PART 4
On board orientation software

Candidates: 76408 & 83886
(Dated: December 17, 2020)

We have constructed software that will enable our rocketshuttle to navigate interplanetary space on our voyage to our destination, planet 6. The software we've created has the ability to determine our shuttles orbital orientation, i.e what direction it is facing in the xy-plane, as well as both position and velocity relative to the star at the center of the solar system. This software is then tested by our friends at the ast2000 research center, and the software passed all their tests.

I. INTRODUCTION

For this study we are going to solve some issues that our shuttle will face in the aftermath of the launch, when we are currently travelling on the interplanetary highway. During the voyage our shuttle must be able to orient itself, in order to determine if it is still on the right course, or if we need to perform adjustments.
More specifically, in this study we are going to create onboard software that can do the following

1. Find our spaceshuttles rotational orientation using our onboard camera.
2. Find our spaceshuttles position in the solarsystem by triangulation.
3. Find our spaceshuttles speed and the direction we are travelling.

As we will explain in further detail in the method section, we will need to write software that can accurately compute our rotational orientation by images produced by our onboard camera compared to a spherical reference image of the night sky. It must also be able to triangulate the shuttles position using measuring tools that returns the distance from the bodies in the system to the shuttle, as well as using dopplershift measured from two reference stars in distant galaxies to determine the shuttles velocity.

II. THEORY

For relevant theory, see section **RELEVANT MATHEMATICS** in AST 2000 - Part 4: Onboard Orientation Software

III. METHOD

In order to manage to orientate in the the vast universe surroundings, we will be needing to generate reference pictures. We will start by generating a flat picture from a part of the spherical picture, because the standard camera which we are using are not able to capture the entire spherical surface.

Naturally, our camera can only capture things that are within its field of view, and because of this we can introduce two parameters, angular width, α_ϕ and angular height, α_θ

It will be easiest to represent the field in spherical coordinates, seeing as the reference picture is of a spherical surface, and we would therefore have two maximum angular widths corresponding to θ and ϕ variables as illustrated in figure (1)

$$\alpha_\theta = \theta_{max} - \theta_{min}$$

$$\alpha_\phi = \phi_{max} - \phi_{min}$$

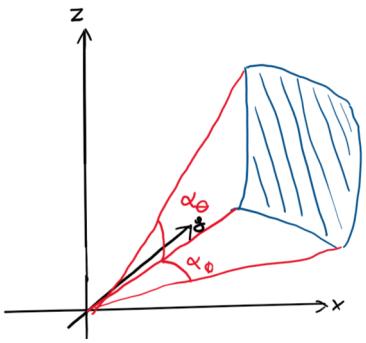

Figure 1. visualisation of α_θ and α_ϕ

where ϕ is the azimuthal angle, and θ the zenith angle.

In order to manage to take a flat picture from a spherical field of view, we will be using steographic projection method (see references for detailed explanation). This method maps the a part of the spherical surface onto a flat surface like a projection with straight lines, as figure 2 illustrates.

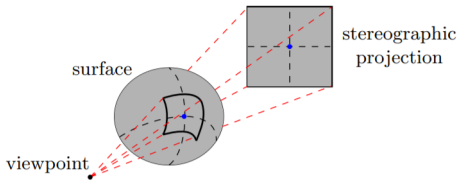

Figure 2. Visualisation of stereographic projection

From the theory behind stereographic projections we find that with limitations on θ and ϕ, the stereographic projection introduces equivalent limitations on X and Y by the given expressions

$$X_{max/min} = \pm \frac{2\sin(\alpha_\phi/2)}{1+\cos(\alpha_\phi)/2} \tag{1}$$

$$Y_{max/min} = \pm \frac{2\sin(\alpha_\theta/2)}{1+\cos(\alpha_\theta)/2} \tag{2}$$

Using this, together with the given field of view for the onboard camera, we can go from a flat picture with x,y coordinates to a spherical image with θ, ϕ. This is essentially doing the projection backwards, and we can use this to define a (X,Y) grid of the arbitrary flat picture of the sky we've taken by using equations (1) and (2).
This gives us the max, and min values in our pixel grid. Therefore we can create our X,Y grid by counting the number of pixels in the picture along the two axes. For example, say we have a 200x200 picture with max 199 and min 0 for both axes, we can create our XY grid by x, y = [0, 1, 2, ..., 199]. This means that we create the values for x and y by creating a list that ranges from the min.value to the max.value with N number of points corresponding to the counted number of pixels in the picture.

