Fast Onboard Stereo Vision for UAVs

Andrew J. Barry[1], Helen Oleynikova[2], Dominik Honegger[2], Marc Pollefeys[2], and Russ Tedrake[1]

I. Introduction

In the last decade researchers have built incredible new capabilities for small aircraft, with quadrotors moving from labs to toy stores and with autonomy reaching smaller and smaller vehicles. As the systems, and their payload capacities shrink, we can no longer use typical aircraft sensors such as RADAR, scanning LIDAR, and other active sensing methods for obstacle detection and avoidance. Smaller vehicles must move to lighter weight sensors.

Cameras are lightweight, fast, and information dense but can require sophisticated, often slow, processing to be useful for robotic applications. Here, we demonstrate that high-speed embedded stereo vision offers a way forward for fast-flying aerial vehicles. The presented solutions are lighter, provide more information, and use less power than laser ranging systems. They work in outdoor environments that overwhelm active IR depth cameras, and do not suffer from scale observability like monocular camera systems. The new algorithms and processing techniques presented here enable us to detect obstacles faster, with less payload, and less computational hardware than ever before.

We present the only two embedded stereo (two-camera) vision systems running at high frame rate with low enough power and weight requirements for small flying platforms. These systems provide full 3D positions of points seen by the cameras, which is essential for obstacle avoidance and safe, robust flight. These two systems were developed independently by ETH Zürich and MIT for solving the problem of providing accurate, rich sensing in a package small enough to fit in the payload constraints of mico-aerial vehicles (MAVs).

The two systems differ in their processing, with one using an onboard FPGA (field programmable gate array) to perform dense semi-global matching (SGM) [4], [6] and the second using conventional ARM processors to perform pushbroom stereo [1]. Both systems run at 120 frames per second (fps) at 320x240 pixel resolution, on duplicate fixed-wing platforms flying over 30 MPH (13.4 m/s).

The authors are with (1) Computer Science and Artificial Intellegence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA and (2) the Computer Vision and Geometry Group, Institute for Visual Computing, Computer Science Department, ETH Zürich, 8092, Zürich, Switzerland {abarry, russt}@csail.mit.edu, {dominik.honegger, marc.pollefeys}@infk.ethz.ch, oelena@ethz.ch

This work was partially supported by ONR MURI grant N00014-09-1-1051. Andrew Barry is partially supported by a National Science Foundation Graduate Research Fellowship. Dominik Honegger is partially supported by the Swiss National Science Foundation (SNF) under grant 150151.

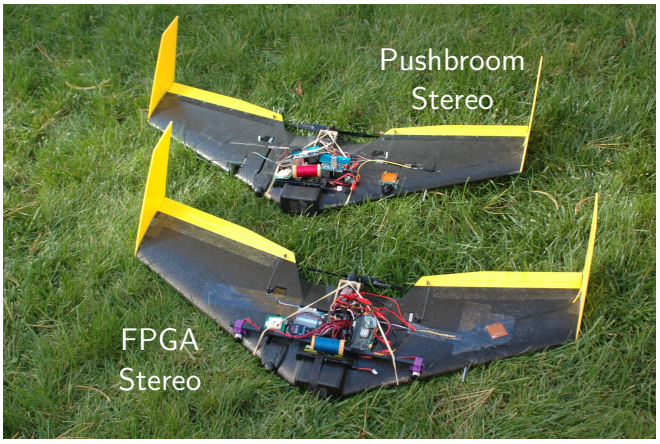


Fig. 1: Our experimental aircraft platforms holding the FGPA stereo system (front) and the pushbroom stereo system (back). Cameras are mounted on the front of the wings at the same baseline (34 cm) on both airframes. Covers over the electronics were removed for this photograph.

II. Background

In the last few years, micro air vehicles (MAVs) have demonstrated amazing agility and aggressive flight in motion capture environments. As we move toward using MAVs for field applications, sensing needs to be moved on-board.

