Evaluation Computer Vision Applications on a Drone Over 5G Connection

Aditi Agarwal
aa2224@cornell.edu
Cornell University
Ithaca, New York, USA

Kevin Huang
kqh3@cornell.edu
Cornell University
Ithaca, New York, USA

Chen-Tung Chu
cc2396@cornell.edu
Cornell University
Ithaca, New York, USA

Nei-Chun Shao
ns948@cornell.edu
Cornell University
Ithaca, New York, USA

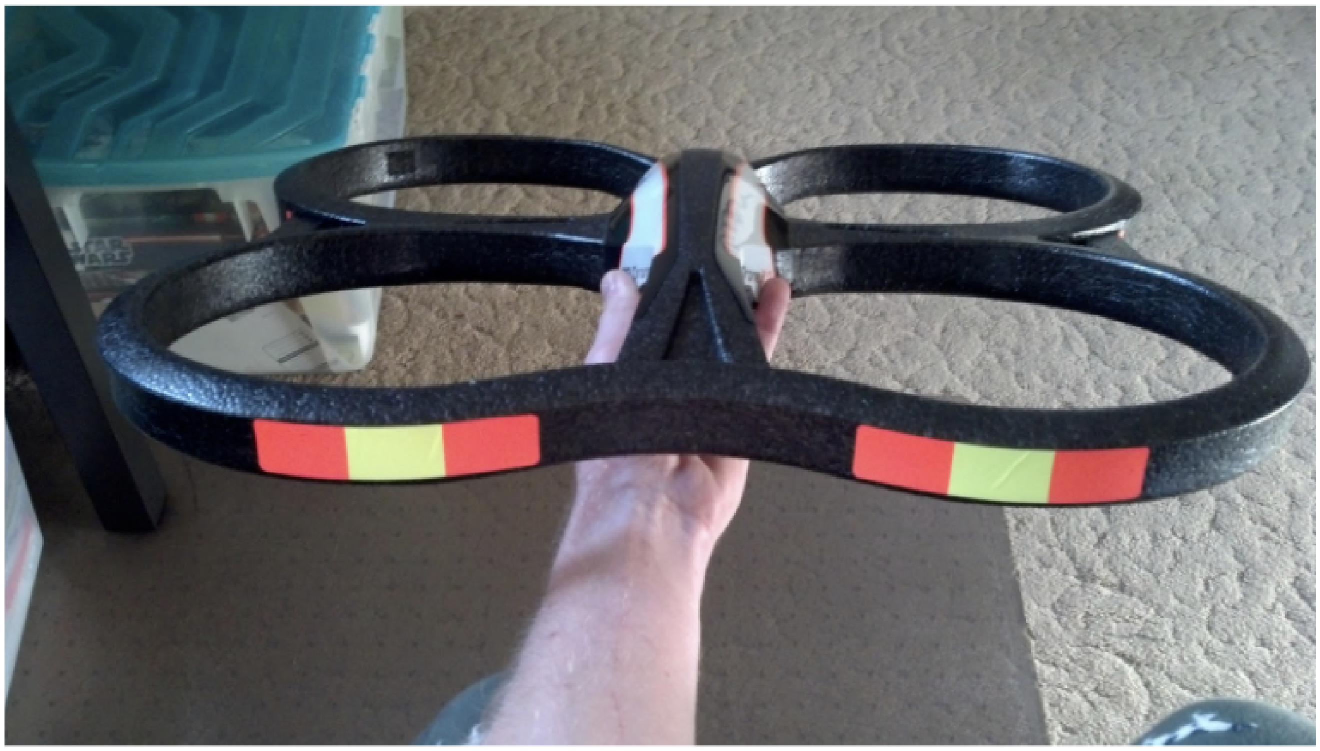


Figure 1: AR Drone 2.0

ABSTRACT

5G is an emerging technology that promises better performance and usage flexibility with many use cases such as in IoT applications. One potential use case is drone

ECE 5710 SP 22, ECE Datacenter Computing, Ithaca, NY
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

applications, which are usually latency-contrained and used in outdoor settings with more limited connection options.

In collaboration with Cornell's *Aether* project to install a 5G local network on Cornell's campus, we evaluated Computer Vision applications on a drone over 5G connection as compared to running over WiFi and running locally on the drone. We found that data-heavy applications, such as those with video streams, are accelerated by 5G, however, less data-heavy applications such as those that use images or any smaller packet sizes have no significant difference in latency. Compute-heavy applications were shown to run poorly locally on the drone and would benefit from offloading to a remote server.

CCS CONCEPTS

• **Networks** → **Network reliability**; *5G*; WiFi; • **Computer Vision** → On-drone vs offload.

