CS 309: Autonomous Intelligent Robotics
FRI I

Lecture 22:
Final Project Proposals
Getting Through HW5

Instructor: Justin Hart

http://justinhart.net/teaching/2018_spring_cs309/

About the homework

- Homework 5
  - Due April 12
  - Start early, it is hard and involves the robots
  - Do everything in the HW5 pdf, but do navigation as updated in this lecture
- Homework 6
  - Also due April 12
  - 1.5 page description of your final project plan
    - Discuss with me in advance

Undergraduate Research Forum!

- Friday, April 13
- 11:00am – 2:00pm
- We will have a poster!
- I am looking for students who would like to present this poster.
- I will be judging the competition.
- This is a good opportunity to practice presenting in a professional setting, and should be easy, laid-back, and fun.
- If you’re interested, contact me.

Robotics Consortium!

- Thursday, April 12
- 12:00pm – 1:30pm – Lunch included!
- We will have the same poster!
- This is similar to the URF, except that it will be to laboratories and companies who collaborate with UT through the Robotics Consortium!

Poster Presentations

- Do either, and earn 0.5 pts of extra credit on your final grade.
- Do both for 1 full point.
- You don’t need to be there the whole time, but we need people the whole time.

A couple of quick notes

- Robotics Study
  - If you're free, we appreciate the help. See the Canvas announcement.
- RoboCup@Home
  - We're still getting ramped up and you're welcome to participate!
- Unique ID for Fall 2018
  - **PLEASE** double-check this, I don't have the official number yet, but I think it will be: **51705**
  - When you sign up, make **SURE** that you are signed up for the correct class (instructor: Hart)

Today

- Final Project Proposals
- HW5

Final Project Proposals – Outline

- Introduction
  - What problem are you trying to solve?
  - Why is it important?
- Background (Optional at this stage)
  - What approaches have previously been taken to solve this problem, and by whom?

Final Project Proposals – Outline

- Approach
  - What approaches are you considering?
  - Is there a piece of software that you intend to run?
  - This part will be the most thought out and should be about half of your paper
- Conclusion
  - 1 paragraph, less than ¼ page
  - Briefly restates your problem and approach, why you think it will work, and what you think you will have accomplished.

Final Project Proposals – Outline

- These (or something similar) should be the *actual* headings in your proposal.

Final Project Proposals – Formatting

- LaTeX is the easiest way.
- Supposing you don’t use latex
  - 10pt font
  - Title centered
  - 2 columns
  - 24pt for title
  - 16pt for authors
  - 16pt for section headings

Final Project Proposals – LaTeX

- Writing in LaTeX is simple
- Download the IEEEtran package from https://ctan.org/pkg/ieeetran?lang=en
- Unzip onto a Linux machine, all of the machines in the lab have LaTeX
- Edit your paper inside bare_conf.tex

Final Project Proposals – LaTeX

- Lines starting %% or % are comments
  - You can safely delete them!
- This will leave you with a block that looks like this:

```
\ifCLASSINFOpdf
\else
\fi
```

  - This block does nothing, delete it

Final Project Proposals – LaTeX

- It will also leave you with a title block with other people's names in it!

  \title{Bare Demo of IEEEtran.cls\\ for IEEE Conferences}\author{\IEEEauthorblockN{Michael Shell}

  - \IEEEauthorblockA{School of Electrical and\\Computer Engineering\\
  - Georgia Institute of Technology\\
  - Atlanta, Georgia 30332--0250\\
  - Email: http://www.michaelshell.org/contact.html}
  - \and
  - \IEEEauthorblockN{Homer Simpson}
- Put your paper's title, your names, and info in there

Final Project Proposals – LaTeX

- Delete this thing, your paper is too short for an abstract.

```
\begin{abstract}
The abstract goes here.
\end{abstract}
```

Final Project Proposals – LaTeX

- Delete this thing, your paper is not going into peer review.

  \IEEEpeerreviewmaketitle

Final Project Proposals – LaTeX

- Each one of these things marks a section of your paper, or a subsection. Delete and re-arrange as appropriate

```
\section{Introduction}
\subsection{Subsection Heading Here}
```

- The text under them is the literal text of your section, so, erase what's already there (including \hfills and such) and put in your real text.

