HawkEye

Final Presentation

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

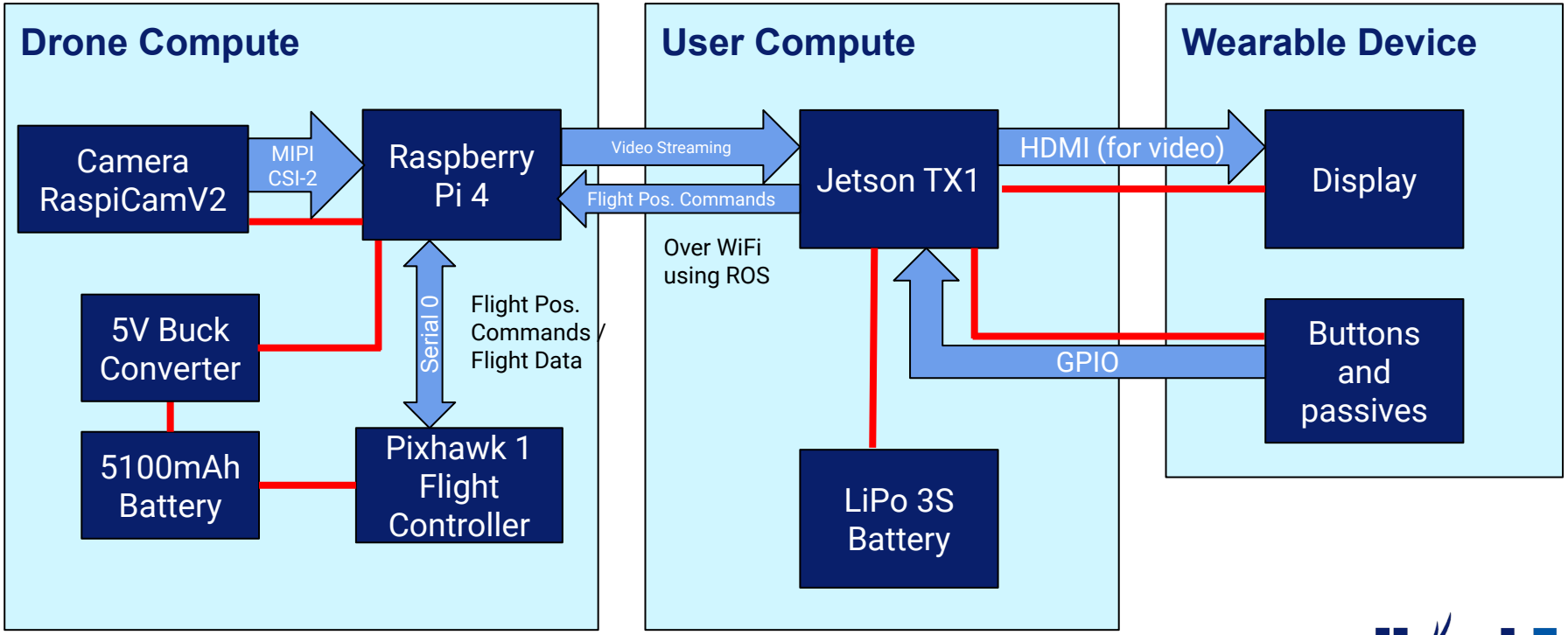
Introduction and Motivation

- 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.
  b. Imagine shooting exciting videos of sports events (your Turkey Bowl game), or having a hands-free vlogging experience!

