1.
    a. Hi, thank you all for coming.
    b. My name is Jacob Huesman.
    c. My research project is on “Converting 3D Point Cloud Data into 2D Occupancy Grids suitable for Robot Applications.
    d. I realize the title is a bit of a mouthful and you may be wondering why I chose such a crazy specific research topic as an undergraduate student. I’ll try to give you a bit of background.

2.
    a. I’m a member of Bison Robotics, a club here at NDSU. The club was refounded end of fall semester last year.

3.
    a. At the start of the club we had two primary projects, build a quadcopter from scratch and participate in the Robot in Three Days mentoring event.

4.
    a. The club has grown quite a bit since then and the number of projects, reflects this. I think we have about 15-20 projects we’re currently working on.

5.
    a. So how does this relate to my research?
    b. Well as a club we often take on projects that have requirements that extend beyond our current capabilities. Every time so far, thanks to the passion and time each member of our group is willing to put into the projects, this hasn’t been a problem. It became a problem when we decided to take on the NASA Robotic Mining Competition.

6.
    a. The NASA Robotic Mining Competition presents a very interesting engineering challenge.
    b. The goal of the competition is to collect as much regolith as possible in a simulated martian environment.
    c. You get points for passing inspection and mining regolith.
    d. You lose points for every kilogram the bot weighs, how much bandwidth it uses, how much power it consumes, and if it’s not dust tolerant. You also lose 500 points if it can’t operate even semi-autonomously.
    e. We would have to mine 167 kg of regolith in order to offset that number. The autonomy is a huge programming challenge.
    f. So last spring I started doing some research, looking at what had already been done and what we could do.

7.
    a. I ended up stumbling upon ROS, The Robot Operating System.
    b. ROS was developed originally by the Stanford Artificial Intelligence Laboratory, followed by Willow Garage, and finally the Open Source Robotics Foundation. It's a very powerful system, my research certainly wouldn't have been feasible without it.
    c. According to ROS.org, "The Robot Operating System is a flexible framework for writing robot software. It is a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms."
    d. It's also open source and community driven. Universities and companies across the world develop and contribute packages to ROS. Last time I checked there were 2,129 packages available for implementation.
    e. ROS was therefore a great choice for our operating environment. However it did have a pretty big learning curve and didn't solve our sensor problem.

8.
    a. Quick heads up, a point cloud is just a collection of points in 3D space. I find it helpful to think of it as a three dimension picture.

9.
    a. We'd chosen the ASUS Xtion Pro Live for our sensor of choice. It was lightweight, could be used for publishing point clouds, and was also cheap. At $170, it was considerably cheaper than the next best option, a laser range finder which can be found for around five grand.

10.
    b. So the sensor publishes a collection of points representing everything it sees. What good is this? Well not a ton initially. The robot we are making for the NASA competition needs to be able to see the terrain it's navigating on and make decisions based on the shape of the terrain. The problem takes a sentence to describe, but actually encapsulates a very big issue.

11.
    a. The easiest way for a robot to make navigational decisions, is to represent the robot's environment as a 2D "Occupancy Grid". Which is literally a grid of cells that show three values. Black for occupancy, grey for unknown, and white for clear. Limiting the environment to 2 dimensions and having three possible states makes making navigational decisions significantly easier.
    b. The occupancy values actually range from 0 to 100, representing probability of that cell being an obstacle. But to keep the project relatively simple this probability functionality is ignored for the time being.

12.
    a. So why bother with the point cloud in the first place? Well, the occupancy grids and the 2D laser rangefinders that are generally used in generating them, were designed for walled environments. Not martian surfaces. There aren't walls on mars or the moon. There are rocks and craters.

13.
    a. The arena our robot will be operating in reflects this. Lots of little craters and rocks to trip one up.
    b. Current packages that convert 3D Point Clouds to 2D Occupancy Grids simply check to see if there is something directly in front of the camera that it in the robot's way.
    c. It is simple, efficient, and does the job well robots in walled environments. However it doesn't satisfy the criterion for this project.

