Drone Photogrammetry Array Scoping Report

Project Background and Overview

Market and Need

International agencies such as the World Bank, governments, industrial companies, and insurers make extensive use of aerial imagery, and pay substantial amounts for it. In Africa, these actors normally hire expensive consultants—usually foreign—using expensive, heavyweight equipment.

High-resolution visual-band satellite imagery (which costs on the order of $25/km² to purchase) tops out at ~30cm resolution. Recent imagery is often unavailable, even for paying customers (clouds, orbits, etc).

Cost and Coverage

Drone imagery can easily achieve 5cm resolution; currently this costs between $300 and $900/km². Agencies often purchase this kind of imagery every few years for large cities.

A $20,000 fixed-wing mapping drone from one the two Swiss manufacturers that dominate that market will cover ~20km² at a 5cm Ground Sampling Distance (GSD).

A $1,000 prosumer-grade quadcopter can—operated by a skilled mapper—cover ~2km² at a similar GSD and quality level (though it requires surveyed Ground Control Points to match the georeferencing precision).

If the operator cost is high (high-income country wages, hundreds of dollars per person-day) the expensive drone is the only feasible option. If the operator cost is based on low-income country wages (~$10-$40 per person-day in sub-Saharan Africa), the inexpensive drone becomes much more cost-effective.

Quality of Imagery and 3D

Most professional mapping drones acquire only a single camera angle (nadir or nearly so, 80° is common). The resulting imagery looks fine in an orthometric projection (straight down) but doesn't do particularly well at creating Digital Elevation Models. Adding shots at oblique angles greatly increases the quality of the 3D terrain reconstruction, as well as revealing features through partial tree canopy. Some professional multirotor drones are capable of moving the camera on a gimbal to capture multiple angles in the same flight (though they typically must slow down to do so), but most drones must fly multiple passes over the same area to acquire multiple angles. A few (mostly very expensive) professional mapping drones have multiple imaging sensors at different angles.

Any drone with multiple sensors has an immediate advantage in capturing multiple angles in a single flight, greatly increasing the quality and value of the resulting imagery and the amount of land that can be imaged in a given time period.

Weight

Many countries are cautious of drones in their territory. The war in Ukraine has only exacerbated these concerns. Drones under the common regulatory limit of 250g are generally less strictly regulated.

Also, it's inherently difficult to weaponize a 249g drone; keeping the weight low is not only good for regulatory compliance but also to avoid our equipment being used as a weapon.

A credible OEM in China has proposed to design and build a quadrotor drone for us including everything except the camera array, coming in at 189g. This leaves 60g for a camera array.

Conclusion and Scope

A relatively inexpensive drone capable of capturing high-quality imagery to support aerial mapping would empower local people in low-income countries to create high-quality aerial 3D imagery using low-cost, lightweight equipment. The goal of this scoping project is to determine the feasibility of a payload that meets cost, weight, and size requirements for such a drone, and evaluate design tradeoffs as applicable.

Cameras and Modules

Overview

Smartphones are far and away the biggest driver of the wide availability of high-quality and high-resolution camera modules. As low light performance and speed become more important (in addition to marketing), camera sensors seem to be increasing in raw pixel count. The advertised pixel count are actually Bayer cells, which need to be demosaiced down the pipeline to generate a true color image. In addition, these larger sensors (48MP, 50MP, 64MP, and 108MP) also support on-chip binning/summing of adjacent pixels into macro-cells (usually 2×2, but 3×3 in the case of the 108MP). The benefits of this include lower data demands downstream and increased light sensitivity. Vendors I spoke to indicated 20-24MP sensors are being deprecated as a result.

Sensor Manufacturers

Sony, Samsung, and OmniVision seem to be the main sensor manufacturers being used in this market. While there are other vendors, the module suppliers we spoke with seem to mostly work with some combination of those big three vendors.

Off-The-Shelf Modules

We found a few different vendors based in Shenzhen who sell modules based around current smartphone sensors: Sincere Information Technology, ZS Semiconductor, and Meiying Optical. Sincere and ZA were willing to provide quotes up to 1k+ units and claimed to be able to supply them in 10s of thousands, so module availability is possible in sub smartphone quantities. Arducam's distributor (UCTronics) also provided me with quotes for 1ku and 10ku. The FOV of the lenses are not all identical, but in a fairly small range that meets our 60 degree HFOV requirement.

Resolution and Pixel Size

Sensor/Module	Resolution (MP)	Pixel Size (um^2)	Binning	Binned Pixel Size (um^2)
IMX686K-S	64	0.64	2×2	2.56

Sensor/Module	Resolution (MP)	Pixel Size (um^2)	Binning	Binned Pixel Size (um^2)
IMX586	48	0.64	2×2	2.56
IMX766	50	1	2×2	4
S5KHM2SP03	108	0.49	3×3	4.41
S5KGM2SP	48	0.64	2×2	2.56
OV64A40	64	1	2×2	4

Based on the above, it appears that 0.64 um^2 pixels are fairly common, 1 um^2 is probably set to be the next standard to meet, and 0.49 um^2 is really only an option for ultra high res sensors and it is made up for by binning the pixels aggressively (3×3 in that case).