KEYWORDS

drone, computer vision, 5G local network, WiFi

ACM Reference Format:
Aditi Agarwal, Kevin Huang, Chen-Tung Chu, and Nei-Chun Shao. 2022. Evaluation Computer Vision Applications on a Drone Over 5G Connection. In *Proceedings of Drone Computer Vision over a 5G Connection (ECE 5710 SP 22).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

5G is an emerging technology that promises better performance, easier maintenance, and automation. 5G has many unique use cases, including in IoT applications which have strict timing constraints. With a smaller wavelength of 5G, it can significantly reduce the latency compared to LTE and wifi. 5G's ultra-reliable low-latency communication allows for high quality connectivity service to devices, such as drones. Enabling 5G on campuses will allow for researchers to access edge computing, which is gaining popularity. Researchers could use 5G to develop novel applications.

One such application that would benefit from the throughput and latency improvements of 5G is drone swarms. For example, HiveMind is a centralized coordination control platform for IoT swarms that seeks to be scalable and highly performant. This platform has been used to implement drone swarm applications, such as sending out drones for image recognition tasks. Since image recognition in this context is latency and throughput constrained, such an application would benefit from a transition to 5G.

Cornell, Princeton, and Stanford are currently working on a project, *Aether* to build a private open-source 5G network for campuses to advance the infrastructure for research. Our project goal is to compare the performance of 5G for communication to the drone versus the current WiFi networking system on campus. Drones have a low-latency requirement so we will be testing how the performance of image recognition for drones gets affected by the use of a 5G infrastructure.

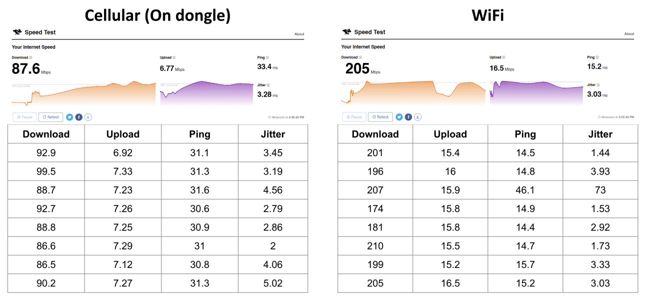
1.1 5G Background

When the radio waves are transmitting, the gas in the atmosphere absorbs millimeter waves, and the range of millimeter waves is smaller than that of microwave radiation, so the reachable range of each partition will be limited; for example, the earlier cellular network before 4G may span several kilometers, but the partition is about the size of one block in 5G. Electromagnetic waves require many antennas to cover a cell because of the difficulty passing through the walls and buildings. The advantage is that the millimeter-wave antennas are only a few inches long, which are smaller than the large antennas used in previous cellular networks, so 5G deployment may be covered by many antennas installed on buildings instead of a base station tower. Another technology used to increase the data transmission rate is massive Multiple-Input Multiple-Output (MIMO) technology. Each cell will have multiple antennas to communicate with all the devices, and each antenna will be received through an independent channel by multiple so that multiple data streams will be transmitted in parallel at the same time. To transmit all the data simultaneously and avoid transmitting the data to a farther place slower and to a closer place faster, there is a technology called beamforming. The base station computer will continuously calculate the best path for radio waves to reach each wireless device. It will organize multiple antennas to coordinate in the form of a phased array. Work to generate a correct millimeter-wave beam that reaches the device. Although 5G is extremely fast, smaller and more cells make 5G network infrastructure more expensive per square kilometer than previous cellular networks.

The research of 5G is complicated, and the cost of manufacturing and deploying devices is extremely high, but why do we still need to develop 5G for the needs of next-generation networks? The answer is that 5G systems are the need to support a variety of vertical industries such as manufacturing, automotive, healthcare, energy, and media and entertainment. To be more specific, there are three characteristics in the next-generation network. First of all, in 2021, the total number of mobile devices will be at almost 15 billion, and it is up from just over 14 billion in the previous year. The number of mobile devices is expected to reach 18.22 billion by 2025, an increase of 4.2 billion devices compared to 2020 levels. Secondly, the high quality of service requires low latency and high throughput, which also means ultra-reliable low latency communication (URLLC). Thirdly, the heterogeneous environment must be supported to allow interoperability of diverse devices such as smartphones, tablets, laptops, or different levels of latency and throughput, network type, etc.

1.2 5G Network Test

To understand how our 5G setup compares to the current Cornell WiFi network, we ran network tests, results shown in Figure 2.