Optical flow sensors, essentially an optical computer mouse with a different lens, are useful for guidance in relatively uncluttered environments [5], have been shown to work for autonomous takeoff, landing, and obstacle avoidance [10], and are now available in commercial products[1]. In highly complex environments, like urban or forest settings, optical flow may not be precise enough to dodge small obstacles along careful paths.

Lightweight and fast cameras have driven a substantial body of work on real-time stereo vision (see [7]). To fit these systems on continuously-smaller UAVs, the field is pushing weight and framerate requirements more than ever. At 30 frames per second (fps), an aircraft moving at 10 m/s will move one-third of a meter between frames, whereas at 120 fps, one acquires new data every 8.3 cm. With a detection horizon of only 5 meters in some cases, fast detection is essential to successful flight.

High speed FPGA-based stereo systems have seen commercial successes like the MultiSense line from Carnegie Robotics, but for the most part are too heavy for MAVs. In [2], the authors demonstrate a low power implementation of

[1]senseFly LTD. https://www.sensefly.com

SGM matching with 680x400 pixels at 25 fps on an FPGA used for pedestrian detection. Their test system performs the stereo matching in hardware but leaves image capture, distortion correction, and rectification for the CPU. Another example, [3], processes 1920x1080 images at 60fps with 256 disparity values using sum-of-absolute-differences, resulting in larger more information rich images than we present. Miniaturization of that system is required before it is ready for flight testing on an MAV that cannot carry a substantial payload weight.

A. Stereo Vision

It is well known that given several 2D images of the same scene taken from different viewpoints, one can establish correspondences between parts of the image to extract 3D depth information from the scene. In stereo, we use two cameras rigidly fixed to each other with a known horizontal displacement between then. If we find a match between points in the image planes, we can use triangulation to calculate the true 3D position that generated the projected image points.

The difficulty is searching for correct point correspondences. Geometric relations can be used to reduce the search range for a corresponding point, from the whole image to a single line. Figure 2 shows a setup with two image planes. Given point X in the 3D space and its projection X_L on the image plane of the left camera, the corresponding point X_R in the right camera plane is located somewhere on the epipolar line (red). This constraint substantially reduces the effort of searching corresponding points. After a point pair is found, the 3D coordinate is calculated using triangulation.

To produce more robust results, the search for correspondences can be based on blocks of pixels instead of single pixels. The search can also be extended to global methods, where cost paths over the entire image are computed. The popular SGM algorithm uses these global consistency constraints to penalize changes and discontinuities in the disparity image.

III. Vision Systems

In this section, we describe the two stereo vision systems, an FPGA implementation of semi-global matching, and a conventaional CPU implementation of pushbroom stereo. We detail how each system works and their key differences for use in obstacle detection on small, lightweight UAVs.

A. Fast Stereo on FPGA

Using an FPGA, we perform semi-global matching stereo (SGM) at high speed [6]. The FPGA allows us to implement the algorithm in hardware, making it vastly more efficient than a CPU or GPU implementation. We run it on a small, lightweight FPGA and companion CPU board that is 76 mm x 46 mm and weighs 50 grams.

Slight modifications to the SGM algorithm allow us to reduce latency to 2 ms. The companion CPU allows the user to send new configuration parameters, recalibrate the system, and easily access the depth map output. The FPGA acts as a layer between cameras and CPU, allowing the CPU to perform normal image-capture and receive a depth map, giving the user substantial flexibility for further processing.