After we have found the grid in (X,Y) coordinates, we can convert it into (θ, ϕ) coordinates. In order to do this, we follow the stereographic projection documentations which states

$$\theta = \theta_0 - \arcsin\left[\cos\beta\cos\theta_0 + \frac{Y}{\rho}\sin\beta\sin\theta_0\right] \tag{3}$$

$$\phi = \phi_0 - \arcsin\left[\frac{X\sin\beta}{\rho\sin\theta_0\cos\beta - Y\cos\theta_0\sin\beta}\right] \tag{4}$$

Where θ_0 and ϕ_0 are the angles the camera was centered about when it shot the picture.

As mentioned in the introduction our great research team have already mapped the entire night sky, and produced a spherical image by doing a similar procedure as we just explained. This means that for each pixel in this nightsky spherical grid, there exist a color code RGB (in the interval [0,255] for R,G,B). So if we take a picture of the nightsky, and perform the stereographic projection backwards, each of the pixels will have a corresponding RBG value and this particular composition of RGB valeus arranged in our grid can only be found one place in the nightsky sphere.

Now we want to use this to find a way to decide our rotational orientation. We have previously assumed that our entire voyage will happen in the xy-plane (meaning $\theta_0 = \frac{\pi}{2}$) so we need only decide the angle ϕ, furthermore we are going to assume that the pictures we are taking will always be an integer angle, and so we can create reference pictures according to each of the ϕ in [0, 1, 2, ..., 359]° using the nightsky grid our research team made for us. This gives us 360 different compositions of RGB-grids corresponding to taking pictures with different angles ϕ.

We can now take a picture centered around an arbitrary angle ϕ, and then compare this image to the 360 reference pictures, and since we know what direction the ref.pic was facing, we can figure out what direction our camera must be facing. Say our pictures RGB values matches ref.pic number 79 - then we know that the camera shooting the picture must be centered about $\phi = 79°$, and this is our rotational orientation!

Now that we know the direction we are facing, we want to figure out our velocity. To do so we will use two distant stars and look at their electromagnetic radiation (light) and measure the Dopplershift $\Delta\lambda$ in the H_α spectralline.
We can illustrate this situation in figure 3.

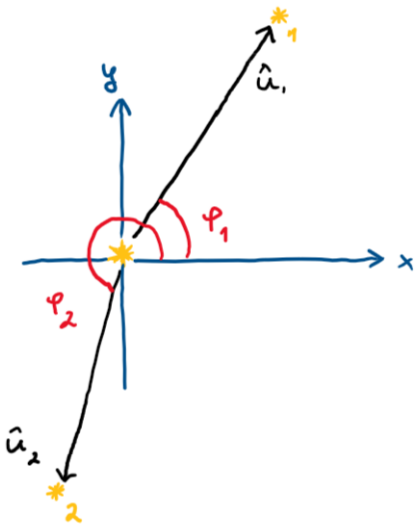

Figure 3. Illustration of our star in the origin and two other reference stars

Here we have defined (x,y) coordinate system where our home star is in the origin. In addition, we have have included two reference stars that our friends from ast2000 research center has informed us about. We can therefore get information about the $(\phi_1,\ \phi_2)$ coordinates of those suns if we do a change of coord.sys to define the new unit vectors $\hat{u}_1, \hat{u}_2$ by the transformation

$$\hat{u}_1 = (\cos\phi_1, \sin\phi_1)$$
$$\hat{u}_2 = (\cos\phi_2, \sin\phi_2)$$

As seen in 3 the two reference stars now align perfectly with our new coordinate axes.

We can use this to find the radial velocity of our star by the formula for dopplershift

$$\frac{\Delta\lambda}{\lambda_0} = \frac{v_r}{c}$$

$$v_r = c\frac{\Delta\lambda}{\lambda_0} \tag{5}$$

where λ_0 is the wavelength seen from the rest frame of the object emitting the light wave, c the speed of light and v_r is the radial velocity resulting in the shift.

This will give us a measure of our star radial velocity relative to each of the two stars. Since we are measuring from two distant star we can safely assume that both our spaceshuttle and our sun lies approximately along the same axis as seen from the distant star. (the angle between them is negligible).

This means that we have a reference frame with velocity equal to the suns radial velocity along each of the new unit vectors u_1, u_2 respectively (we will call this for $v_{ref,i}$ where the index tells us witch stars we are referring).

With this we can also calculate the radial velocity of our spaceshuttle doing a new measurement of dopplershift using (5) to find the radial velocity of the shuttle relative to the stars in the ref.frame with velocity $v_{ref,i}$.

If we now want to find the velocity of the spaceshuttle relative to the sun in our solarsystem (we are free to change reference systems), in a reference frame where the sun is at rest, and the distant stars are moving, we will need to add these velocities together (this is a standard galilaen transformation)

$$v_i = v_{ref,i} + \frac{\Delta\lambda}{\lambda_0}c \tag{6}$$

for v_1, v_2, which will be the spaceshuttles velocity along the ϕ_1 and ϕ_2 axis respectively $(\phi_1,\ \phi_2) = (\hat{u}_1,\ \hat{u}_2)$ if we look at the illustration in figure (3).