Cellular (On dongle)

Download 87.6 Mbps · Upload 6.77 Mbps · Ping 33.4 ms · Jitter 3.28 ms · Measured at 2:06:26 PM

Download	Upload	Ping	Jitter
92.9	6.92	31.1	3.45
99.5	7.33	31.3	3.19
88.7	7.23	31.6	4.56
92.7	7.26	30.6	2.79
88.8	7.25	30.9	2.86
86.6	7.29	31	2
86.5	7.12	30.8	4.06
90.2	7.27	31.3	5.02

WiFi

Download 205 Mbps · Upload 16.5 Mbps · Ping 15.2 ms · Jitter 3.03 ms · Measured at 2:02:43 PM

Download	Upload	Ping	Jitter
201	15.4	14.5	1.44
196	16	14.8	3.93
207	15.9	46.1	73
174	15.8	14.9	1.53
181	15.8	14.4	2.92
210	15.5	14.7	1.73
199	15.2	15.7	3.33
205	16.5	15.2	3.03

Figure 2: Network test: Download, upload, ping, and jitter comparison between Cornell Campus WiFi and 5G Dongle

We found that the download, upload, ping, and jitter speeds of the campus WiFi is actually better than those of the 5G dongle connection's. We suspect that this is because Cornell has a very robust WiFi network and routers in close proximity, whereas 5G is not yet implemented in Ithaca, NY so it takes much longer to connect to 5G.

2 RELATED WORK

We did this project to support the much larger project, *Aether*. Aether is a project being conducted by Cornell, Princeton, and Stanford to implement a Private 4G/5G Connected Edge Cloud that can be used for next-generation campus research. A drone application that has been used with coordination with Aether is the project *HiveMind*. *HiveMind* is a Scalable and Serverless Coordination Control platform for UAV Swarms conducted by Prof. Delimitrou's group in Cornell University *https://arxiv.org/pdf/2002.01419.pdf*. They were able to develop this by using the edge compute nodes of the *Aether* project.

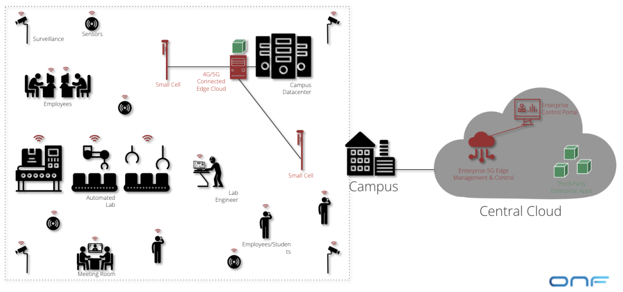


Figure 3: Aether Connected Edge Cloud

To further explore the benefits and limitations of a campus 5G network, we built on top of this work to conduct a thorough analysis of how 5G and WiFi connections compare for applications that involve computer vision on drones.

3 DESIGN

Broadly, our design seeks to run and evaluate *representative* Computer Vision drone applications, comparing *edge* computation and *remote* computation network configurations, evaluated based on *end-to-end latency.*

3.1 Computer Vision Applications

We selected representative Computer Vision applications in our experiments to emulate potential real-world use cases. We designed our code for testing and our experiments so that the built-in camera in the Parrot AR Drone 2.0 sends a VideoStream over a TCP connection that we are able to see using NodeJS HTTPS connection.

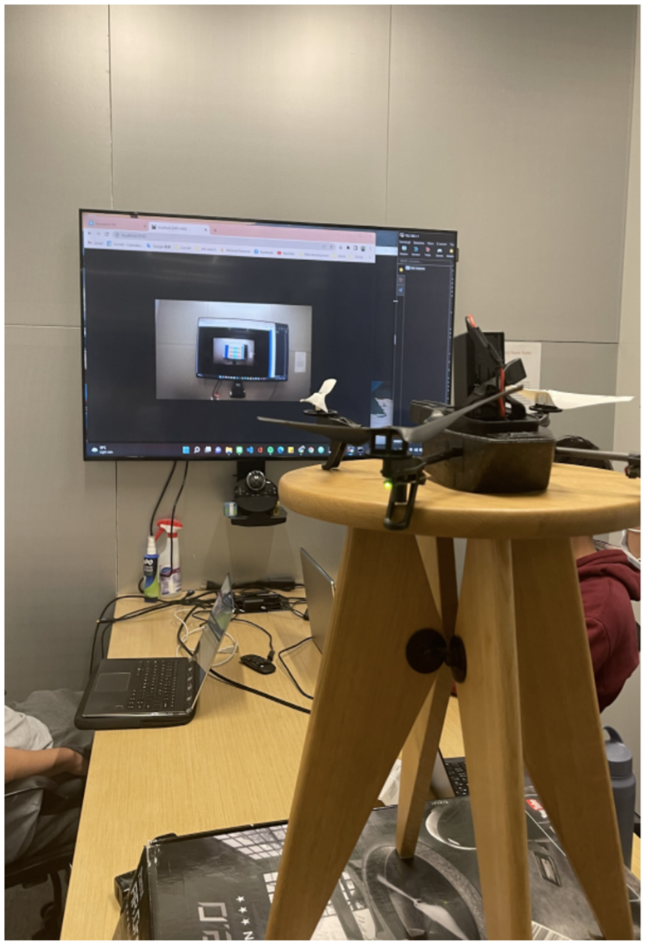


Figure 4: Laptop Connected to Monitor to Display the HTTPS Stream of the Drone's Video

Stripe Detection Given a video stream, determine if the video contains a set of distinct and alternating stripes based on selected color choices. This application is useful for drones for determining a target destination, for example the location for the drone to land.