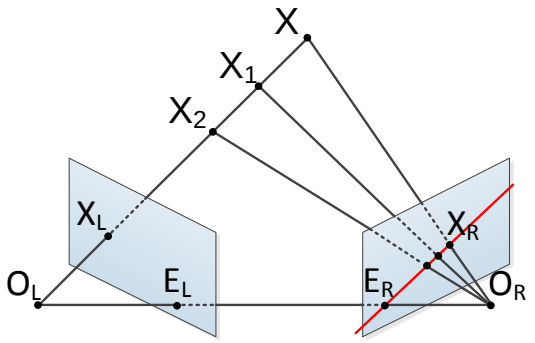


Fig. 2: A point X_L and candidates X_i for the corresponding point X_R. All of the candidates are located on the epipolar line (red) defined by the intersection of the plane X_L, O_L, and O_R with the right image plane.

B. Fast Stereo on ARM

Our second approach uses conventional ARM processors to perform a subset of standard stereo vision computation at high framerate [1]. Conventional block-matching stereo vision estimates depth by matching blocks of pixels in the left image to their counterparts in the right image. The search checks pixels at different disparities, from zero disparity where the camera's separation is insignificant compared to the object's distance, to a large disparity where the corresponding pixels appear far apart in the two images.

We think of this search as a search through depth. As disparity decreases, we are searching for matches further and further from the cameras. Given that model, if we constrain ourselves to search only at one depth, d meters away, we substantially reduce our computational load. Instead of *searching* through depth for a match, we ask, "does this pixel block appear to be an object d meters away?" In practice, this results in about a 20x reduction in computational load.

Once we can identify when obstacles are d meters away, we can use a state estimator to remember where they are and update their relative position as we continuing flying. Thus, like a broom sweeping a floor, we sweep through depth and can recover a full, local 3D pointcloud (Figure 3).

C. Comparison

1) Latency: The FPGA stereo system directly interfaces the image sensors and reads out the pixel data stream. All hardware blocks are pipelined and run at pixel clock speed of the image sensors. The overall latency of the disparity estimation pipeline is a constant 2 milliseconds, and is primarily caused by a buffer required to align the images to fulfill epipolar geometry.

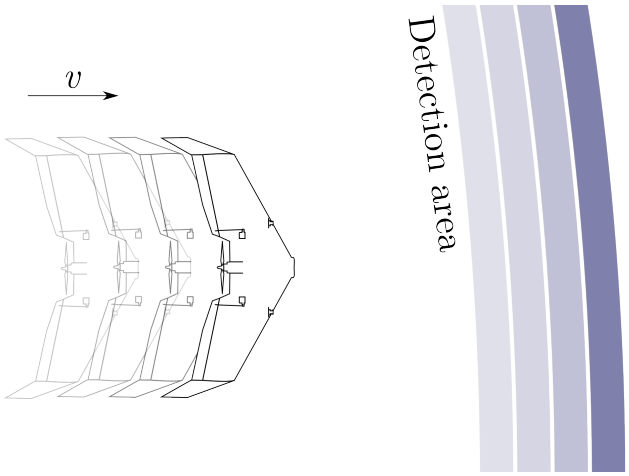


Fig. 3: Pushbroom stereo detects at only one depth (darkest blue), but can recover full depth information by remembering the obstacles it has seen (lighter blues).

Latency in the pushbroom stereo system is constrained to framerate, or 8.3 milliseconds. In the results shown here, we use a simple "detect twice" filter to remove false-positives, requiring two frames of data before deciding something is an obstacle, increasing latency to 16.6 ms. Future work will implement more sophisticated filtering to remove this additional latency.

2) Power Consumption: The total power consumption of the FPGA system is less than 5 Watts for the disparity estimation including cameras, FPGA, mobile CPU and power converters. The clock speed of the FPGA processing pipeline is selected as low as possible while maintaining full frame rate. The fully loaded pipeline leads to a uniform power consumption.

The pushbroom system requires 20 watts, running on two ODROID-U3 computers requiring 10 watts each. When idle (not capturing images) power consumption is reduced via traditional automatic scaling of CPU speed.