Lens Options

While the vendors I spoke with provided a fair amount of information about their camera modules and the sensors used, they have not responded to any of my lines of inquiry around custom lenses. It's possible that in sufficient quantities they would take the lens request more seriously, but the stock lenses appear to be adequate for this application. The project requirement was stated to be 60 degrees HFOV and the stock lenses range from 65 to 73 degrees HFOV. I am confident that custom options exist (Arducam definitely offers them), but they have been reluctant to discuss them with me without more of a purchasing or sampling commitment.

Module Cost

The prices I was quoted for 1k+ units was in the ballpark of $19-$25 per module. It didn't seem like there are really tiers of these modules (e.g. low, medium, high) at 1x, 2x, 3x cost, for example. Likely, because these are for smartphones made in the millions, there is an appetite for a little variance in the price for these modules, but little to no market for something 1.5-2x the "standard" price. A minor exception is the Arducam module based on the OV64A40, which is ~$33 at 1ku, but only $25 at 10ku. It is unclear how a module with a custom lens would be priced. Unfortunately, US-based camera modules (from Leopard Imaging, for example) are priced closer to $300 for the same sensors in single quantities (about 10x

those of the Chinese companies) but with a larger lens. I would not expect them to be able to supply modules in line with the cost targets of this project.

Ground Sampling Distance (GSD)

GSD (effectively the pixel size on the ground) is a key specification for this system, as it dictates how precise the maps can be. It must be sufficient for high quality mapping at a reduced cost compared to alternatives in order for this project to be viable. 5cm was given as a desired target at an altitude between 100m and 150m.

Due to the very high resolution of these modules and the reasonable FOV of the stock lenses, GSD itself is not necessarily an issue with any of them. At full resolution (e.g. no summing of adjacent pixels), all surveyed modules and sensors can capture at well below 5cm GSD between 150m and 100m altitude except in the most extreme case of the oblique camera (45 degree angle) at 150m altitude, in which case half of them just barely miss 5cm GSD. Based on the high quality of imagery collected by Ivan with a 48MP camera binned down to 12MP, it seems that 12-16MP binned sensing is probably not a problem, even if it misses exactly “5cm” by a bit. Since the oblique sensing provides data that is otherwise not available by ANY other source, even greater than 5cm GSD will be very valuable.

Module Weight

Most vendors quoted me around 1.6g per module, except for the Arducam module which was around 3g. This will mostly be driven by the lens choice, which can almost certainly be customized for the application at sufficient order quantities. Depending on how other elements of the payload shake out in terms of weight it would make a lot of sense to invest any spare grams into a larger fixed-focus lens.

Data Rates

Data ingestion is not trivial for this project. Even at low frame rates (assuming RAW8 or RAW10), we are pushing up against the 2.5 Gbps limit for single-lane MIPI. On the other side of that bus will eventually be a storage medium that will have a hard time keeping up with 3x that MIPI rate.

A few examples

- 64MP RAW10 @ 2 fps = 1.28 Gbps
    - x3 = 3.8 Gbps
- 48MP RAW10 @ 2 fps = 960 Mbps
    - x3 = 2.88 Gbps
- 16MP (binned) RAW10 @ 2 fps = 320 Mbps
    - x3 = 960 Mbps

As long as the resolution/GSD is sufficient, binned pixels will yield better sensitivity than the other two and require far less total data storage and relaxed throughput requirements. Assuming the line scan rate is fairly consistent between sensors, binned sensing will also result in less rolling shutter effect since the whole image will be able to be read out faster.

For more on data rates, processing, and throughput, see Technical Details.

Recommendations

While the sensors and modules covered here are surely not exhaustive, they provide a reasonable snapshot where the industry is right now, which is to say “48-64MP sensors and modules built around them are widely available and reasonably priced”. Based on what has turned up during this scoping task and the original asks, my recommendations are as follows.

First Choice:

Source the Arducam B0483 “OwlSight” (or equivalent in fixed lens) module based around the **OV64A40** (64MP). It has 1um pixels (the largest I have come across thus far) and the 64MP resolution binned down 16MP yields nearly the highest active area per pixel. The only one surpassing it by ~10% active area is the S5KHM2SP03 (108MP binned down to 12MP), but then you lose 25% of your pixels. I doubt that tradeoff is worth it, especially considering this is only supposed to be flown during the daytime anyway. It is a little bit more expensive at lower quantities, but at 3 modules per drone, it should be a lot easier to hit higher order volumes than most application and will likely deliver the best overall image quality. A module with a fixed lens can probably bring down the cost as well. Harder to quantify, but worth considering, is that Arducam makes modules for the

Raspberry Pi and has an active forum and a support team. We won't be using a Raspberry Pi of course, but if we run into technical challenges running or integrating a module made by them, we are more likely to find answers in the community or from them than we are from Samsung, Sony, or the more wholesale distributors that comprise the other sources I found. The only potential drawback here is that this module might weigh a bit more than the others, but since image quality is really key here I would think it would be worth prioritizing the camera quality if the grams are available.