Final Project Proposals – LaTeX

- Delete this thing, your paper is not going into peer review.

  \IEEEpeerreviewmaketitle

Final Project Proposals – LaTeX

- It will also leave you with a title block with other people's names in it!

  \title{Bare Demo of IEEEtran.cls\\ for IEEE Conferences}\author{\IEEEauthorblockN{Michael Shell}

  - \IEEEauthorblockA{School of Electrical and\\Computer Engineering\\
  - Georgia Institute of Technology\\
  - Atlanta, Georgia 30332--0250\\
  - Email: http://www.michaelshell.org/contact.html}
  - \and
  - \IEEEauthorblockN{Homer Simpson}
- Put your paper's title, your names, and info in there

Final Project Proposals – LaTeX

- It will also leave you with a title block with other people's names in it!

  \title{Bare Demo of IEEEtran.cls\\ for IEEE Conferences}\author{\IEEEauthorblockN{Michael Shell}

  - \IEEEauthorblockA{School of Electrical and\\Computer Engineering\\
  - Georgia Institute of Technology\\
  - Atlanta, Georgia 30332--0250\\
  - Email: http://www.michaelshell.org/contact.html}
  - \and
  - \IEEEauthorblockN{Homer Simpson}
- Put your paper's title, your names, and info in there

Final Project Proposals – LaTeX

- Acks

  % use section* for acknowledgment

  \section*{Acknowledgment}

  The authors would like to thank…

- You can safely delete this.
  - This is where we say who paid for everything.
  - Or, if someone helped you do your project, you thank them
    - But getting real, they'd rather be listed as a co-author in the real world.

Final Project Proposals – LaTeX

- You can delete this thing, too

```
\begin{thebibliography}{1}


\bibitem{IEEEhowto:kopka}
H.~Kopka and P.~W. Daly, \emph{A Guide to \LaTeX}, 3rd~ed.\hskip 1em plus
0.5em minus 0.4em\relax Harlow, England: Addison-Wesley, 1999.
\end{thebibliography}
```

- For your final project report, you will **need** a bibliography, but we will use LaTeX and Bibtex for that

Final Project Proposals – LaTeX

- You can delete this thing, too

```
\begin{thebibliography}{1}


\bibitem{IEEEhowto:kopka}
H.~Kopka and P.~W. Daly, \emph{A Guide to \LaTeX}, 3rd~ed.\hskip 1em plus
0.5em minus 0.4em\relax Harlow, England: Addison-Wesley, 1999.
\end{thebibliography}
```

- For your final project report, you will **need** a bibliography, but we will use LaTeX and Bibtex for that

Final Project Proposals – LaTeX

- Acks

  % use section* for acknowledgment

  \section*{Acknowledgment}

  The authors would like to thank…

- You can safely delete this.
  - This is where we say who paid for everything.
  - Or, if someone helped you do your project, you thank them
    - But getting real, they'd rather be listed as a co-author in the real world.

Final Project Proposals – LaTeX

- Acks

  % use section* for acknowledgment

  \section*{Acknowledgment}

  The authors would like to thank…

- You can safely delete this.
  - This is where we say who paid for everything.
  - Or, if someone helped you do your project, you thank them
    - But getting real, they'd rather be listed as a co-author in the real world.

Final Project Proposals – LaTeX

- It will also leave you with a title block with other people's names in it!

  \title{Bare Demo of IEEEtran.cls\\ for IEEE Conferences}\author{\IEEEauthorblockN{Michael Shell}

  - \IEEEauthorblockA{School of Electrical and\\Computer Engineering\\
  - Georgia Institute of Technology\\
  - Atlanta, Georgia 30332--0250\\
  - Email: http://www.michaelshell.org/contact.html}
  - \and
  - \IEEEauthorblockN{Homer Simpson}
- Put your paper's title, your names, and info in there

Final Project Proposals – LaTeX

- It will also leave you with a title block with other people's names in it!