14.
    a. The North Dakota Space Grant Consortium, an organization dedicated to encouraging NASA relevant research and interest in STEM, was offering summer research fellowships at the time.
    b. Taking into consideration the amount of time this portion of the NASA Robotic Mining Competition was going to take. I thought it would be best to turn it into a separate research project.
    c. While the conversion software I was originally planning on writing would have been made explicitly for the Xtion Camera. It wasn't a huge leap to expand it's scope to any sensor that publishes point cloud data. An algorithm like this could potentially be exceptionally useful for autonomous vehicles on any rugged terrain, including extraterrestrial bodies like the moon and mars. I applied for the fellowship and was lucky enough to be accepted.
15.
    a. But enough background.
    b. My research goal was to create a conversion algorithm that converts point cloud data into occupancy grids. The data collection source was an ASUS Xtion Pro Live. Chosen due to it's affordability and immediate applicability to the NASA Robotic Mining Competition Project.
    c. So how do I take this (point at point cloud) and convert it into this (point at occupancy grid).
    d. I chose to approach the problem by first learning everything I could about the operating environment I'd be working in, how point clouds are generated and used, and how occupancy grids are also generated and used.
    e. Honestly the concepts themselves aren't that complicated, it's just the discovery of them that can be difficult.
16.
    a. ROS is really just a vast collection of packages that can talk to each other. Here is an example from NC State University.

17.
    a. A point cloud file is just a collection coordinates to a point in space. This is actually a part of a point cloud data set. X,Y,Z, and the RGB values that correspond to the color.

18.
    a. A occupancy grid can best be visualized by a table of ones and zeros. The ones representing occupancy, the zeros representing free space.

19.
    a. The best way in my mind to solve the problem was probably influenced most by my recent course in Calc 3. We used vector's a ton in Calc 3 and one of the most important vectors was the normal vector. A normal vector is always perpendicular to the surface of the object it's attached to. So if I could calculate the normal vector for each of the points in the point cloud I was collecting from the sensor, I could determine its orientation in relation to the ground plane with some fairly simple math.
    b. Note that for the first iteration of the algorithm, the ground plane is assumed to be the same as the xy plane, which simplifies the algorithm and math quite a bit. With this assumption you don't have to require the client to specify a ground plane for comparison and you also don't have to worry about the x or y components of the vector.

20.
    a. Let's look at the equations though.
    b. There is a theorem that relates the angle of two vectors with their dot product:
    c. Doing some algebra and taking into account the assumptions mentioned prior, the angle of the normal vector in relation to the ground plane is found to be equal to the arccos of the z component of the normal vector.

21.
    a. Knowing the angle between the surface and the ground plane a cutoff angle can then be specified at which a point is considered an obstacle and not a navigable surface.

22.
    a. Cloud_To_Map the program created to fulfill the research project does exactly that.

23.
    a. It reads in a point cloud being published by ROS

24.
    a. It calculates the normals to the surface of each point, based on a search radius that the user can specify

25.

a. It calculates the size of the occupancy grid.

26.

a. It then populates the occupancy grid with values using the math specified earlier.
b. You may notice that the values being added to the occupancy grid aren't a simple 1 or 0.
c. Rarely, usually in the areas of the point cloud that are sparse, or have few points, a normal vector is calculated in the wrong direction.
d. Since there are often multiple points in the point cloud that correspond to an x,y coordinate of the occupancy grid, to increase accuracy and data reliability, a sum is done.
e. For each point in the point cloud that is not an obstacle a -1 is added to the corresponding cell of the occupancy grid.
f. For each point in the point cloud that is an obstacle a 1 is added to the corresponding cell of occupancy grid. In this way the stray points are absorbed by the numerous correct points.

27.

a. To create a valid occupancy grid, cells that are greater than a threshold are considered an obstacle, all grid cells that are lower than that threshold are considered navigable.

28.

a. The occupancy grid is then published to ROS, for use by other applications.

29.

a. As you can see the program works quite well. This animation is actually a recording of my room being mapped as a point cloud using a SLAM package called rtabmap. The colored points are the point cloud. The black and gray points are the occupancy grid.
b. The program works, however it isn't finished.
c. Cloud_To_Map was designed initially using rtabmap to publish the point clouds used in the conversion. Unfortunately during development I didn't realize it was doing transformations in the background that simplified the conversion process. In order for Cloud_To_Map to work, the points in the point cloud need to correspond to the cells of the occupancy grid. Rtabmap did this transformation for me. In addition to save processing, it runs the point cloud through a filter to decrease the number of points it has to process.

30.

a. As you can see running cloud_to_map on on a point cloud directly from the sensor results in an occupancy grid that while, accurately dimensioned and correct in most aspects, needs to be transformed into the right frame.
b. The processing also takes longer than is reasonable. With Rtabmap the conversion takes a few milliseconds. With a raw point cloud in this demo it takes about 10 seconds.

This is way too long and is due to the program getting bogged down by redundant points. These are problems I’m planning on addressing over winter break.

31.
    a. Once the package is finished and tested for bugs. Once it works consistently as well as this example, it will be published on ROS.org for use by developers around the world.

32.
    a. Any questions?