Second Choice:

This is a bit of a toss-up between an **IMX686** (64MP) based module, an **IMX766** (50MP) based module, and an **S5KHM2SP03** (108MP) based module. The IMX686 has the benefit of 64MP→16MP binning, but the pixel active area is smaller. The IMX766 and S5KHM2SP03 both bin down to 12MP with a large total pixel size. Based on drone data collected from 12MP cameras, perhaps that is sufficient and worth it to have higher sensitivity. On the other hand, daytime-only applications may make the higher resolution more attractive since light sensitivity is less likely to be a limiting factor. Purely considering the modules presented (and ignoring custom lens options), the IMX686K-S (64MP) module lens also has a narrower FOV, giving an additional edge in GSD over the other two. If the first choice was passed over due to cost concerns, perhaps that could be the deciding factor, but it ultimately comes down to a choice between 16MP @ 2.56um^2 pixels or 12MP @ ~4um^2 pixels.

Third Choice:

Everything else. I don't think a 48MP→12MP sensor with a smaller pixel size is worth it unless cost is a major factor. Anything smaller than 48MP will soon be obsolete if it is not already, and any arrays of 12-20MP will almost certainly have a far inferior pixel size compared to the modules above.

Cameras and Modules Appendix: Camera Module Specs and GSD Calculations

https://docs.google.com/spreadsheets/d/1pRgM2yN30DYYLLO2ghkNGgDLZ0fSQqfIQMXTUQW083Q/edit?usp=drive_web

Data Ingestion, Processing, and Storage

Overview

The core requirements of the processing system are the following:

- Ingest 3 camera feeds
- Store the camera data in a lossless raw format
- Retrieve the data for upload to the mapping service
- Low cost in production
- Quick time to market
- Light weight

Nice-to-have requirements of the system are the following:

- OpenSource as far as possible but can be compromised on

There are two large categories of processing solutions available.

1. Dedicated, proprietary, ASICs and micro-controllers. Those range from low cost chips intended for the use with dashcams, runcams and GoPro type devices, to high power devices designed for the use with edge computing, like car lane keep assist or even self driving.
2. FPGA solutions. Those also have a range from big and flexible, through dedicated parts for camera data feed processing to very small ones.

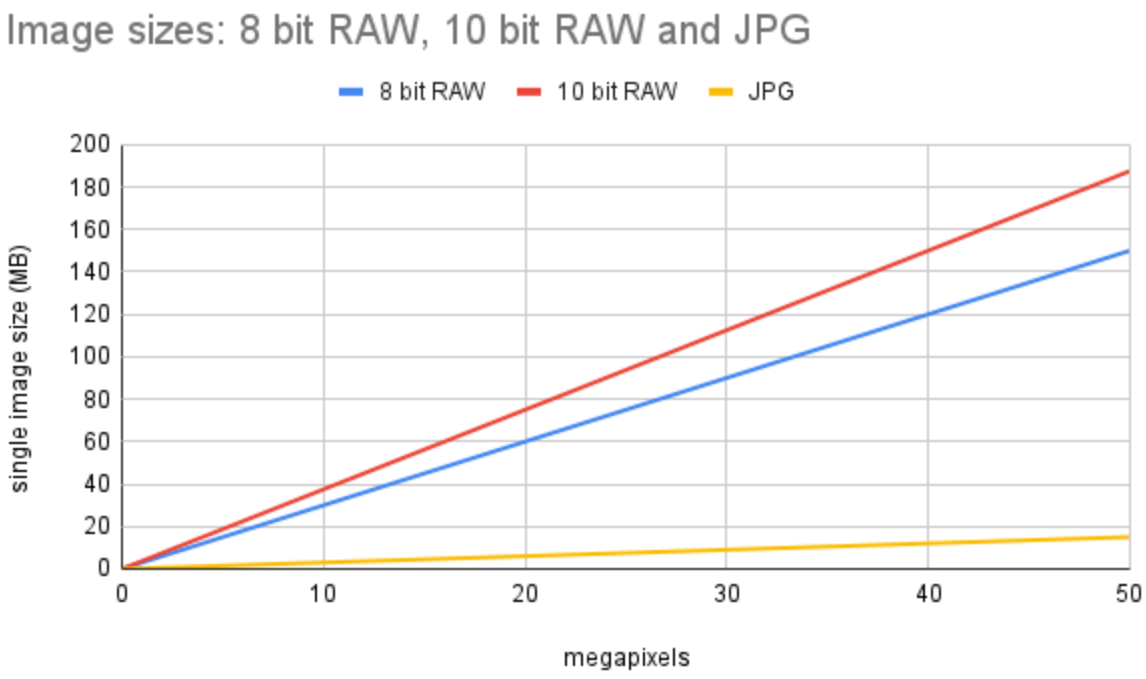
Technical details

Camera Interface

The most common interface available on camera modules is MIPI-CSI-2. It is pretty clear that this is the interface we will have to be able to ingest. The MIPI-CSI-2 interface has multiple versions and a varying scale of bandwidth. A common capability is up to 4 lanes at 2.5Gbps each. Ideally the system we choose should be able to handle simultaneous ingest of 3 cameras using 4 lanes of 2.5Gbps.