Face Detection Given an image stream, determine if the images contain human faces. Some useful applications of facial detection include estimating crowd size, tracking people's movements within a field, and returning to a person's location.

3.2 Network Configuration

We run representative drone applications on a set of network configurations. They are as follows:

Local (Edge) Compute Compute is done locally on the drone, using the drone's hardware.

Compute Over 5G The drone sends the appropriate data (either Video or Image stream) over 5G network to a remote server. The remote server then runs the computation.

Compute over Drone WiFi The drone sends the appropriate data (either Video or Image stream) over the drone's local WiFi network to a remote server to run the computation.

3.3 Evaluation Design

Our evaluation metrics should be most pertinent to the end user. Since drone applications run in real-time with changing data as the drone flies around, there are relatively tight latency contraints. Thus, *end-to-end latency* is the main evaluation metric chosen. There are two main components to the end-to-end latency.

Compute Latency is thee latency incurred to the computation of the input. This varies based on the network configuration, with the remote server having much more compute resources than is available on the drone.

Network Latency is the latency incurred due to sending data over the network. This varies based on the size of the data as well as the type of network (5G or Wi-Fi). When compute is done locally on the drone, the network latency is zero. When compute is done remotely, the network latency depends on network qualities as well as the data qualities.

The end-to-end latency is directly affected by the network configuration and we are interested in the tradeoffs. We also consider the *variation* of the end-to-end latency for the different network configurations.

4 IMPLEMENTATION

Implementing our design involves selecting the hardware to use, setting up network communication, and writing and running the software applications.

4.1 Hardware Selection

The hardware of our setup include the selection of drone, network adapter, and remote server.

Parrot AR Drone 2.0. This is the drone selected to run the Computer Vision applications. We chose this model because it is commonely ued as well as highly programmable, running BusyBox, a stripped down version of Linux, that can be remotely accessed. This drone also has built-in WiFi capabilities, which are evaulated.

SERCOMM Adventure Wingle. This is the USB dongle used to facilitate the 5G connection. This Wingle is used to connect to the private 5G network set up as part of Cornell's Aether project, as mentioned in the Section 1.

Remote Server. We simply use our personal laptops as the remote server. This is because we do not have access to commercial remote servers with the security access to allow us to set up the direct communication channel with the drone. An area of future work would be evaluation on a commercial remote server, which would have more compute capabilities.

4.2 Communicating With The Drone

Setting up the proper network communication with the drone proved to be a bit of an implementation challenge.

We modify the Linux configuration to connect to a WiFi hotspot instead of acting as an access point. We also install NodeJS on the drone, giving us the ability to execute Javascript programs locally on the drone.

Our drone needs to be connected to a hotspot rather than your computer. To do this, we need to Telnet into it and reconfigure the WiFi.

4.2.1 Telnetting into the drone. On boot, the drone exposes a Wi-Fi network that we can connect to on our laptops. The default IP address the drone assigns itself is 192.168.1.1. Thus, we can open up a terminal window on our laptops and type *telnet* 192.168.1.1. This command opens a telnet session with our drone, with success indicated by the "Busybox" prompt that appears. From there, we can execute standard Linux commands on the drone remotely from our laptops.

4.2.2 Reconfigure Drone Network Connection. When we are doing the test of Stripe Detection and Face Detection, we will send the video stream and pictures over 5G dongle. Ideally, we would connect our laptops over 5G ethernet via USB on the dongle and have a separate 5G dongle connected via USB on the drone. However, perhaps due to hardware or OS limitations on the drone, we were unable to get the dongle's ethernet interface running locally on the drone. The workaround was to plug the dongle into the wall and have it expose a WiFi connection that the drone may connect to. Thus, we had to reconfigure our drone's network settings to connect to 5G dongle's WiFi.

To do this, the WiFi network from the 5G dongle should not have a password and should nor use any encryption methods like WPA2 or otherwise. Then, we run the command "killall udhcpd; iwconfig ath0 mode managed essid drone; /sbin/udhcpc -i ath0; sleep 5; route add default gw 192.168.1.1;" This will kill all the WiFi processes and reconfigure the WiFi drivers to connect to our "drone" hotspot (with the network SSID "drone") rather than acting as an access point.

After we reconfigure the WiFi, the drone has automatically assigned by a new IP address, so we will use Angry IP Scanner to find out what the IP is. When we find out the IP address that look like the drone, we try to telnet into it. If it lets us in and shows the busybox prompt, we're in!

4.3 Running Applications on the Drone

For all of our configurations, the Drone must run some application. When compute is offloaded to a remote server, the application is simply forwarded the video or image streams over the network. But when the compute is done locally, we need to configure the drone to be able to run more complex applications.

This proved to be quite a challenge. Because the drone used a stripped-down version of the Linux, we couldn't directly run any JavaScript or Python applications on the drone. We even couldn't install any programming language into the drone by the method we do in our regular machines.