  \title{Bare Demo of IEEEtran.cls\\ for IEEE Conferences}\author{\IEEEauthorblockN{Michael Shell}

  - \IEEEauthorblockA{School of Electrical and\\Computer Engineering\\
  - Georgia Institute of Technology\\
  - Atlanta, Georgia 30332--0250\\
  - Email: http://www.michaelshell.org/contact.html}
  - \and
  - \IEEEauthorblockN{Homer Simpson}
- Put your paper's title, your names, and info in there

Final Project Proposals – Word/Libre/OpenOffice

- If you’re more comfortable with these programs, just make the output look similar

HW5

- Stop trying to make my code compile!!
  - This is a huge waste of your time and energy.
  - I took my example code from class and deleted the sections that give you the answer.
  - I also deleted the parts where I do it ***incorrectly!!***
  - Getting this code up and running ***would*** solve your homework.
    - But I think it's harder than your homework is.
  - Also, the ***point*** is that you ***understand*** how this program works.
    - If you're really stuck on getting something I wrote to compile, it's because you don't understand how it works.

AlvarMarker

- This is where the Pose of the marker comes from.
  - It is relative to the frame you provide to the class
    - If you are using the newer package. Use the one from the newer package.
    - If you are using the older one, it’s with respect to the kinect’s frame, but this needs to be modified in the robot case.

AlvarMarker

- AlvarMarker ONLY fires when Alvar is running, connected to a Kinect, and when a marker is in view.
  - If these three things are not true, you cannot test your code properly.
  - So share the Kinect in the lab.

PoseRecipient

- PoseRecipient is a class that is intended to receive a geometry_msgs::Pose
  - virtual void receivePose(geometry_msgs::Pose &pose) = 0;
- You provide a PoseRecipient to AlvarMarker to get the Pose of the marker relative to the camera.
  - DO NOT TRY TO USE TF FOR THIS!!
  - AlvarMarker has the **correct** solution.
  - If you solve this using TF, good for you, but it’s **much** harder.

PoseRecipient

- Inherit from PoseRecipient when you implement most of your classes.
  - The way that I solved the homework involved almost everything inheriting from PoseRecipient.

TFBroadcastPR

- Implement a PoseRecipient that broadcasts the pose it receives first.
  - This lets you see that you’re using AlvarMarker correctly.
  - It introduces you to using PoseRecipients and if they are working correctly.
  - You need to do this anyway.
- Re-implementing the broadcast functionality for “offset” and “offset_flipped” is **silly**
  - You can simply pass this PoseRecipient to your class and call it with the result.

OffsetPR

- OffsetPR should compute the offset pose.
- Think of a rough outline like this
  - OffsetPR(double x, double y, double z, PoseRecipient &nextInChain);
- Then, inside your receivePose method

```
void OffsetPR::receivePose(geometry_msgs::Pose &pose) {
    //Transform the pose here
    _nextInChain.receivePose(new_pose);
}
```

- Where _nextInChain has been set to be your TFBroadcastPR

OffsetR / OffsetTR

- These are the examples from class where I showed you **incorrect implementations** of the homework.
  - Seriously, stop trying to get these to compile.
  - You need to write code that you **understand**.
  - The code I put online is intended as a guide book, not a fill-in-the-blanks method for solving the homework.

Computing your offset

- Order of operations MATTERS.
- I don’t want to give away the answer here, but the order matters.
  - The offset is with respect to the MARKER.
  - So 1 meter in front of the marker.
  - This is one meter ROTATED WITH RESPECT to the marker’s frame.
  - The marker itself is translated with respect to the frame it is computed relative to.
    - This is the frame AlvarMarker is relative to.
  - This gives you a really straightforward order of operations.
    - Translate the point in front of the marker into the MARKER’s frame, rotate into the orientation that the marker is with respect to its parent frame, then translate relative to that frame.
  - TF can’t solve this for you because YOU are introducing this frame.

Computing your offset

- Note that we just put the **position** of the offset
- The **orientation** is supposed to be the same as that of the marker.
- In the **flipped** case, it should be rotated 180 degrees with respect to the marker.
  - Flipping 180 degrees should be the **first** rotation you do.