Storage

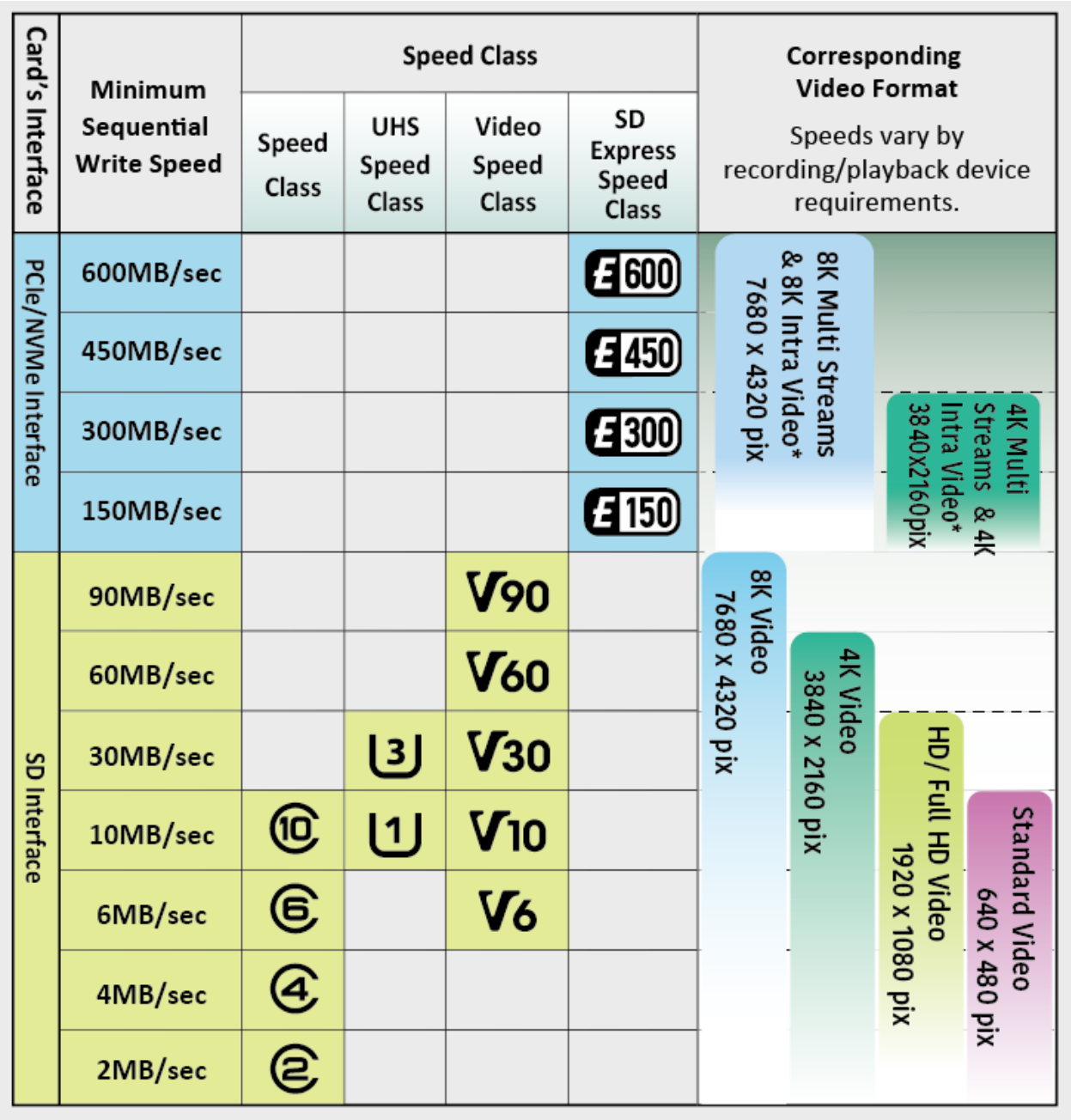
There are multiple approaches to storage. Initially the assumption was the use SD-Cards. SD-Cards are a common way to handle image data. The issue with SD-Cards is that there are many kinds of them, and it is hard to assure a reliable source with bandwidth capabilities needed by the system.



Size of a single capture (3 stills from 3 cameras).