However, we want to transform from the $(\phi_1,\ \phi_2)$ coordinate system, to (x,y) coordinate system. As we know from linear algebra we can return to our original coordinate system by doing the inverse operation. Since our original change of coordinate system was done by the matrix

$$\begin{pmatrix}\hat{u}_1\\ \hat{u}_2\end{pmatrix} = \begin{pmatrix}\cos\phi_1 & \sin\phi_1\\ \cos\phi_2 & \sin\phi_2\end{pmatrix}$$

the reverse operation will be done by the inverse of this square matrix. Thus we can find

$$\begin{bmatrix}v_x\\ v_y\end{bmatrix} = \frac{1}{\sin(\phi_1-\phi_2)}\begin{bmatrix}\sin\phi_2 & -\sin\phi_1\\ -\cos\phi_2 & \cos\phi_1\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix} \tag{7}$$

However, we need to know not only the orientation and velocity, we also need to know another crucial aspect of the space travel, the position of the spacecraft itself. In order to know the position of our spacecraft, we will be using the trilateration method. This method is widely used by the GPS or GPSS (Global positioning system, global positioning satelite system). This method is represented in figure 4

Here we can see three bodies with position P1, P2 and P3. If we know how far we are from each body, we can then draw circle with radius of this known distance. We call those distances for r1, r2 and r3. If we would for example have only two bodies, let's say P1 and P2, we would conclude that we could be either in position A or B, because these points only verifie the distances from you and the bodies. However, if we include the third body P3, we can see that the only position you can be at and verify the distances is at point B.

We use the same principle in space as well, we only choose these bodies to be our sun, and two other planets.
But first, we have to make an assumption about our spacecraft equipment made by our great engineers in our home planet. By using the built in radar, our spacecraft

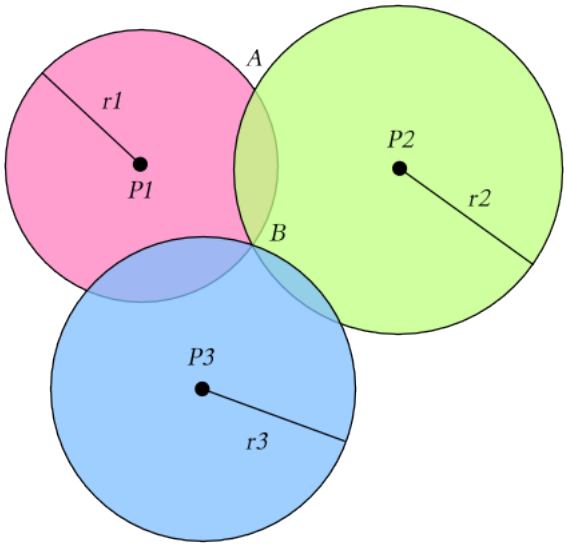

Figure 4. representation of trilateration method

is capable of measuring the distance between itself and another object (in this case those two planets and our sun)

We will always know how far we are from the sun and those two planets. We could make our equipment to search for the position with distance r_s from the sun around it with 2π degrees, and find the right angle where all these three distances are valid. Illustration in figure 5 illustrate the situation

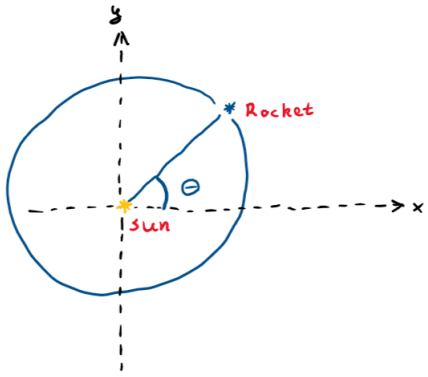

Figure 5. Illustration our spaceships position by the sun with radius r and angle θ

IV. RESULTS

We check the software we created to decide rotational orientation against 5 sample images that we received from our research team, and of which we are already aware of the angle the picture was centered about.
See table for values after software test

Table I. Software test

Image	**Angle**
sample0000.png	0°
sample0200.png	20°
sample0911.png	91°
sample1400.png	140°
sample1900.png	190°

We can clearly see from table I that our software is indeed working as intended.

As mentioned in the introduction, and explained in the method section, we will attempt to determine our spaceshuttle velocity by measuring dopplershift from two distant stars. We showed that we will need some values to do so, and our research team has supplied us with what we need, the angles to define our new unit vectors and also the dopplershift in the H_α-line measured at our suns ref.frame from each of the two reference stars.

