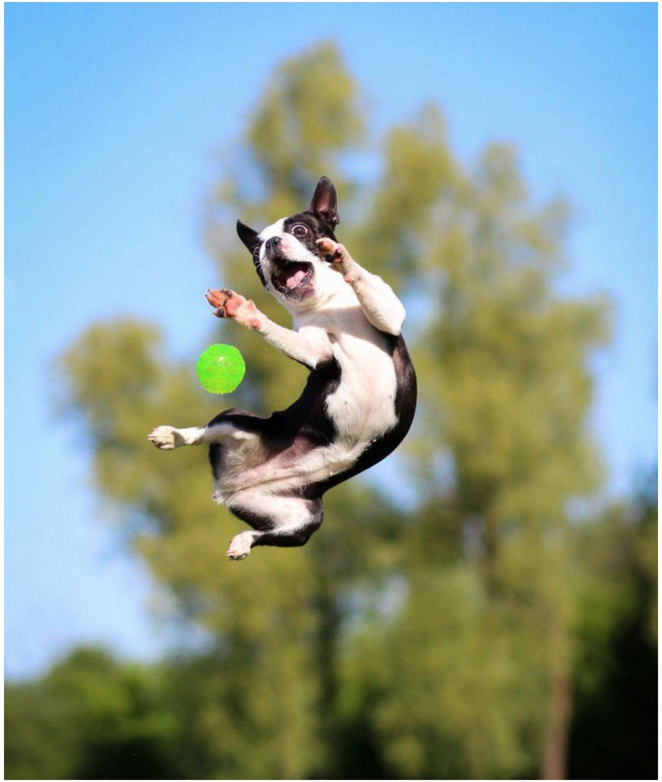
That's So Fetch

B4 (Luca Amblard, Dan Barychev, Hana Frluckaj)
Presented by: Luca Amblard

Application Area

- Motorized device that can:
    - Anticipate user's throw using motion sensors on hand
    - Move to predicted landing location in real time
    - Catch the object thrown
    - Returns to original position
- Users:
    - People allergic to dogs but still want to play a game of Fetch
    - Fun alternative to having a pet



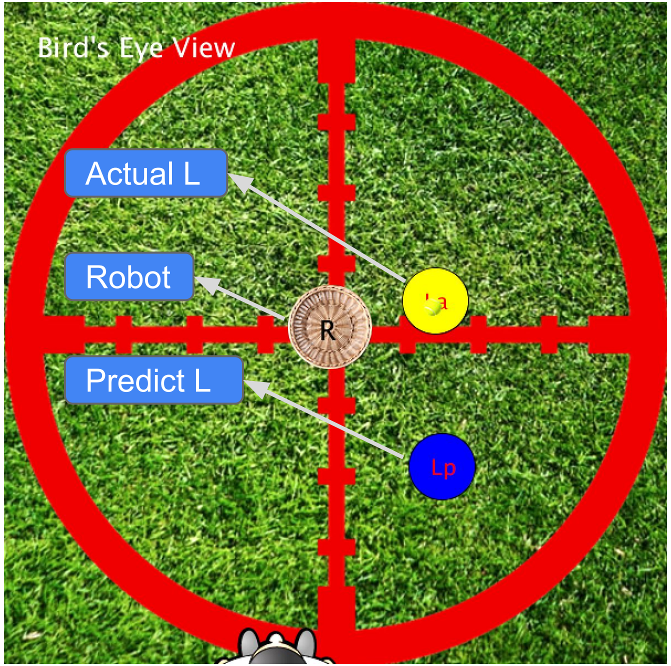
Solution Approach: General

Photon reads IMU data	Detect ball release	Determine throw data	Prediction	Simulate Fetch
Photon reads finger IMU and hand IMU data through I2C	Detect when ball is thrown from hand through finger IMU angular velocity	Determine Vx, Vy, Vz, throw height, and horizontal angle of ball at throw using Madgwick's AHRS sensor fusion algorithm	Predict ball landing location and time of flight using equations of motion in 3D and measure actual landing location in real life grid	Simulate the throw/catch process after data is fed into the simulation