3) Synchronization: The FPGA system generates a global clock domain for the camera sensors. A reset trigger is set simultaneously on both sensors what results in a synchronization on a per pixel base. Therefore no buffer is needed to synchronize the image streams and the synchronization latency is minimized. This synchronization by hardware configuration allows for simultaneous exposure of both image sensors.

The pushbroom stereo system synchronizes images by initiating capture as close as possible in software via USB communication to the cameras. Interestingly, the high framerate and short exposure time limits the mismatch between frames, allowing the system to operate at high speeds.

4) Reusability/Flexibility: Given the FPGA and camera interface, the onboard configuration is interchangeable. Precompiled hardware blocks can replace the stereo core to accelerate any other algorithm, allowing even inexperienced users to modify the configuration to suit their application.

By using commercial, off-the-shelf parts and open source code, the pushbroom stereo system is easy to adapt to new platforms and hardware. Unlike the FPGA system, most choices of cameras, CPUs, and mountings will work. Any existing stereo rig with a state estimator can be reconfigured in software for pushbroom stereo.

Though the FPGA system presented is currently development hardware, it is planned to be commercially available as part of the visual-inertial navigation package called the VI Sensor from SkyBotix AG[2], which also features an IMU tightly-synchronized to the image input [8].

IV. System and Parameter Design

Stereo suffers with varying inter-camera roll, pitch, and yaw. Increased depth map density and increased detection distance forces stiffer and stiffer mounts. On MAVs, weight considerations constrain the options primarily to lightweight carbon fiber and plastic. For sufficient range on our systems with wide angle lenses, we used 34 cm baselines. It was unknown at the beginning whether this baseline could be made robust on a fast-moving airframe with vibrating propellers without adding significant weight. Our experiments confirm that our mount designs prevent both high-frequency vibration and one-time impact from misaligning the cameras.

A. Baseline and Physical Mounting

Stereo baseline, or the distance between cameras, roughly defines how far a stereo system can detect obstacles and the mounting hardware required to maintain calibration. To detect obstacles at z meters away, we note:

$$z = \frac{fb}{d} \tag{1}$$

Where z is the metric distance of the object from the camera image plane, f is the focal length in pixels, b is the baseline in meters, and d is the disparity (i.e., difference in pixel location between the left and right image) in pixels.

In the FPGA system, we search over a disparity range from 1 to 31 pixels. Since disparity is the inverse of depth, our error scales inversely with the disparity: at $d = 2$ pixels, an error of 1 pixel in the disparity estimation can result in an error of tens of meters in depth estimation. At $d = 31$ pixels, each pixel of disparity error, leads to an error of centimeters or less.

The FPGA system, with a dense depth map and long range required a stiff carbon fiber rod that rigidly attached the two cameras together (Figure 4a). The sparser, shorter range pushbroom stereo system is much more robust to small disturbances in camera calibration, which allowed us embed the cameras in the aircraft's foam body without adding additional members (Figure 4b).

Both mounts maintained calibration throughout a variety of flight conditions and ground impacts. Surprisingly, we did not find that the aircrafts' high speed motors (14,800 RPM) caused particular issues. We note that we balanced our propellers[3], which can make a substantial difference.

[2] `skybotix.com`

[3] Propeller balancing is an easy process and only requires about $20 USD for a balancer plus some sandpaper.

Rotation Axis	lower bound (deg)	upper bound (deg)	Effect
Optical Axis	0.15	2.13	no matching
Horizontal Axis	0.11	1.6	no matching
Vertical Axis	0.11	-	wrong matching

TABLE I: Effects of single-camera rotations on the stereo system. These values are computed with the camera parameters of our FPGA system.

B. Mount Tolerance Analysis

Since no camera mount is perfectly rigid, vibrations will degrade calibration and disparity estimation. Here we compute bounds for these effects on stereo vision. We assume the matching algorithm is using the camera parameters from our FPGA system. The following equations are for rotations of one camera with respect to the other.