ϕ_1	216
ϕ_2	117
$\Delta\lambda_1$	0.0127490
$\Delta\lambda_2$	0.0040528
λ_0	656.3

Table II. Information about ref.stars

where the wavelength are measured in [nm], and the angles are defined as azimutal from x-axis in degrees.

Following the formula we derived in the method section, we can just put in these 4 values and find ref.velocities in $[\frac{AU}{yr}]$

$$v_{ref,1} = c\frac{\Delta\lambda_1}{\lambda_0} = 1.22847$$
$$v_{ref,2} = c\frac{\Delta\lambda_2}{\lambda_0} = 0.39052$$

where this will be the speed the two stars travel along the new unit vectors $\hat{u}_1, \hat{u}_2$. Lets see what happens if we assume our shuttles instruments measure a dopplershift of $\Delta\lambda_1 = \Delta\lambda_2 = 0$, meaning our shuttles velocity along $\hat{u}_1, \hat{u}_2$ from the suns ref.frame (using (6))

$$v_1 = v_{ref,1} + 0 = 1.22847$$
$$v_2 = v_{ref,2} + 0 = 0.39052$$

which is reasonable, if we think about what happens when we measure 0 dopplershift in our spaceships velocity must "compensate" for the shift caused by the ref.star radial velocity away from our sun, meaning our shuttle is

moving towards the ref.star at the same velocity as the ref.star is moving away from the shuttle (and our sun).

This means that the redshift from the stars motion, and the blueshift caused by the shuttles motion cancel out.

With this we can find the velocity in our x,y plane by using the reverse transformation (ref (7)) But first we convert the angles to radians

$$\phi_1 = \frac{216\pi}{180} = 1.2\pi$$
$$\phi_2 = \frac{117\pi}{180} = 0.65\pi$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{\sin(1.2\pi - 0.65\pi)} \begin{bmatrix} \sin 1.2\pi & -\sin 0.65\pi \\ -\cos 0.65\pi & \cos 1.2\pi \end{bmatrix} \begin{bmatrix} 1.22847 \\ 0.39052 \end{bmatrix}$$
$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -1.34027 \\ -0.24518 \end{bmatrix}$$

where the velocity again is in astronomical unit per year.

We can test our software by checking if we can find our velocity directly after launch relative to our star, which we know from previous studies to be $v = [2.5450, 11.1890]$ in AU/yr. We use our on-board equipment to measure the dopplershifts to our ref.stars

$$\lambda_1 = 0.10236$$
$$\lambda_2 = -0.08739$$

Putting this into our software, our orientation software computes the velocities

$$v_x = 2.54478328$$
$$v_y = 11.18901763$$

in AU/yr.

Lets also test the rotational orientation. We use our onboard camera to shoot an image of the nightsky directly after launch, and feed this image into our orientation software.

The orientation software returns the angle $\phi = 263$ degrees azimuthal from the x-axis.

We do the same with the triangulation software, measuring the distances from shuttle to the star, and the planets with index 0 and 1. Feeding this data into the triangulation software returns the position

$$x = 0.12815454$$
$$y = 0.00010204$$

in [AU].

With these data we are going to run this against the testing software given to us from the great scientist at the ast2000 center of astrophysics. They inform us, after running their testingmodule of our software, that our implementation of the orientation software has been a great success and we can safely install this on our shuttle before we send it into space!

V. DISCUSSION

We saw that our imageanalysis software was successful in determining what direction our onboard camera was facing, and thus finding our orbital orientation. We did however simplify greatly by assuming our camera is facing along the xy-plane, and also the angle the pictures are centered about is a whole number in the range [0 - 359].

This is very unlikely to actually happen in a real life situation, and our software will only be sufficient for our simplified, virtual model of this voyage.

We also saw that our orientation software was succesful in finding the spaceshuttles velocity relative to our star using the Dopplershift measured from the two

reference stars.

In addition, in order to find our spacecraft position, we require quite sensitive equipment. However, if we assume that the onboard distance checking equipment is sufficient precise, we are still relying on the simulated planet positions in our solar system from previous studies. This one we had to get some assistance from the ast2000 research center to be able to deduce the planetary positions, and it is their positions we have used as a basis in the triangulation software.

VI. CONCLUSION

The software we created was sufficient enough to pass the tests that the ast2000 research center presented to us, and this means we were successful in the implementation of the software. However, we did make some major assumptions that will only make the software work in our specific simulation of the solarsystem, but it does give a good idea of how real life software could be made.

ACKNOWLEDGMENTS

Special thanks to the ast2000 research center for aiding us in this study with the testing module for our software.

REFERENCES

- https://en.wikipedia.org/wiki/Stereographic_projection Wikipedia - Stereographic projection

- https://www.uio.no/studier/emner/matnat/astro/AST2000/h20/undervisningsmateriell/prosjektlop/prosjektdeler/part4.pdf AST2000 - Part 4 project