4.3.1 Installing NodeJS. After a couple of weeks of research, we found a way to install NodeJS on the drone. We download the stripped-down version of the NodeJS binary code that was especially for the drone's hardware specification. Also, there are other directories of node modules in this repository called node_modules. This folder includes all the dependencies required to run the drone and image recognition. These modules allow us to interact with the drone over the internet. Instead of using an FTP client to connect to our drone, we use a USB flash drive and navigate to the /data/video/usb directory. Then copy the contents of NodeJs binary code and all the modules into the drone.

4.3.2 Running NodeJS Application. In our telnet session, we can run the command ./bin/node –expose_gc drone.js (drone.js is the application we are running).

The –expose_gc enables garbage collection within the NodeJs process. This is essential because, due to the drone's hardware limitations, with NodeJS running, the AR Drone only has about 3% memory left.

4.4 Application Implementation

The two applications we ran, either locally on the drone or on a remote server are facial detection and stripe detection. Stripe Detection wA implemented in Javascript and run as NodeJS application. Facial Detection was implemented in Python using the OpenCV library. Specific implementation details are described briefly:

Stripe Detection was implemented by sweeping over the pixels of the input images. The colors of each small region of the image is recorded and compared to determine if the target stripe pattern is detected.

Facial Detection was implemented using a Haar Cascade Classifer. The pretrained data is provided as an XML file to the classifer. This application used the provided Cascade Classifer class in OpenCV.

5 METHODOLOGY

Our project tests the latency for different packet sizes between signal transmission over a 5G dongle and a drone's WiFi. Then we do two kinds of computer vision applications to evaluate the performance between a 5G dongle connection and without a 5G dongle connection. The first application is stripe detection. Then the second application is face detection. In this section, we will explain how we measure the data.

5.1 Measurement of Different Packet Size

We first did a full latency test comparing communication from our laptops to the drone using the drone's WiFi network vs. using the 5G dongle. To conduct this experiment we sent pings with different packet sizes, first over a WiFi connection, then over the 5G Router connection. To collect the data, we looked at the total Round-Trip Time each packet size took and recorded the mean over 5 pings for each packet size.

5.2 Stripe Detection

We pieced all the code together refer to the repository and finally successfully executed the code of stripe detection on the drone. After the stripe detection works on the drone, we divide the test into three different network conditions.

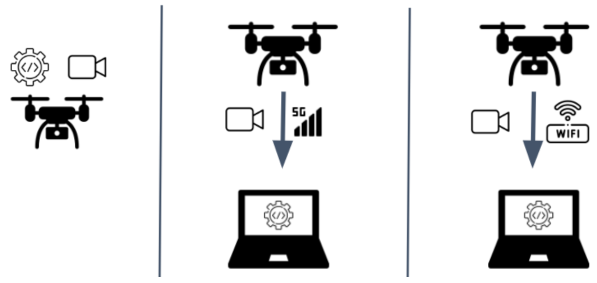


Figure 5: Stripe Detection, (Left) stripe detection locally on drone, (Middle) stream video over 5G dongle; stripe detection on laptop, (Right) stream video over WiFi; stripe detection on laptop

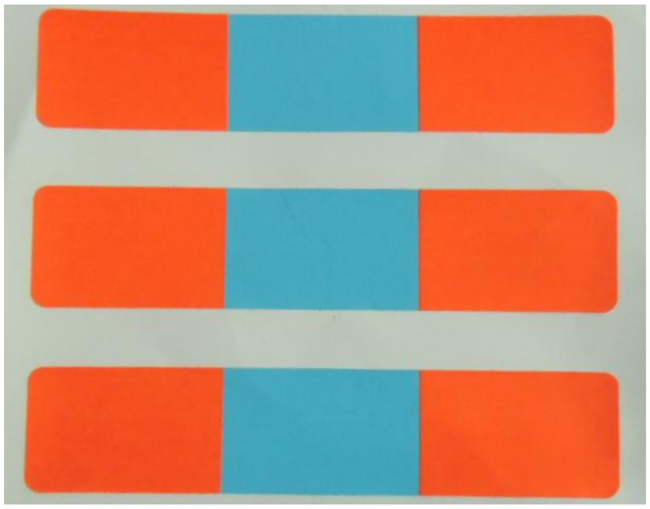


Figure 6: Orange and blue stripe use for stripe detection

The first configuration is to run the stripe detection directly on the drone. In this condition we can get the latency only provided by the drone's computing power. We first telnet into the drone by the drone's WiFi from our laptop, and use the module we mentioned above to execute the stripe detection script we wrote. After execution, we point the drone's camera at the blue and orange stripe that showed in Figure 4 and Figure 5. The drone and stripes are placed such as detection should be immediate. When the drone successfully detects the stripe, our terminal will show "Detected". To measure the latency of the computing power, we wrote the time