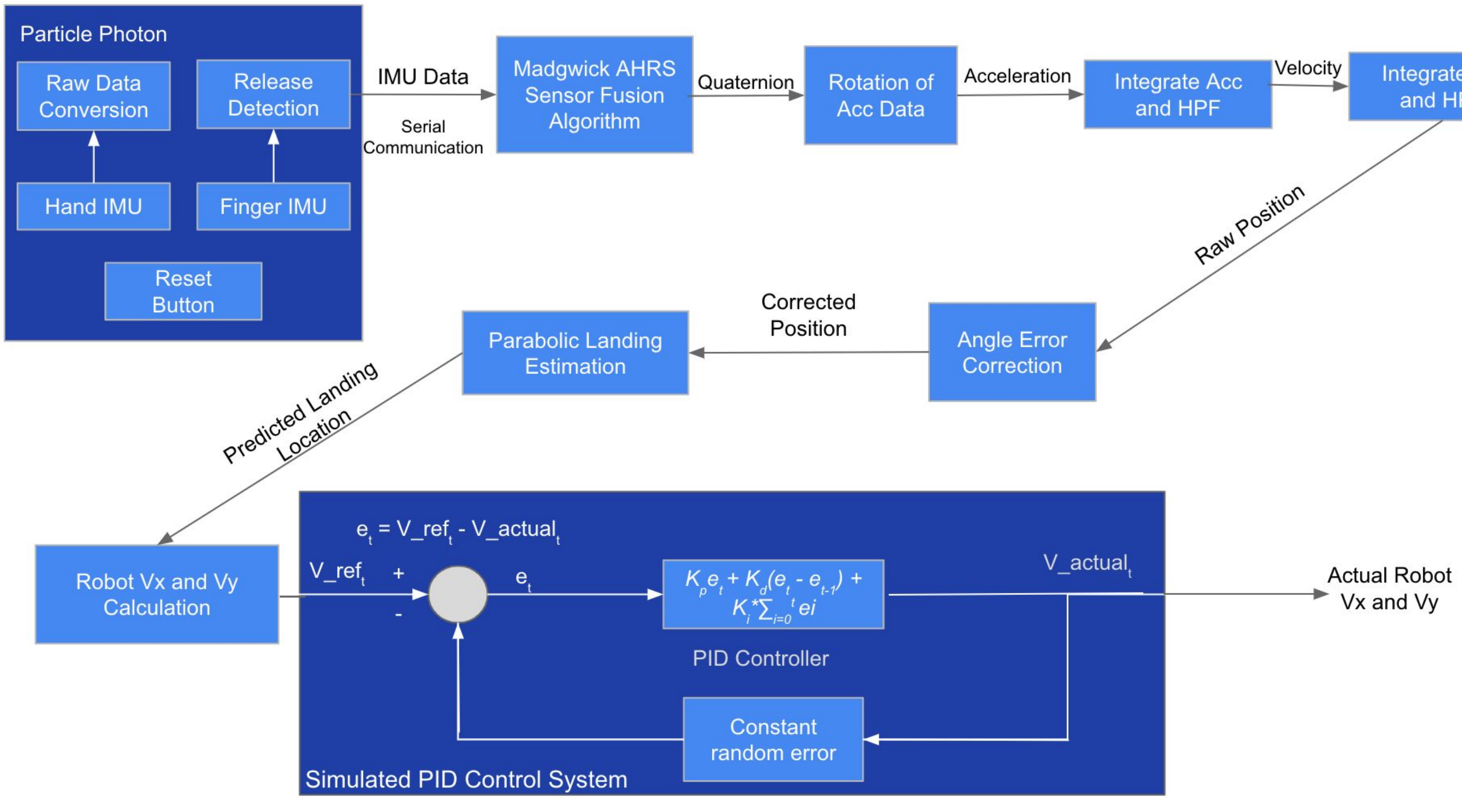
Solution Approach: Changes

- Changes due to COVID-19 constraints:
    - No physical robot for retrieval → move to simulation based motorized retriever
    - Simulate Fetch using inputs and timing
- Changes due to current testing constraints:
    - Wireless capability too slow → serial output recording through micro-usb used instead
    - IMU sensing misinterprets fast throws → robot catching range decreased to 1m
- Changes due to Design Improvement:
    - Kalman filter solutions resulted in too much drift with our sample rate of 50Hz.
    - Switch made to AHRS for accuracy

Simulation based design:



