

# **Final Presentation**

Vedant Parekh, Siddesh Nageswaran, Alvin Shek Advisors: Professor Savvides, Rashmi Anil

# Introduction and Motivation

- What is Hawkeye?
  - a. An automatic drone tracking system with live aerial footage
  - b. Shoots aerial video that does not require human control, eliminating human error
- Use Cases:
  - a. Useful for recreational filming, rescue missions etc.
  - Imagine shooting exciting videos of sports events (your Turkey Bowl game), or having a hands-free vlogging experience!
- Key User Requirements:
  - a. Drone Tracking: % of frames with target in shot
  - b. Drone Stability: Minimal drone jitter, steady shot





### System Specification



# System Specification – Software





# **Complete solution**

- We will show:
  - Drone manually taking off and streaming video to wearable device
  - Live display of target detection and state estimation results along with the resulting planned flight path
    - Flight path will not be broadcasted back to the drone
    - Drone will be guided manually
  - Full demonstration of autonomous motion in simulation
    - Drone will follow the given flight path







# Challenges

### **Local Position Error:**

- Flight controller's internal x,y,z local position estimates are very off
- When still, positions drift within +/- 3 m every few seconds
- Seemingly random where the drone ends up when given the same positional waypoint

### Extreme Sensitivity to Wind:

• Drone topples sideways in even slight wind gusts

#### Dealing w/ the Issues:

- Cannot fly drone safely fully autonomously
- Instead, we can demonstrate target detection / motion planning while flying drone manually rather than having the drone follow the resulting motion plan
- Autonomy demonstrated on simulated flight controller with exact same interface as real flight controller





**Figure:** Drone flailing wildly while trying to hold position during mild wind



# Testing: Image Processing

Operation	Average Time Taken (s)	FPS	
Capture Image	0.344	2.91	Bottleneck
Stream Image to TX1	0.25	4	
Convert Image to Cv2	7.14e-5	14006	
Detect Target	2.67e-4	3745	
Estimate 3D Position + Kalman Filter	6.50e-4	1538	
Motion Planning	0.0135	74.07	
Overall	0.344	2.91	
Desired		5 - 10	

HøwkEye

# Testing: Target Detection

False Positive Rate:#Images with detection / No target presentFalse Negative Rate:#Images without detection / target is presentAverage Pixel Error:Distance from predicted target center to actual center

	FP Rate	FN Rate	Avg. Pixel Error				
Actual	0%	14.78%	11.87				
Desired	2%	10%	(N/A)				







 $\frac{1}{N} \sum_{i=1}^{N} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}$ 

Figure: Example false negative



Figure: Predicted vs. actual target center

# Testing: Drone Stability and Tracking (simulation)

**Drone Tracking:** % of frames where the target is within frame **Drone Stability:** % of 3 second windows where drone position is stable

Testing Condition	Tracking	Stability					
Walking Only	100%	100%					
Running Only	88%	100%					
Walking + Running	97%	93.75%					
Desired	90%	90%					





### Trade-Offs

#### **Motion Planning**

- Balancing control cost vs. tracking accuracy
- More control cost = more stable/minimal movement
- Less control cost = jerky movement but higher tracking accuracy

	Current Design	Higher Control Cost	Lower Control Cost
Tracking	97%	47%	84.58%
Stability	93.75%	100%	43.75%



**Figure:** Motion planner with very high control cost (stable but can't adapt to changes in target motion)



Figure: Motion planner with very low control cost (jerky movement)



### Trade-Offs

#### **Image Streaming**

- Uncompressed Images (what we use) vs. Compressed Images
  - Bottleneck -> camera capture
    - Uncompressed: 2.91 FPS, Compressed: 2.94 FPS
  - There IS a difference in streaming (4 FPS vs. 6.67 FPS), but that's irrelevant since it isn't bottleneck
  - No benefit to loss of quality from compression

### State Estimation

- New target detection data vs. Current target model
  - The more that new data is weighted, the noisier the predicted path potentially becomes
  - The more the current model is weighted, the more the predicted path drifts from the actual one
- Model of target's acceleration
  - Modeling a higher potential for acceleration makes path jerkier
  - But modeling a lower potential for acceleration may cause predicted path to lag behind actual one



# Updated Schedule

		Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14		1	Legend:
1 Phase 1: Design															/edant
1.1 Make overall block diagram															Alvin
1.2 Choose/finalize components															Siddesh
1.3 Set-Up Jetson TX1															Together
1.4 Order components													10		
1.5 Familiarize ourselves with dro	one API and simulator														
1.6 Familiarize ourselves with Je	Ison TX1 API														
1.7 Familiarize ourselves with we	arable display communication protocol														
1.8 Design Presentation															
1.9 Design Report															
2 Phase 2: Pre-integration															
2.1 Implement color filtering and	blob detection					1									
2.2 Test detection on static image	25														
2.3 From target data across multi	ple frames, calculate velocity vector					2									
2.4 Use calculated velocity along	with a model for target movement to accurately predict future movements						1								
2.5 Test detection on live video (I	ong distance with smaller target since blob detection parameters need to be tune	ed)													
2.6 Successfully send sample mo	otion commands to drone in simulation, then on physical drone														
2.7 Design a general motion plan	ning stack; design to easily integrate with target tracking														
2.8 Test and debug the motion pl	anning stack														
2.9 Implement communication be	tween camera and RPi Zero, film sample video														
2.10 Implement video streaming o	ver Wifi between RPi and TX1					-									
2.11 Implement hardware protocol	(ex. UART, I2C) to interface between TX1 and display														
3 Phase 3: Integration							-								
3.1 Map target's motion in video	to desired motion of drone														
3.2 Design circuitry for the weara	ble device (ex. display, buttons, power supply etc.)														
3.3 Hook up buttons to the TX1 a	ind configure them to send start and stop instruction to RPi														
3.4 Integrate RPi start stop signa	Is received from nano with the drone flight controller														
3.5 Design safety fallback behavi	or and validate on simulation and on drone														
3.6 Integrate the flight parameter	data generated by TX1 with the flight controller														
3.7 Create housing for Jetson TX	1 and display so that they can easily be worn by the user														
3.8 Create chassis to house RPi	and camera onto the drone and verify camera angle														
4 Phase 4: Performance testi	ng	-													
4.1 Verify that video streaming pe	erforms with a stationary target														
4.2 Verify drone can continuously	r track target during flight w/o streaming video II														
4.3 Verify that drone can match s	peed of target that is varying between speedsll														
4.4 Verify live video from drone c	an be seen on wearable display for a moving targetll														
4.5 Iterate and repeat tests II															
4.6 Slack															
5 Final Report															
5.1 Record demo video II															
5.2 Edit video II															
5.3 Final presentation II															