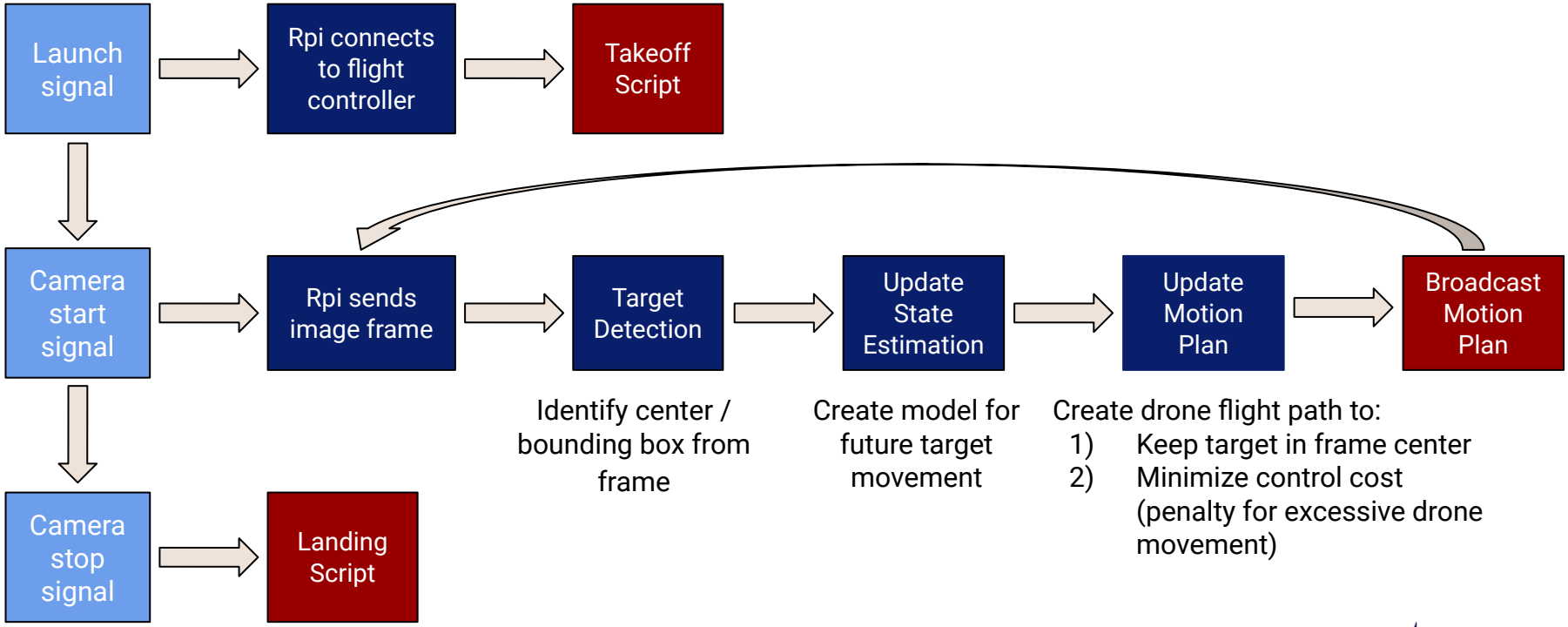
- Key User Requirements:
  a. Drone Tracking: % of frames with target in shot
  b. Drone Stability: Minimal drone jitter, steady shot

System Specification



HawkEye

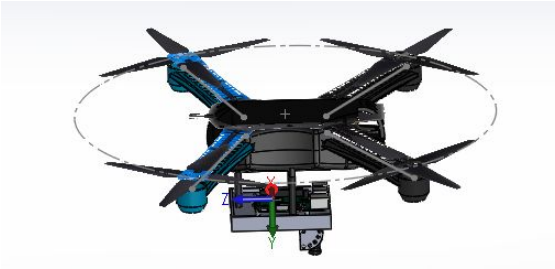
System Specification – Software



HawkEye

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



HawkEye

Challenges

Local Position Error:

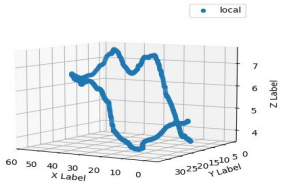
- Flight controller's internal x,y,z local position estimates are very off
- When still, positions drift within +/- 3 m every few seconds
- Seemingly random where the drone ends up when given the same positional waypoint

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



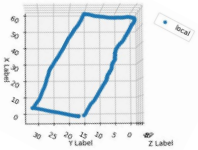


FIgure: Local position estimates for the path walked above

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

HawkEye

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**	

HawkEye

Testing: Target Detection

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

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

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

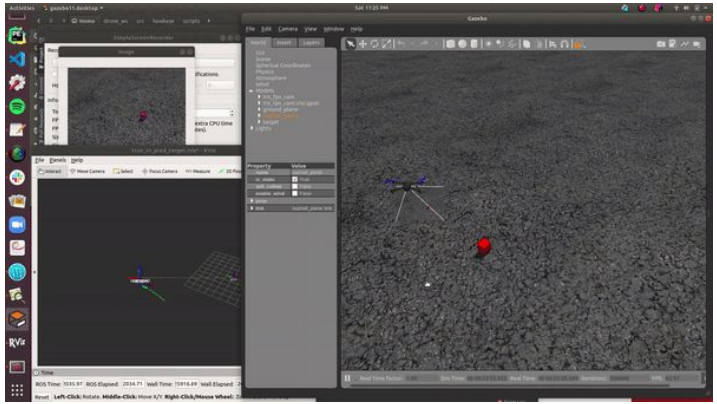
Figure: Predicted vs. actual target center

Figure: Example false negative

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%



HawkEye

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%

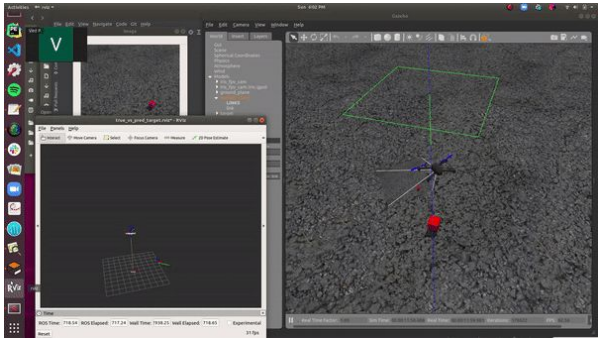


Figure: Motion planner with very high control cost (stable but can't adapt to changes in target motion)