Block Diagram



Solution - Simulation

The simulation presents two views of the project:

Bird's Eye View ➡️

View of robot moving in order to retrieve ball

Side View

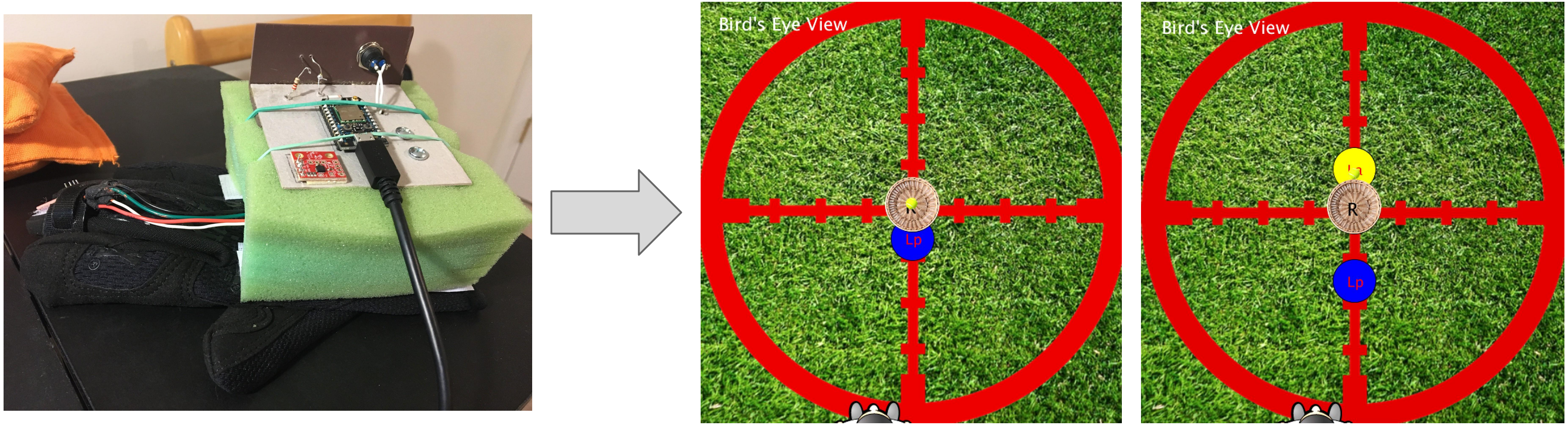
View showing the ball's trajectory and how far up/back the robot must move to retrieve it

Metrics and Validation

Process	**Specs**
Success Rate *(#balls thrown v. #balls caught)*	> 50%
User throw range *(distance between user and dog)*	1m radius
Device retrieval range	1m radius
Device basket diameter	25cm
Difference predicted ball landing position and actual landing position	< 12.5 cm
Minimum prethrow number	20
AHRS computation time	< 0.5s