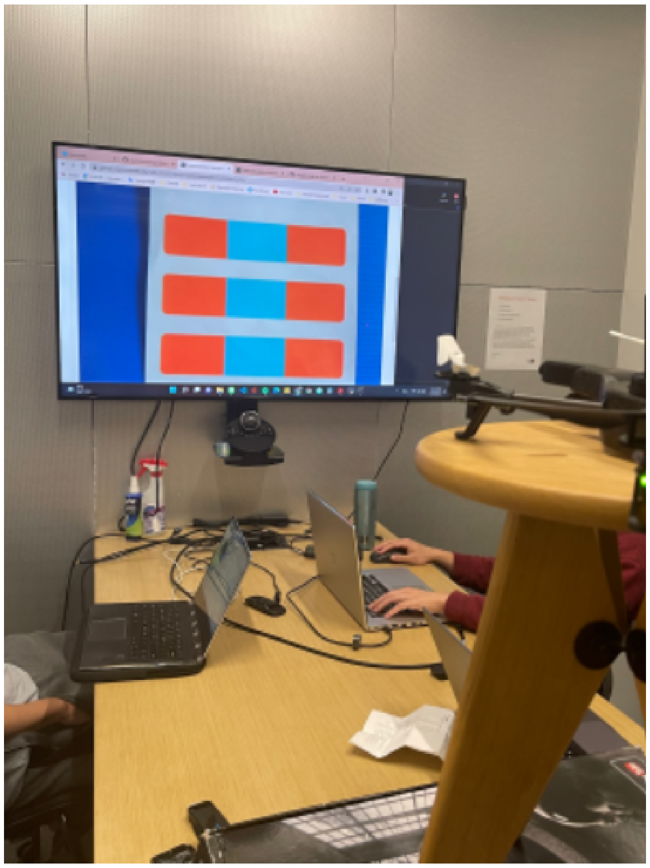


Figure 7: Stripe detection setup

function into our JavaScript program. We subtract the time when the drone take a image from the detected time, and print it out on the terminal. We run 10 trials and record the latency data.

The second configuration is to transmit the image through 5G dongle's WiFi and run the stripe detection on the laptop. The third configuration is to transmit the image through drone's WiFi and run the stripe detection on the laptop. In both configuration, we run the stripe detection on our laptop, which means the total latency equals to "laptop compute latency + network transmit latency". The difference between second and third configuration is that second configuration, we transmit the video stream over 5G dongle. The third configuration, we transmit the video stream over local WiFi from the drone. We modify the programs to include timers to measure latency.

5.3 Face Detection

In the face detection, we divided the detection into two configurations, as shown in Figure 6. The first configuration is to send the picture over 5G dongle and do the face detection on laptop. The second condition is to send pictures over drone's WiFi and do the face detection on laptop. The only difference between two configuration is that we send the image over different signal. Note that the local compute configuration is not present for this application. This is due to issues with running OpenCV locally on the drone. However, this configuration still provides valuable insight into the differences between offloading over WiFi and offloading over 5G.

We first start the applications on the drone to stream out images. We then run the Face Detection Python program on our laptops, with the appropriate timers to measure latency. As with the Stripe Detection, we place the drone pointing such that face detection should be immediate. We run 10 trials and record the latency results for each configuration.

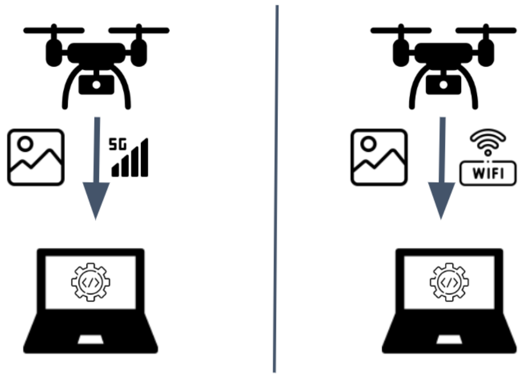


Figure 8: Face Detection Network Configurations. (Left) Stream Images Over 5G; Face Detection on Laptop. (Right) Stream Images Over Drone's WiFi; Face Detection on Laptop

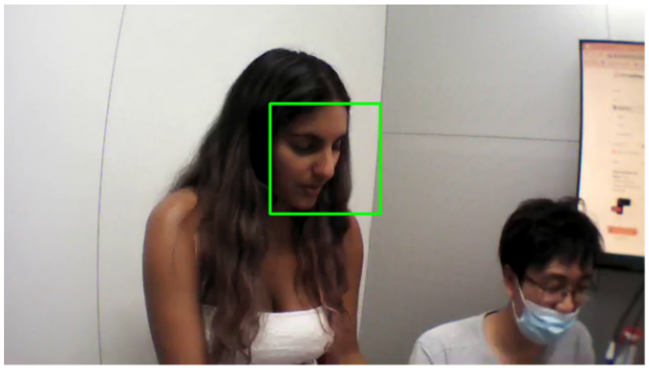


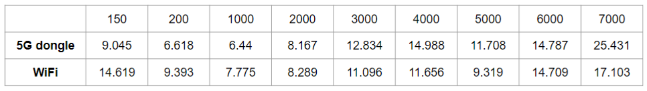
Figure 9: Face Detection Output Image

6 EVALUATION

In this section, we will perform multiple measurements of the above three tests, including latency of different packet sizes, stripe detection latency, and face detection. Then we will explain the phenomenon corresponding to the measured data. According to our on-site measurement to see the situation, we will give a reasonable explanation.