Misalignment from rotation around the vertical or horizontal axis is computed as follows:

$$\alpha = \frac{\Delta pixel}{f} \tag{2}$$

where α is the rotation in radians, f is the focal length in pixels and $\Delta pixel$ is the shift in pixels. Rotation around the optical axis caused by a shift of the outermost pixels of a matching line is:

$$\beta = \arctan \frac{\Delta pixel}{s_x/2} \tag{3}$$

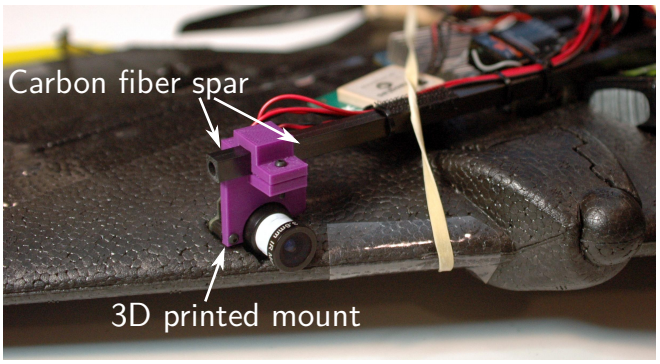
where β is the rotation in radians, $\Delta pixel$ is the vertical shift in pixels and s_x is the horizontal resolution of the image.

A rotation around the optical or horizontal axis results in an offset of the epipolar line, causing valid candidates for a corresponding point to move outside the search space. A rotation around the vertical axis shifts the candidates on the same epipolar line, resulting in an incorrect disparity estimate but valid matches by the stereo algorithm.

We provide upper and lower bounds for these rotations in Table I. The lower bounds indicate rotations where there is no effect on the matching, while the upper bounds indicate rotations large enough for the system to fail to estimate depth.

C. Focal Length

On aircraft, focal length (and thus field of view) determines how fast the aircraft can safely turn. Wide angle lenses make obstacles visible earlier in the turn, but force lower pixel density throughout the image. In practice, a wide-angle lens will reduce the distance the system can accurately perceive obstacles. The pushbroom stereo system uses a focal length of 2.1 mm, allowing for approximately 150 degrees field of view, but limiting it's effective range to about 5 meters. The FPGA system uses focal length 3.6 mm lenses, allowing it to see further with similar accuracy. We note that focal length also affects radial distortion, complicating camera calibration. In practice, most careful calibration is possible, although increasingly time-consuming with wide-angle lenses.



(a) Rigid FPGA camera mount with carbon fiber spar connecting the cameras.

(b) More flexible pushbroom stereo mount.

Fig. 4: Comparison of stereo camera mounts.

V. Test System

In order to directly compare the two vision systems, we mounted them on duplicate platforms. We chose a high-speed, maneuverable fixed-wing airplane to demonstrate the capabilities of the two systems, and see how they behave in real outdoor environments at speed.