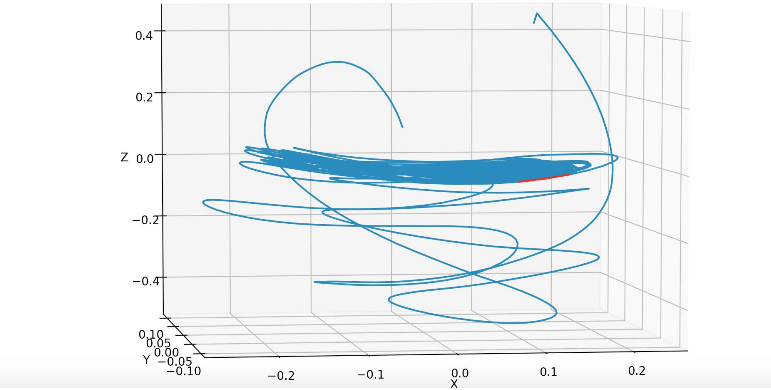
Metrics and Validation: Results

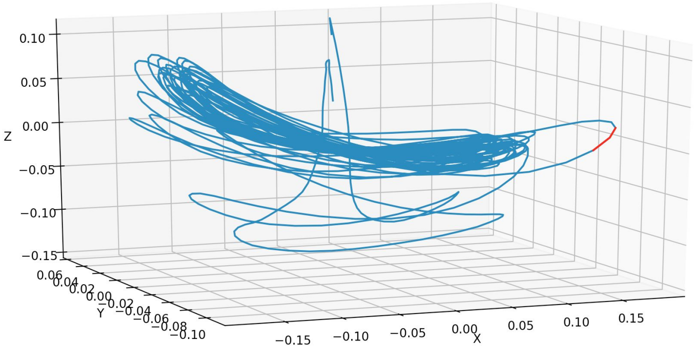
- Our tests focused on the system’s ability to create accurate estimates upon a straight line
- We currently have a ~50% catch rate with our most reliable data
- These are the results of two straight throws of around 1m length



Metrics and Validation: Results Cont.

- Madgwick AHRS Algorithm gives us clear parabolic throw data
- Our throws conform to the type of cyclic motion the algorithm handles well
- 95% of our data files process the results in under 0.5s (allowing the simulated robot enough time to make the catch)





Metrics and Validation: Trade-offs

- Kalman v. Madgwick's AHRS
  - Kalman filter: best between 512Hz and 30kHz, but exhibited far too much drift at 50Hz
  - Madgwick's AHRS filter: uses gradient descent and quaternions to give rotation data, allowing for integrable acceleration data
- Wireless v. Serial
  - Able to achieve wireless functionality with Particle Photon but data transmission rate was too slow. Instead, relied on long micro-USB cable for serial communication to have free movement
- IMU in ball v. IMU on hand
  - IMU in the ball would give information concerning the ball's path. This would be hard to estimate with IMU positioning so we decided just to place one on the hand instead
- Cornhole bag v. hacky sack
  - Decided to use a cornhole bag since it rarely bounces, although a hacky sack is much easier to throw and restricts arm motion much less

Validation - IMU data and Simulation

How to verify IMU data

- IMU provides height object is thrown at, as well as velocity and angles in three dimensions
- These data points can be roughly checked through recorded video
- Madgwick's AHRS algorithm prefers cyclic motion since mean position and velocity are 0
- Deviations from this in the throw pattern lead to angle and position errors that we correct for using trigonometry

How to verify a simulation

- Difficult to take all variables into account: air resistance, object weight, etc.
- Ball will be thrown and ideally land within the bounds of a measured grid
- The actual landing location will be compared to the result of the simulation
- Accuracy goal: >50%

Project Management

Tasks	M	T	W	R	F	S	S
Motion Sensing							
Determination of user rotation							
Determination of user starting height							
Refinement 3D Positioning							
Recording of more trials							
Simulation: Bird's Eye							
Display altitude, time of flight, and distance travelled							
PID simulator							
Incorporate real world behavior							
Display distances on screen							
Simulation refinement							
Simulation: Side View							
Graphics for side view							
Display inputs and distance							
Ball motion synchronization with bird's eye view ball							
Final Report							
Final Video							
Key:	**Luca**	**Daniel**	**Hana**	**Luca/Daniel**	**Luca/Hana**	**All**	