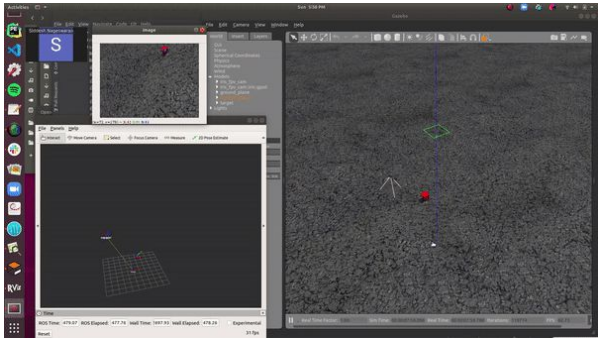


Figure: Motion planner with very low control cost (jerky movement)

Trade-Offs

Image Streaming

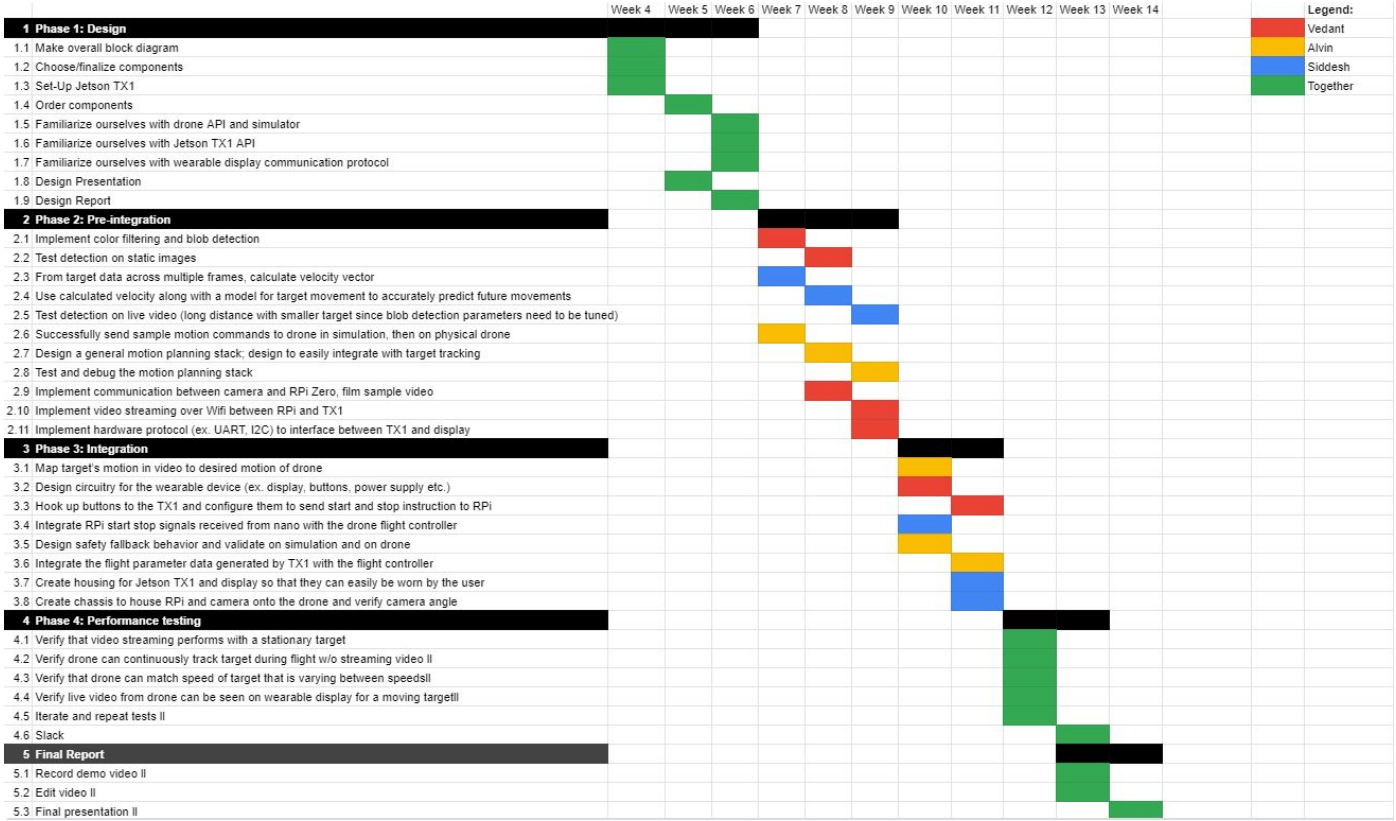
- Uncompressed Images (what we use) vs. Compressed Images
  - Bottleneck -> camera capture
    - Uncompressed: 2.91 FPS, Compressed: 2.94 FPS
  - There IS a difference in streaming (4 FPS vs. 6.67 FPS), but that's irrelevant since it isn't bottleneck
  - No benefit to loss of quality from compression

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

HawkEye

Updated Schedule



HawkEye