We need to store all three camera sources to the storage medium. Considering the recommended Arducam camera (see Camera Sensors and Modules) we have to store 16MPixel data RAW10 Bayer at 2FPS. This results in about 114MByte/s write

rate. Unless we implement some kind of compression this will require one of the very new NVMe/PCIe interface cards, and result in a cascade of requirements.



There are multiple alternatives:

- EMMC → it is essentially an SD-Card on a chip. This will allow for much tighter control of how the data is stored on the medium, as well as remove the problem of users buying the wrong card. Interfacing to the medium can be

implemented using regular SD-Card interfaces. The performance of the part can be evaluated before production assuring we can meet our bandwidth requirements.

- UFS → Universal Flash Storage, a new JEDEC standard. It combines the advantages of SSD and SD-Card interface. It uses a LVDS interface. It is faster than EMMC and is lower power consumption than SSD type storage. There are commercial FPGA gateware cores available on the market. MCU support does not seem to exist yet.
- raw NAND → This is just the raw memory and needs to be managed. At volume it can be the lowest cost solution, but requires the most amount of engineering to make sure it works reliably and does not wear out the chips quickly.
- SSD → Is not really an option here, due to mechanical form factor.

Storage Pricing

Type	128GB	256GB	512GB	1TB	Physical Dimensions	Notes
Micro SD-Card	$30.34	$40.34	$80.34	$170.34	13.5mm x 15.0mm	SanDisk Extreme Pro, "up to" 140MB/s write speed. 90MB/s for 128GB model
EMMC	$19.60	$47.73	$85.94	N/A	11.5mm x 13.0mm	
UFS	$34.22	$58.18	$62.46	$124.99	11.5mm x 13.0mm	

The above table shows the prices of the solution depending on the amount of memory. Micro-SD card price includes the PCB socket price of $0.35 on top of the SanDisk Extreme Pro cost.

Data Offload

If a removable SD-Card is chosen, the offload can be similar to a regular camera. Remove the SD card, and use some kind of a dongle or slot on laptop or phone to download the files. We are then forced to store the data in exFAT format to assure interoperability.

The alternative solution is a USB port on the camera itself. This allows for the use of built in memory instead of removable memory. It offers flexibility to the way the data is stored and then presented to the offload device. This means we can store the data in a custom format on our storage device and present it as a FAT

filesystem to the host. We can also expose it as a custom USB device that can be accessed using WebUSB directly. It does require some engineering and development.

Considering that we are storing the data in a RAW format we will have a lot of data to offload. It is important to consider the use of a high speed USB interface. USB HighSpeed (480Mbit) is likely the maximum this system would be capable of at the moment. USB SuperSpeed (5Gbit) requires much higher end FPGA and/or micro-controller. An FX3 FIFO type chip could be an option to provide a 5GBit data pipe. FX3 is not as flexible as one might wish though, it is mostly meant as a high speed data pipe not a flexible USB device.

Recommendations: Storage

First Choice:

EMMC

Pro:

- We can manage our storage
- We can assure bandwidth capabilities in production
- We can format the data on the storage medium any way we want to ensure we meet our minimum write speed requirement
- Simpler to manage than NAND
- Does not require us to deal with UFS that is quite new on the market
- Wide availability of chips on the market
- Can be used with Microcontrollers as well as FPGA
- No need for an opening or retainer to access the card

Con:

- Need to implement a good USB interface from the get go
- Still might run into bandwidth limitations despite having full control over the storage medium
- More expensive than SD-Cards

Footnotes:

This could potentially be also implemented as a factory installed SD-Card that is not meant to be user replaceable. But it means we still have to deal with the bulk and weight of the SD-Card socket and it’s added cost.

Second Choice:

UFS

Pro:

- We can manage our storage
- We can assure bandwidth capabilities in production
- We can format the data on the storage medium any way we want to ensure we meet our minimum write speed requirement
- Simpler to manage than NAND
- Will definitely have the necessary size and write bandwidth we need (the core reason for it’s existence is to compete with SDIO for bandwidth while maintaining low power usage)
- No need for an opening or retainer to access the card
- Cheaper than SD-Cards at bigger capacities

Con:

- Need to implement good USB interface from the get go
- Can only be used with FPGA (MCU that support UFS don’t seem to exist yet)
- Being a new interface it does not offer as many chip options and vendors as EMMC

Third Choice:

SD-Cards

Pro:

- Ubiquitous & easy to source
- Big selection of sizes, depending on what the user needs/wants to spend

- Easy to quickly offload and keep flying a mapping mission by having multiple cards
- Known how to handle by users as it is a common solution on many cameras and drones
- No need to implement USB interface from the get-go

Con:

- Cards that meet our write speed spec are the newest generation cards
- We might need to use the PCIe interface which has knock-on effects regarding our FPGA/MCU capabilities
- Very easy for the user to buy wrong type of card and run into issues
- A lot of engineering has to be done to validate and make sure the SDIO interface works well with a variety of cards
- Mechanical opening in the case to allow for SD-Card swapping, resulting in another place for dirt and moisture ingress, not to mention the weight of the retainer
- More expensive than UFS at larger capacities

Recommendations: Hardware Architecture

First Choice:

Crosslink-NX + STM32F7

The Crosslink-NX provides 2 hardened MIPI-CSI-2 gateware cores. All IO pins are capable of being used for MIPI interfaces. This solution would use the two hardened gateware cores and one soft core to interface with the 3 cameras.