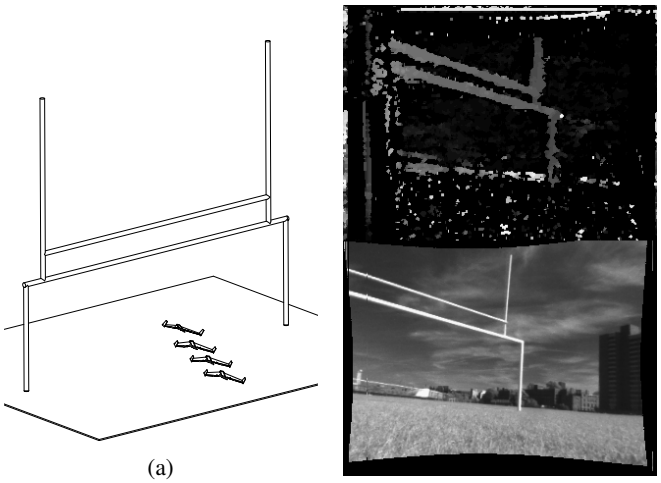
A. Fixed-Wing Airplane

We used a modified Team Black Sheep Caipirinha airframe with an onboard IMU, GPS, and pitot tube airspeed sensor from 3D Robotics in the APM 2.5 package (Figure 1). The deltawing aircraft has a 86 cm (34 inch) wingspan, a stall speed of approximately 7 m/s (15 MPH), and a maximum speed around 22 m/s (50 MPH). With the pushbroom stereo system, the aircraft weighs 664 grams. The aircraft holding the FPGA system weighs approximately 710 grams. Both airframes have a roll rate in excess of 300 degrees per second and are powered by an 8-inch propeller spinning at approximately 14,800 RPM. We note that the airframes support a passively spooling non-conductive safety tether which we used for all experiments.

VI. Results

A. Test Scenario

We flew both systems in the same location, near the same obstacles at different times (Figure 5a). Each test flight consisted of a launch from a catapult, manual flight near an American football goalpost, and landing. During each flight, we logged all sensor and perception data.



(a) (b)

Fig. 5: (a) Sketch of the flight experiment near an obstacle. (b) The FPGA produced grayscale depthmap (top) compared with raw image (bottom) of the obstacle.

This scenario allowed us to test the capabilities and robustness of our vision systems in real flight conditions, with vibration, lighting variations, and high G-forces from launch and landing present. Autonomous flights with the fixed-wing aircrafts will be explored in future work.

B. Obstacle Detection

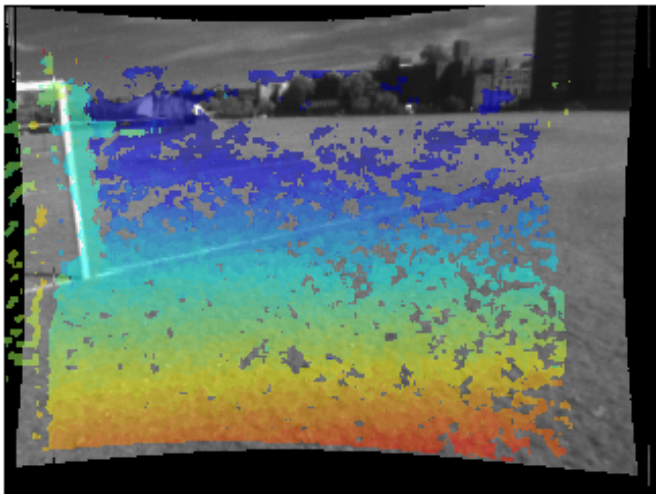
In every flight, both systems correctly identified points on the fieldgoal obstacle. The dense FPGA system correctly identified almost the entire obstacle, whereas the pushbroom system identified sections of it (Figures 6a and 6b). Each system demonstrated robustness to vibration, lighting variation, and G-forces (Figure 7).

In the case of the FPGA stereo system, the image matching stayed dense through 8+ Gs and 75 degree rolls. It also managed to remain in good calibration through approximately 10 rough landings, but needed to be recalibrated after that. Note that the failure case for the calibration is that the matches become more sparse, i.e., more similar to the ARM system.

Figure 8 demonstrates the primary difference between the two systems. The FPGA system (blue dots) produces many matches, increasing in number as the distance to the obstacle decreases. The pushbroom system detects nothing until the threshold distance (4.8 meters), around which it find matches (green stars). Past that distance, there are no additional detections, but past detections remain in memory (red crosses).

C. Disparity Data

The FPGA system produces dense stereo data, giving depth on almost the entire fieldgoal (Figure 5). The pushbroom stereo system is tuned to reject almost all outliers, so all detections can be treated as obstacles without further processing. The dense data delivers more information about nearby obstacles, but also requires more intelligent filtering



(a) Depth output from the FPGA system overlaid on the right camera image. Depth ranges from red (close) to blue (far).

