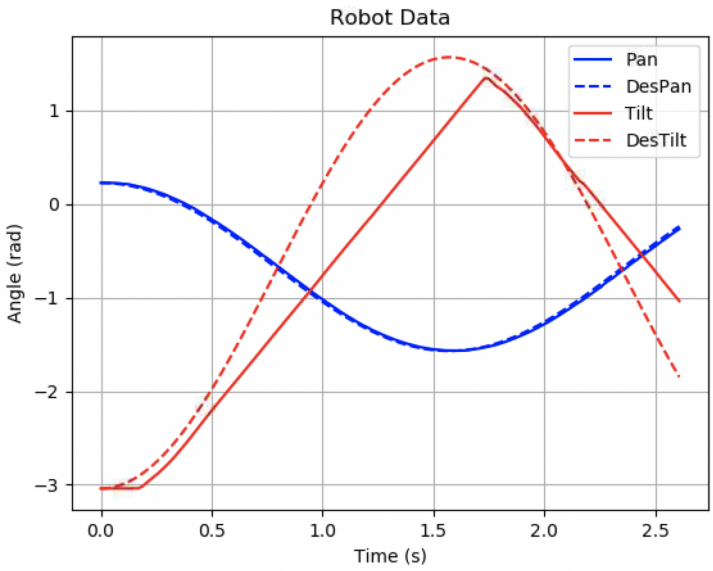
Can you see the lag between the command and actual? About how long is the lag? How does that compare with the update time step of 10ms?

Problem 3 (Faster!):

Try increasing the speeds of your motion. That is, change `np.cos(t)` to `np.cos(2*np.pi*t/T)` and change the period T.

How fast can you get before things don't track well any more? Check this using the plots.

I am hoping for something like



where you can see my pan motor is doing well, but my tilt motor is not keeping up.

Return the speed to something that does not saturate thee motors or cause problems.

Problem 4 (Changing Movement - First Spline):

Which brings us to changing the movement. As a first step, please change the code to simply start at the initial positions and move "home", meaning to zero.

We'll do this with a spline for each axis:

$$\text{despan}(t) = a_{\text{pan}} + b_{\text{pan}}(t - t_0) + c_{\text{pan}}(t - t_0)^2 + d_{\text{pan}}(t - t_0)^3$$

$$\text{destilt}(t) = a_{\text{tilt}} + b_{\text{tilt}}(t - t_0) + c_{\text{tilt}}(t - t_0)^2 + d_{\text{tilt}}(t - t_0)^3$$

where you can determine the parameters

$$a = p_0$$
$$b = v_0$$
$$c = 3\frac{p_f - p_0}{T^2} - \frac{v_f}{T} - 2\frac{v_o}{T}$$
$$d = -2\frac{p_f - p_0}{T^3} + \frac{v_f}{T^2} + \frac{v_o}{T^2}$$

from the initial and final conditions and time for the motion. That means you should have 9 variables in your code (8 spline parameters and the time T) which should be initialized.

For now, assume the initial and final velocity is zero, the final position is zero, and the motion should stop (breaking out of the loop) when the time is over.

$$a = p_0$$
$$b = 0$$
$$c = -3\frac{p_0}{T^2}$$
$$d = 2\frac{p_0}{T^3}$$

which means

$$p(t) = p_0(1 - 3\frac{(t - t_0)^2}{T^2} + 2\frac{(t - t_0)^3}{T^3}$$

To test, move the gimbal somewhere, e.g. by hand. Then run the code. What is a reasonable time, for a "worst-case" far away starting position?

Problem 5 (Second Scanning Movement):

New we'll add a second movement. In particular, I would like to add a sinusoidal "scanning" movement to the end of the previous "go-home" movement.

Use

$$\text{despan}(t) = \text{offset}_{\text{pan}} + \text{amplitude}_{\text{pan}} \sin(2\pi \frac{t - t_0}{T_{\text{pan}}})$$

$$\text{destilt}(t) = \text{offset}_{\text{tilt}} + \text{amplitude}_{\text{tilt}} \sin(2\pi \frac{t - t_0}{T_{\text{tilt}}})$$

Starting from zero, the offsets will be zero. For amplitude, use $\pi/3$ for pan and $\pi/4$ for tilt. For periods, pick something you find reasonable, but ideally make pan twice as fast as tilt, so it moves left/right quickly and up/down slowly. Basically scanning.

For code, the means

(a) added the additional variables,

(b) in the main loop, when the first time is complete, reset the start time and parameters for the sinusoid

(c) introducing a `mode` variable to distinguish which phase you are in.

So in the code, when calculating the command, you'll have

```
if mode == 'Spline':
    t1 = t - t0
    despan  = apan  + bpan*t1 + ...
    destilt = atilt + ...
else:
    t1 = t - t0
    despan  = offpan  + amppan * np.sin(2*np.pi * t1/Tpan)
    destilt = offtilt + ...
```

In the event phase you'll have:

```
if mode == 'Spline':
    # Check whether to transition
    t1 = t - t0
    if (t1 > T):
        # Reset mode and parameters.
        t0 = t
        mode = 'Scan'
        ....
else:
    # This should never end... so do nothing.
    pass
```

Yes, you will notice the the scan starts very abruptly, non-smoothly? Does that show in the plots?

Problem 6 (Smooth Transition):

To avoid the sudden velocity change, we need to update the initial spline parameters to end the first motion at a non-zero velocity.

Does that fix the problem? Please show the plots which should now be smooth!

Problem 7 (Keyboard Event):

Now, to add some features, let's watch the keyboard. You will need to import

```
import select
import sys
```

Then, in the event handling portion of the code, when in the 'Scan' mode add

```
if sys.stdin in select.select([sys.stdin], [], [], 0)[0]:
    # The user has hit return!  Grab their input.
    usertext = input('')
```

When the keyboard is hit, I would like to transition back to the home position and stop the code. That means, reseting the spline parameters and start time, and changing modes. A few challenges

(a) You will want to know the initial velocity for the spline segment to calculate the parameters. That means, in the main loop, in addition to calculating `despan` and verb—destilt—, you should also calculate `despanvel` and —destiltvel—, their respective velocities. That means coding up the analytic derivative.

(b) Then you will know the initial position and velocity from the last computed commands, and the final position and velocity as 0 and 0.

(c) if you switch mode back to "Spline", will this cause the code to then re-transition to scanning again? How do you prevent that?

This means we

- start with a spline with a non-zero final velocity,
- transition to scanning after the spline completes,
- transition to a spline with non-zero start velocity when the keyboard is hit,
- end the code when the last spline finishes.

All this should be smooth - please check with a data plot.

Problem 8 (Challenge Problem):

If you are feeling so inclined, you might add the following options.

(a) If the user enters

- '0' go to the zero position,
- 'A' go to a position (pan,tilt) = $(-45°, +30°)$
- 'B' go to a position (pan,tilt) = $(+30°, 0°)$

(b) Wait at those points for 2 seconds, then go home and restart the scan.

The challenge here will be how best to store all the information. In particular, a single `mode` variable is perhaps not the best?

We'll let you be creative, though please ask for help/guidance if you want other ideas?