6.1 Latency to Different Packet Sizes

After sweeping a series of packet sizes up to the maximum that the pings would allow us to do, we found the latencies for the packet range 100 MB - 7000 MB to are similar between WiFi and 5G Dongle.



	150	200	1000	2000	3000	4000	5000	6000	7000
5G dongle	9.045	6.618	6.44	8.167	12.834	14.988	11.708	14.787	25.431
WiFi	14.619	9.393	7.775	8.289	11.096	11.656	9.319	14.709	17.103

Figure 10: Latency of different packet sizes

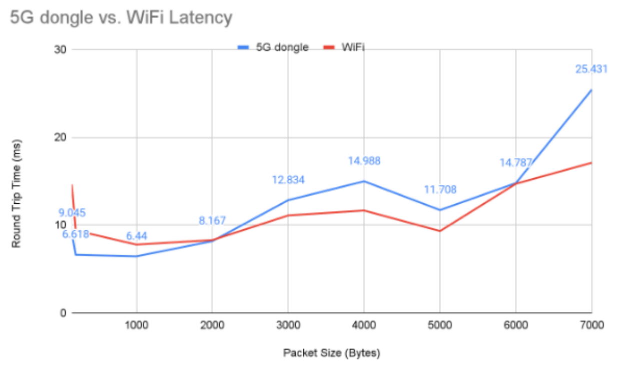


Figure 11: The figure of different packet sizes

The graph in Figure 10 shows the expected increase in Round-Trip Time over an increase in packet size for both the 5G and WiFi connections. A noteable finding is that the trends over the packet size increase in a similar way until the last datapoint of 7000 MB packet size at which point the 5G dongle is significantly slower than the WiFi.

The conclusion we can extract from this data is that for fairly small packet sizes there is no difference in latency for communication to the drone from the laptop and from the drone to the laptop.

6.2 Stripe Detection Latency

We tested 10 times each for the three configurations we mentioned in the previous sections. The results are shown in table form in Figure 10 and in chart form in Figure 11.

First, we look at the difference between drone WiFi (red dots) and 5G (yellow dots) . We observe that the average latency of WiFi is relatively high and the variability between trials is very large as well. We believe this may be because drone WiFi is relatively unstable due to the hardware limitations leading to latency fluctuations as well as due to the network speed difference when transmitting data.

Second, we compare on-drone detection (blue dots) and laptop detection when transmitting data over 5G dongle (yellow dots). On-drone detection only incurs a compute latency while transmitting over 5G incurs both a compute and network transmission latency. Both of the latency is very stable and fluctuate within a certain range. However, we observe that the average latency on the laptop through the 5G dongle is much lower than on drone. We believe this is because the computing power of the laptop is much faster compared to on drone computing power, leading to higher compute latency on the drone. Although there is additional network transmission latency on the 5G configuration, this latency is still lower than the longer compute latency from running locally on drone. This result follows some industry trends, where nowadays many products choose to upload the data to the cloud, and then send it back to the device after the calculation is completed because it is still faster than computing directly by the edge device.

	1	2	3	4	5	6	7	8	9	10
On drone	117	124	144	113	120	119	138	143	118	137
WiFi	82	271	310	19	267	27	50	304	741	85
5G	45	24	70	29	17	57	70	38	53	60

Figure 12: Latency Measurements of Three Configurations Running Stripe Detection

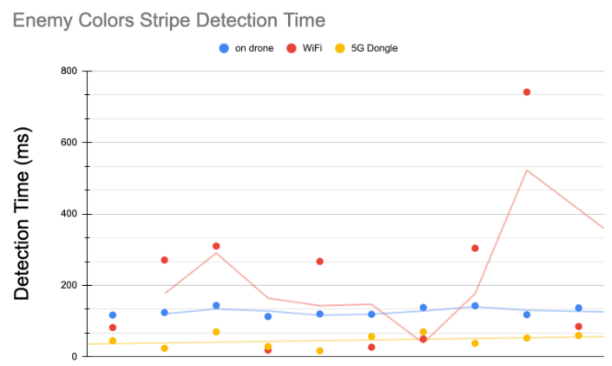


Figure 13: Stripe Detection Latency Line Chart

6.3 Face Detection Latency

We ran 10 trials for each of the two configurations described in previous sections. The results are shown in table form in Figure 12 and in chart form in Figure 13.