Computing your offset

- So:
  - Position with respect to the marker
    - 1m in front (on the z axis)
  - Orient with respect to the marker.
    - Nothing really to do here.
  - Then orient with respect to the marker’s PARENT FRAME.
    - ROTATE into this frame, the same rotation as the marker.
  - And translate with respect to the marker’s PARENT FRAME.
    - TRANSLATE into this frame, the same origin/translation as the marker.

Flipping your offset

- Flipping 180 degrees happens **FIRST**, so rotate 180 with respect to the **MARKER’S** frame prior to moving into the **MARKER’S PARENT’S** frame.

Break the problem into small parts

- FIRST
  - Broadcast TF and check in rviz that this is working, with respect to the AlvarMarker.
- Second
  - Compute your offset. Stick that computation in between AlvarMarker and your TF Broadcast.
- Third
  - Flip your offset, and test that.

To help you out..

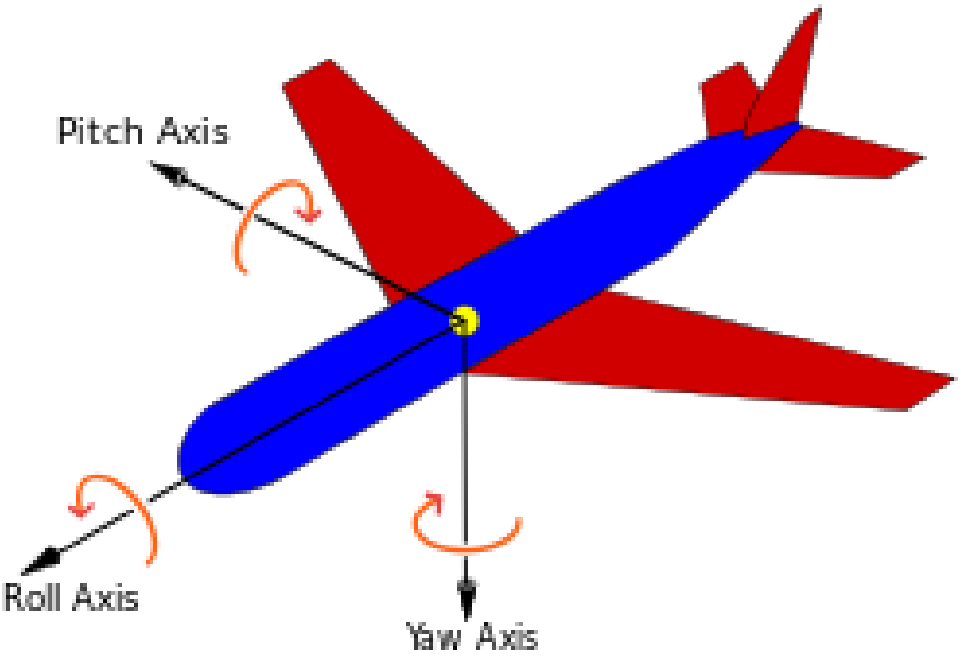
- Take a look at SplitterPR. That will allow you to fork the pose from PoseRecipient into multiple PoseRecipients so you can run more than one at a time.

Following the marker

- The second half of this is getting the robot to follow the marker.
- Write a SECOND program.
  - Think of a triangle that looks like this.
    - One corner is the camera on the robot.
    - One corner is the marker *if* it were at the same height as the camera (y=0)
    - One corner is the point that would form a right angle in this triangle

tf::Quaternion RPY

- Our robot really can only turn on its YAW
- tf::Quaternion has an RPY constructor
  - Set R & P to ZERO
  - Set YAW to a fraction of the yaw you computed.
  - If you set it to the full yaw, you will overshoot.
    tf::Quaternion rpy(0, 0, frac*yaw);
  - Now, you can broadcast the x,y,z,y from your quaternion.



move_base_msgs::MoveBaseGoal

- Remember that steering the robot uses MoveBaseGoals.
  - MoveBaseGoal has a pose in it, which is the target pose.
  - Which has a position (xyz) and an orientation (xyzw)
- Don’t use the full yaw angle computed or full distance, you will overshoot your goal.
- Send your MoveBaseGoal using MoveBaseClient.sendGoal();