The storage device is connected directly to the FPGA to allow direct pass-through of the camera data stream onto the storage medium.

The STM32 implements all the coordination software tasks removing the need of implementing a soft core on the FPGA. (it is an option for cost optimization down the road) It also provides the USB interface for image data download from the storage medium. It has access to the storage via the FPGA.

Pro:

- This solution provides a good balance between closed and open frameworks.
  - Crosslink-NX has the prospect of getting a fully open source FPGA tool-chain (there are usable elements of it already present)
  - Even though there are more open solutions than this, the Crosslink-NX can be used for relatively fast time to market by using the proprietary tool-chain and gateware cores. More open FPGA will require much more engineering to get it off the ground.
- The main selling point of the Crosslink-NX is use as a MIPI-CSI bridge. It is pretty much built for this purpose.
- The STM32 allows for quick development of the coordination code, removing potential issues of using a soft core. But it allows for a future cost-optimized version that removes the STM32.
- STM32 is the best choice if the firmware should be implemented in Rust. It is the best supported family of chips in the ecosystem.
- This setup allows for a good flexibility of the path forward. Very little in this setup would not be transferable to another set of chips from another vendor.
- The FPGA and MCU both are relatively low cost in production.
- None or very little information that covered under an NDA is needed for development or reproduction. (depends on how much proprietary FPGA gateware is used and how those are licensed)
- All parts are off-the-shelf and easy to purchase from multiple resellers (DigiKey, Mouser) without an NDA or contracts
- All software and gateware should be possible to publish in a public open-source repository. (there might be some limitations when cutting corners to expedite engineering by using proprietary gateware aka. IP cores)

Con:

- This setup requires multiple chips, which will pose a routing and packaging challenge.
- Requires FPGA development

- The STM32 does not have a built in USB HS Phy variant, it requires the use of a separate HS ULPI Phy chip.
- Crosslink-NX only has up to two hardened MIPI-CSI interfaces, the third needs to be a soft core.

Footnotes:

The STM32 can be replaced with a whole bunch of other MCU that have tradeoffs regarding software ecosystem and capabilities. There might be a better choice than STM32 that meets USB and SDIO speed requirements better. Even more so if we want the MCU to have access to the video data, the NXP i.MX RT series MCU might be a better choice here depending on the requirement specifics.

Second Choice:

Efinix Titanium

In this solution everything is implemented in the FPGA. The MIPI-CSI is a hardened core and there is a hardened RISC-V MCU core available on the Titanium. (there is a smaller Efinix FPGA that only has the MIPI-CSI cores on board)

Pro:

- It is still quite flexible due to an FPGA fabric.
- We know people that built camera systems with the Efinix part at the center with good success
- Simpler setup with just one chip (if using a smaller FPGA might require an external MCU)
- Has enough MIPI-CSI hardened cores to interface to all three camera sensors.

Con:

- It is quite a big FPGA package (18 × 18 mm 484 balls BGA) and might be harder to integrate than even several smaller chips.
- It is extremely unlikely there will ever be any open source tool-chain for it.
- There is no known Rust ecosystem support for the built in MCU. (Can potentially be brought up though)
- USB situation is unclear

- 3x price of the first choice solution (part cost in production)

Footnotes:

Just as with the “First Choice”, this solution can be augmented with an external MCU that can make software development more flexible and open. But will add complexity, size, and cost. Even the Efinix Trion parts that don’t contain a CPU core are about twice the price of the Crosslink-NX FPGA we would consider using here.

Another equivalent alternative to this choice would be the use of Xilinx parts. There is much more developer knowledge dealing with Xilinx parts in general. There are gateware cores for all the parts we would need here. The pro for Efinix is that they are a smaller company and likely more eager for smaller customers than Xilinx. Otherwise the pros and cons are very comparable between the two solutions.

Third Choice:

Ambarella SOC

Everything is implemented using the Ambarella framework. No FPGA flexibility.

Pro:

- Likely the fastest development cycle.
- Simple setup, one chip
- Considering these chips are used in cheap consumer hardware it is likely price competitive with the other options.

Con:

- 100% proprietary, completely relying on the vendor
- Chips can’t be purchased without a relationship with Ambarella
- Requires NDA for development/evaluation
- Requires familiarity with the Ambarella ecosystem
- Unclear if Ambarella would even consider doing business with a “small” project like this

- No chance of publishing any of the code as Open Source due to the inherent NDA requirements

Footnotes:

This solution is comparable with NVidia Tegra. NVidia might be slightly more open to get started with, potentially more expensive though, and is likely an overkill from the processing capabilities standpoint.