(b) Obstacle detections with the pushbroom stereo system. Detections generated by this frame are in blue and past detections reprojected onto the frame via the state estimate are in red.

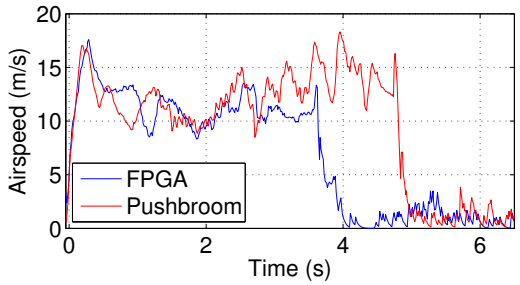
Fig. 6: Comparison of outputs detecting the same obstacle on different flights with the FPGA and pushbroom systems. Note that the pushbroom system produces substantially more sparse data than the FPGA's dense depth map.

for autonomous operation (Figures 6 and 9). We have implemented such a system in real-time in prior work using U- and V- disparity maps, as described in [9].

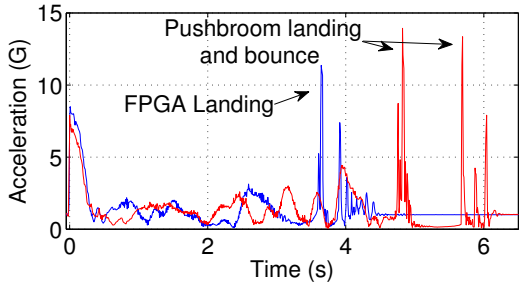
The pushbroom system, by its nature, provides meaningful data only when the vehicle is moving. This is a concern for a quadrotor in hover, but we note that hovering vehicles can often tolerate lower framerates while they are stationary.

D. Limitations

All stereo systems face limitations in untextured scenes. Two camera systems fail in the face of symmetry about the camera baseline axis, in this case, the horizontal axis, when pixels on an object look similar at many disparities. The pushbroom system is also susceptible to moving obstacles, although in this work flight speed limits that to objects moving far faster than any we encountered. Finally, high-



(a) Airspeed data from two similar flights with the FPGA system (blue) and the pushbroom system (red). Note our aircraft flies at speeds up to 17 m/s (38 MPH or 61 km/hr). Wind during both tests flights was minimal, so here we can use airspeed as a reasonable approximation of ground-speed.



(b) Total acceleration over time for a flight of the FPGA configuration (blue) and the pushbroom configuration (red). Both systems withstand approximately 8 Gs at launch, operate between ± 3 Gs, and withstand larger accelerations during landing.

Fig. 7: Data from two selected flights demonstrating the systems' robustness to high speeds and accelerations.

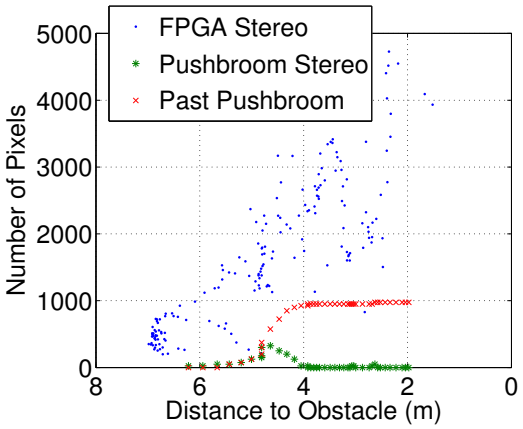


Fig. 8: Number of pixels detected while flying towards the fieldgoal obstacle. The FPGA system produces an increasing number of detections as the obstacle nears (blue dots), while the pushbroom system only detects the obstacle around the set distance of 4.8 meters (green stars). Past that, pushbroom detections (red crosses) remain in memory for avoidance. Note that the X-axis is reversed allowing time to flow left-to-right.

