Nelly

This doc is only intended for quick notes. Official stuff should be stored in the ADD, SWDD and elsewhere.

----------------------------------------------------------------------------------------------------------------------------------------------------

05.18.2018

**Starting with Tracking**

We are going to piecemeal this algo, beginning with tracking. Reason for starting here:

* Image processing will be harder, but we already know it is possible.
* We will be encouraging strong code reuse for image processing and we need to still get what we need (libraries, mods, etc…)
* Tracking will be our bread and butter ahhahah.

Things to read up on for tracking:

* Digital Image Correlation (DIC)
* Direct Linear Transformation

Email from max (well actually Noah)

Max,

Digital Image Correlation (DIC) is the technique we use that I was explaining to you on the boat ([Wikipedia](https://en.wikipedia.org/wiki/Digital_image_correlation_and_tracking), [Industry](https://www.dantecdynamics.com/measurement-principles-of-dic)). A full DIC analysis goes way more in-depth than you would have to for identifying just the location of objects of interest, but the first step in 3-D DIC is centroid tracking of the displacement markers using two cameras, and then running a Direct Linear Transformation (DLT) algorithm ([Wikipedia](https://en.wikipedia.org/wiki/Direct_linear_transformation), [Better Explanation](https://me363.byu.edu/sites/me363.byu.edu/files/userfiles/5/DLTNotes.pdf)). I think a DLT algorithm would be pretty easy to implement after an image processing algorithm identified the centroid of objects of interest, and the DLT algorithm shouldn’t be too computationally expensive.

I was also just playing around with Google Street View, and that there is no distortion at the center of the field of view, no matter what view you are looking at. To me, that implies that we would not have to worry about lens distortion as long as all images used for the DLT had the object of interest centered in the frame.

-Noah

My first-thoughts:

DIC - would have a tough time applying here because we are dealing with positional changes, not deformations. In theory, this could be helpful but it is a very verbose approach. I still think something like Auction, or Greedy, would be the best approach since we are simply solving an assignment problem from points, not a deformation of an object.

DLT - Complicated. Abstract. Should revisit if we get stuck later but will should start simpler. Also, there’s no way we can force our objects in the field of view.

Google streetview api

There are two different python modules for the same API:

1. Google\_streetview.api
2. Streetview

We will use 1 for now.

Python module documentation:

<https://pypi.org/project/google-streetview/#description>

<https://rrwen.github.io/google_streetview/>

----------------------------------------------------------------------------------------------------------------------------------------------------

5.18.2018

Info on the difference between geodetic and geocentric. Which one is the GOOGLE API using ?

<http://agamenon.tsc.uah.es/Asignaturas/it/rd/apuntes/RxControl_Manual.pdf>

Uploaded a small class for coordinate transformations between the Geodetic lat/lon/el and ECEF. We now have two transformation routines and can go back and forth. The original plan was to move into tracking, but I think it makes more sense to first look at the triangulation routine.

First start in 2D, this will be simple since any two lines (unless parallel) intersect. We will then go to 3D, which will be a little more difficult. We will need to write an algorithm for the point at which they are closest together in ECEF space.

The triangulation 3D routine should have 4 outputs for every triangulation: X Y Z (of closest point) and distance the two lines are off.

Inputs: camera locations in ECEF (both locations)

Both look vectors in ECEF

Distance between two lines algorithm:

<http://pages.pacificcoast.net/~cazelais/251/distance.pdf>