Data Processing Glossary

HDL: (Hardware Description Language) a language that defines digital behavior. It can be used to synthesize logic that in turn can be turned into an FPGA configuration. Examples of HDL are: VHDL, Verilog, SystemVerilog, Amaranth, SpinalHDL.

FPGA: (Field Programmable Gate Array) a reconfigurable logic chip

FPGA Fabric: This is the tiled repetitive structure of the FPGA used to implement the desired design. The Fabric usually consists of a 2d array of cells. Each cell contains a multiplexer, some d-flip flops and a lookup table to implement logic operations. The specifics may vary from chip to chip.

Gateware: the configuration of an FPGA. Usually defined using an HDL. Also known as IP.

IP: (Intellectual Property) an old industry term for Gateware.

Gateware Core: A self contained block of logic with a specified function. An equivalent of a library in software. A Core can represent a peripheral, protocol or a CPU. A core can be implemented from scratch or purchased

Soft Gateware Core: Is the same as a Gateware Core, but with the emphasis that it is not Hardened.

Hardened Gateware Core: A Gateware Core implemented as a dedicated piece of silicon instead of the regular, general purpose, FPGA fabric.

Motion Blur and Vibration

Our goal is to take clear photographs of the ground, however we have high-frequency vibration and low-frequency translation motion to deal with.

The classic solution for stabilizing images from drone-based cameras is a gimbal. A gimbal helps with both high-frequency and low-amplitude stabilization like vibration, as well as low-frequency high-amplitude stabilization like tracking video shots. In our application, we are not concerned with dynamic video movements, so we are only concerned with managing vibration and linear translation.

Minimum possible blur from ground speed translation

Because the drone is in motion, there will always be some blur during sensor exposure. Reducing this kind of blur means flying slower, exposing faster, lowering sensor resolution, increasing FOV/mi, and/or flying higher.

For example, imagine a drone flying 5 m/s at 100m altitude with a 60° FOV. The image projected on the drone's sensor will "slide" across the sensor's surface at the ground speed: 5 m/s. This means that 5/115 = 4.3% of the image is replaced on the sensor every second.

For overcast days with a decently-fast lens and moderate ISO, it's reasonable to expect that we can expose our images a 1/500 second. In this amount of time, 0.0086% of the image will be replaced on the sensor.

The OV64A40 we recommend is 9.6 × 7.2 mm in size and 9248 × 6944 pixels. Each pixel is 1 µm square, so the pixels cover 96% of the linear dimension of the sensor. In a 1/500 second exposure, 0.0086% of the sensor's 9.6 mm axis will be replaced, or about 0.83 µm of linear image movement over the sensor.

This linear movement of the image translates to about a 83% of the pixel width if using the non-binned 1 µm width pixels, or 42% of the 2 µm binned pixels. A standard rule of thumb for image sharpness is to try to keep the total movement of the image projected onto the sensor to below 50% of a pixel's width. Unfortunately for us this means that even without vibration we have exceeded the blurriness rule of thumb for full resolution, and we have almost no room to spare for binned 2 µm pixel data.

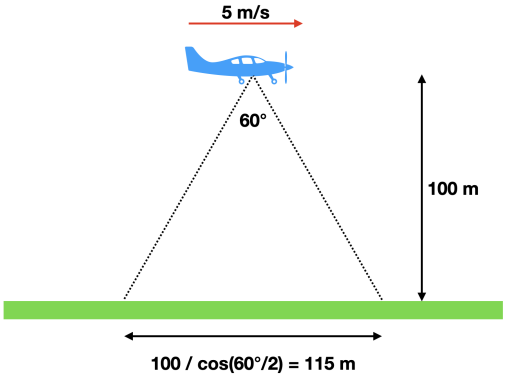
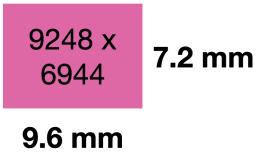


Diagram of translational blur



Camera sensor dimensions

I picked median values for altitude, ground speed, and shutter speed, but here are some bounding conditions:

- 50 meter altitude, 10 m/s ground speed, 1/250 second exposure = 665% of pixel width moved.
- 150 meter altitude, 1 m/s ground speed, 1/2000 second exposure = 3% of pixel width moved.

Play with this tool to calculate your own blurriness values.

> https://docs.google.com/spreadsheets/d/1uicselT_HtTS2us4EGqRM3fM62aF0yrUr609P-9F_q8/edit?usp=sharing

Vibration

Vibration is likely to be the most difficult challenge to address in terms of blur. Keeping in mind that almost the entire blurriness budget is already consumed by ground speed translation, reducing the contribution of vibration to a minimum is vital.