We observe no significant differences in latency between the 5G configuration and the WiFi configuration. This result deviates from the previous Stripe Detection result, where 5G performed much better. We attribute this differences between the two applications. First, Face Detection is more computationally complex, meaning more time is spent on computation. Second, in our setup, Face Detection is streaming images over network instead of video, meaning the amounts of data sent over network is less, resulting in lower network latency.

	1	2	3	4	5	6	7
WiFi	117.106	139.509	130.348	122.589	125.104	122.199	116.171
5G	112.629	115.414	119.292	122.605	116.94	122.152	121.406

Figure 14: Latency Measurements of Three Configurations Running Facial Detection

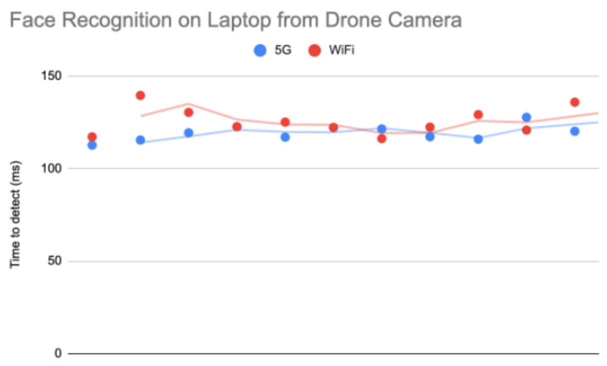


Figure 15: Face Detection Latency Line Chart

7 CONCLUSIONS

In this project, we want to compare the latency performance on the drone over 5G dongle, drone WiFi and local campus WiFi. We evaluated the latency performance through stripe detection and face detection.

In stripe detection, 5G dongle has the lowest latency performance compared to the drone WiFi itself and local on-drone computation. We also find that the drone's WiFi is more variable than 5G. This provides evidence that for this application, there is benefit in offloading compute to a more powerful remote server, as the additional network latency is negligible. Further, this result provides evidence that 5G connection provides benefit over WiFi in terms of latency and latency variability.

In the face detection experiment, we came up with the similar latency results over WiFi and 5G. Compared to the stripe detection, face detection uses pictures whereas stripe detection uses video steam. Thus, compared to stripe detection, our face detection implementation is more compute heavy application with less data sent over network.

These two experiments generally provide evidence that compute-heavy applications will benefit from offloading compute to a remote server over 5G. Data-heavy applications would benefit most from offloading over 5G instead of WiFi. Compute-heavy and data-heavy applications would likely benefit most from 5G offloading, but further work is needed to make that conclusion more definitively. However, our results do show no detriment in performance between WiFi and 5G. And recall that 5G has many benefits over WiFi including flexibility in connection area especially in outdoor settings. These benefits should be considered as well when comparing 5G and WiFI.

8 FUTURE WORK

At present, we face a number of limitations and restrictions in its evaluation and implementation. There is more work to do and experiment in the future on the drone.

Connect 5G Dongle directly to drone. Currently we are unable to connect the cellular network directly to the drone. This is perhaps due to a hardware or OS limitation on the drone. Thus, this limited our evaluation about performance and latency comparisons. If we can connect it directly to drone, we are able to compare the latency in different environments with more precision.

Run OpenCV applications directly on drone. In our experiments we are only able to run OpenCV program on our laptops for face detection. We are seeking ways to run the program directly on the drone to truly compare the latency difference and computing power between the drone and our laptops for OpenCV applications.

Evaluate more compute-heavy applications. In the stripe detection and face detection, we only experimented on relatively simple objects. In order to explore how well the drone can handle the computer vision related tasks without affecting the performance greatly. Also, we can evaluate the performance and latency difference on both our laptops and on the drone with such compute-heavy tasks.

Run on more powerful server. In terms of real comparisons between the 5G Dongle, campus WiFi and the local drone WiFi, we wish to run our drones on a more powerful server over the Dongle. This enables us to evaluate the power and the latency of 5G implementation.

Test in outside environment. Since drone are mostly deployed in the outside environment, we also want to test the drone outside the laboratory. A typical wireless router has a outdoor range of 300 feet, this may affect the latency performance if the drone is away from the router. Thus, network connection over 5G dongle can be a more suitable and stable solution. Testing outside can truly reflect the performance difference on the drone between 5G dongle and WiFi.

Compare with commercially-available WiFi and 5G solutions to evaluate capabilities and cost. Since the campus WiFi has very high bandwidth and more robust, the experiment result may have a huge difference if we run the drone over commercial WiFi with smaller bandwidth and throughput. With this comparison, 5G may outperform the commercial WiFi in households. We think that this comparison is important since not every environment has such high bandwidth of network access.

ACKNOWLEDGMENTS

We thank our course lecturer, professor Christina Delimitrou for suggestions and guides with to project. This work was held and experimented in CSL laboratory.

A SOURCE CODE

The source code for the applications we ran on the drone for our experiments can be found at this GitHub repo: https://github.coecis.cornell.edu/aa2224/drone5g