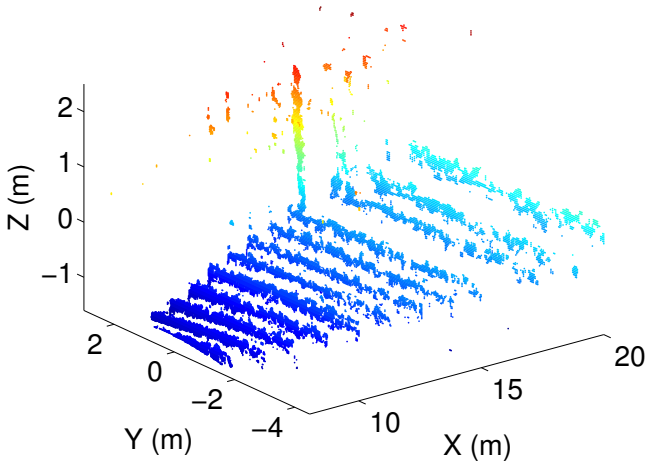


Fig. 9: 3D pointcloud generated from the depth map in Figure 6a. False colored by height ranging from blue (low) to red (high).

speed vision is forced to use short exposure times, limiting both systems to relatively bright environments.

VII. Conclusions and Future Work

Future work includes improvements to the sensor platforms and integration with autonomous control systems. With faster hardware, the pushbroom stereo system can detect obstacles at multiple discrete depths, allowing for temporal consistency checks as the vehicles moves forward. We believe that both of these systems are sufficient for obstacle avoidance in natural scenes on fast-moving vehicles. Current [6] and future work will focus on high-speed autonomous flight.

We have described the first-ever comparison of any two high-framerate stereo vision systems flying on such small platforms in real outdoor environments. The systems handle varying lighting conditions, sensor noise, blur, vibration, and high-G impact successfully. The FPGA system generates dense data, useful for a variety of tasks but requires custom hardware and software, while the pushbroom system generates highly sparse data, but is more accessible. Going forward, we see a bright future for high-speed vision systems on UAVs. As computation and sensors improve, these algorithms will scale to higher resolutions, framerates, and accuracy.

References

[1] A. J. Barry and R. Tedrake. Pushbroom stereo for high-speed navigation in cluttered environments. In *3rd Workshop on Robots in Clutter: Perception and Interaction in Clutter*, Chicago, Illinois, September 2014.

[2] S. K. Gehrig, F. Eberli, and T. Meyer. A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching. In *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, volume 5815 of *Lecture Notes in Computer Science*, pages 134–143. Springer, 2009.

[3] P. Greisen, S. Heinzle, M. Gross, and A. P. Burg. An FPGA-based processing pipeline for high-definition stereo video. In *EURASIP Journal on Image and Video Processing*. Springer, September 2011.

[4] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814 vol. 2, 2005.

[5] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *International Conference on Robotics and Automation (ICRA)*, pages 1736–1741, 2013.

[6] D. Honegger, H. Oleynikova, and M. Pollefeys. Real-time and low latency embedded computer vision hardware based on a combination of FPGA and mobile CPU. In *Intelligent Robots and Systems (IROS), IEEE International Conference on*. IEEE, 2014.

[7] F. Mroz and T. P. Breckon. An empirical comparison of real-time dense stereo approaches for use in the automotive environment. *EURASIP Journal on Image and Video Processing*, 2012(1):1–19, 2012.

[8] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Y. Siegwart. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *International Conference on Robotics and Automation (ICRA)*, 2014.

[9] H. Oleynikova, D. Honegger, and M. Pollefeys. Reactive avoidance using embedded stereo vision for MAV flight. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2015.

[10] J.-C. Zufferey, A. Beyeler, and D. Floreano. Near-obstacle flight with small UAVs. In *Proc. International Symposium on Unmanned Aerial Vehicles (UAV08), Orlando, FL*, 2008.