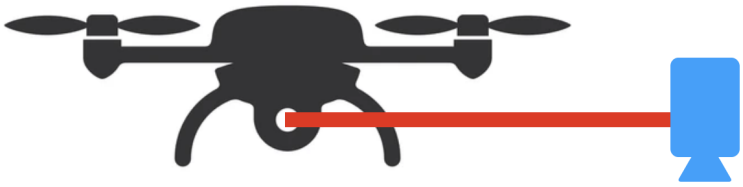
There is the fundamental problem of the translational and rotational kinetics of vibration and their effect on image projection onto the sensor. But there is also the

added effects of vibration on the camera's optics, especially the auto-focus mechanism. The Arducam OwlSight module with phase-detection autofocus we recommend is reported by customers to be more resilient to vibration than its electromechanical autofocus Hawkeye brethren. Still, these autofocus modules offer additional degrees of freedom and challenges for managing vibration.

We recommend two primary strategies for reducing vibration: (1) creating a low natural-frequency camera beam that is resonantly decoupled from the primary rotor frequency, and (2) high-frequency damping.

Low-frequency camera beam

The idea of this system is to use a light-weight and passive cantilever beam system to hold the camera. If the system has a low natural frequency and is decoupled from the rotors' resonance, then we should expect fairly low-amplitude quarter-wave sinusoidal motion at the tip of the beam. The goal here isn't to hold the camera dead-still like a gimbal would. Instead, this low-mass and low-cost system will make sure the the camera moves slowly with respect to shutter speed, minimizing blur.



Camera on a low-frequency beam

The beam's mass and rotational moment will damp high-frequency vibrations from the rotors. The motion at the tip of the beam should follow Euler-Bernoulli Beam Theory with a first mode shape of a quarter-wave sinusoid and a frequency that is inversely proportional to the square root of the mass and inversely proportional to the length of the beam to power of 1.5. If we design the mass and length of this beam to ensure it is not close to an integer multiple of the rotor frequency or any other high-energy part of the airframe's vibrational power spectral density, then most high-amplitude vibration should be eliminated by this system.

The mass of the beam will be critical because we have a very tight weight budget. We believe we can make this beam and mounting system using about 20g of carbon fiber or 3D-printed truss (and fasteners) leaving 40g of budget for modules, electronics, and wiring harnesses.

$$EI\frac{\partial^4 y}{\partial x^4} + \rho A\frac{\partial^2 y}{\partial t^2} = 0$$

Euler-Bernoulli beam theory

$$\omega_n = \sqrt{\frac{3EI}{mL^3}}$$

Natural frequency of a beam with a point mass at the end

Representation of a fixed mass at the end of a cantilever with first-mode quarter-wave sinusoidal motion

If the angular velocity of the bending beam becomes an issue with respect to blur, we could write firmware that synchronizes the photographs being taken with the moment of zero angular velocity of the camera at the end of the beam. This would require an additional IMU at the end of the beam. As long as the natural frequency of the beam is at least an order of magnitude lower than the inverse of the shutter speed (e.g. <50 Hz for a 1/500 second shutter speed), we should have a long window of time to take a still picture as the beam flexes. However, the goal is to not need this in the first place.

We are not very worried about the translational degrees of freedom with respect to vibration and blur — these are likely to be small when compared with ground speed.

High-frequency vibration isolation

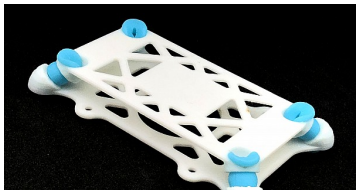
The beam will need to be mounted to the drone chassis with some mechanical mechanism. This mechanism will have a low natural frequency as discussed above, but it's still possible for high-frequency vibration to travel through the beam system, especially along its longitudinal axis. The mounting system that connects the team to the airframe is a great place to insert some damping mechanisms for high-frequency vibrations.

In order to design the damper properly, we will need to characterize the spectrum of the drone's rotors. Some searching indicates that most small-size commercial drones operate with rotor frequencies between 5,000-15,000 RPM resulting in dominant vibration frequencies between about 80-240 Hz. We can test empirically this with our drone and our airframe.

The drone photography community uses a variety of materials with different density and damping characteristics to accomplish the goal of damping vibrations from the rotors. Some of the most common materials and strategies include:

1. Metal wire rope dampers.
2. Elastomeric damper pads (neoprene, silicone, and often earplugs!).
3. Molded rubber mounts (like on DJI drones).

Wire rope damper



Earplug damper with 3D printed plates

Conceptual 3D printed drone parts. We could print the whole airframe or just the camera truss.

Conclusion

After living with this design challenge for the last few months, our team believes that there is a viable and achievable solution. We recommend building the drone using the Arducam B0483 “OwlSight” (or equivalent in fixed lens) module based around the OV64A4, the combination of Crosslink-NX + STM32F7, a low-resonance beam, and some high-frequency passive dampers. This design will results in a robust, affordable, and reliable product that will meet HOT’s imagery requirements.