Pushbroom Stereo for High-Speed Navigation in Cluttered Environments

Andrew J. Barry and Russ Tedrake

***Abstract*— We present a novel stereo vision algorithm that is capable of obstacle detection on a mobile ARM processor at 120 frames per second. Our system performs a subset of standard block-matching stereo processing, searching only for obstacles at a single depth. By using an onboard IMU and state-estimator, we can recover the position of obstacles at all other depths, building and updating a local depth-map at framerate.**

Here, we describe both the algorithm and our implementation on a high-speed, small UAV, flying at over 20 MPH (9 m/s) close to obstacles. The system requires no external sensing or computation and is, to the best of our knowledge, the first high-framerate stereo detection system running onboard a small UAV.

I. Introduction

Recently we have seen an explosion of exciting results on small, unmanned aerial vehicles (UAVs) such as obstacle avoidance and trajectory tracking [24], formation flight [20], [25], and cooperative interactions with the environment [6], [23], [29]. All these systems, however, rely on an external motion-capture apparatus that gives the vehicles almost perfect state information at high rates. As we move these tasks out of the lab and into the field, we need new techniques to provide this sensing information.

A major challenge in gathering sensing data necessary for flight is the limited payload, computation, and battery life of the vehicles. These small aircraft, weighing under 1-2 kg, struggle to carry enough sensing payload for high-speed navigation in complex 3D environments. Lightweight cameras are a good solution, but require computationally efficient machine vision algorithms that can run within the limits of these vehicles. For these reasons, camera based systems have so far been limited to relatively slow, stable flight regimes [10], [31]. We aim to fly at speeds of 7-15 m/s through cluttered environments like a forest, well outside the typical speed regime. While we need fast framerates with short exposures to avoid motion blur and to have enough time to avoid oncoming obstacles, the short wingspan required to fit through gaps in the clutter limits our payload capacity.

To this end, we propose a novel method for stereo vision computation that is dramatically faster than the state of the art. Our key observation is that vehicles moving quickly through the environment can build a sufficient map using a small subset of the normal stereo computation. Using this technique, we can detect obstacles in real time at 120 frames-per-second (fps) without specialized hardware. Our system is lightweight and accurate enough to run in real time on our aircraft, allowing for true, self-contained obstacle detection.

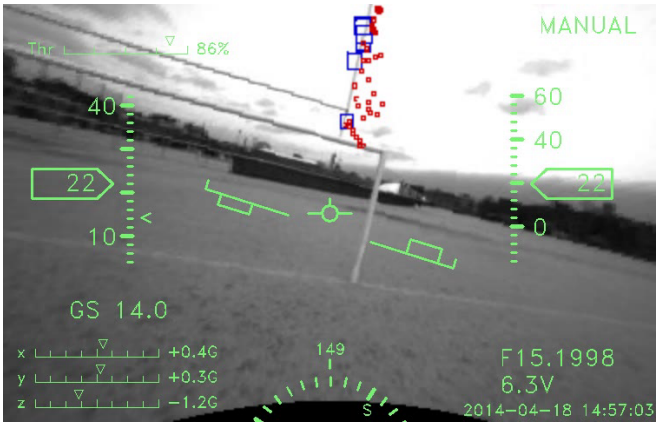


Fig. 1: In-flight snapshot of single-disparity stereo detections on a goalpost (blue boxes) and past detections integrated through the state estimate and reprojected back on the image (red dots). Overlay includes relevant flight data such as airspeed in MPH (left) and altitude in feet (right).

II. Related work

A. Obstacle Detection on UAVs

Obstacle detection on small outdoor UAVs continues to be a challenging problem. Laser rangefinders usually only support 2D detections and are generally too heavy for flight, although some systems with large wingspans [7] or limited flight time [28] exist. Other active rangefinders such as the Microsoft Kinect[1] and PrimeSense[2] systems rely on less focused infrared light and do not work in bright outdoor environments. Here we detail some of the related vision and other lightweight sensors for this class of vehicles.

B. Optical Flow

Embedded optical flow techniques rely on hardware (such as commonly found in optical mice) to compute the inter-frame changes between images to extract depth information. These techniques have worked well on UAVs, demonstrating autonomous takeoff, landing [2], [3] and obstacle avoidance [4], [36]. This technology has been successful for aircraft flight control and is now available commercially[3].

The authors are with the Computer Science and Artificial Intellegence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. {abarry, russt}@csail.mit.edu

This work was supported by ONR MURI grant N00014-09-1-1051. Andrew Barry is partially supported by a National Science Foundation Graduate Research Fellowship.

[1]Microsoft, Inc. http://www.microsoft.com/en-us/kinectforwindows/

[2]PrimeSense, LTD. http://www.primesense.com/

[3]senseFly LTD. http://www.sensefly.com/

Embedded optical flow, however, is limited in its resolution, providing only general guidence about obstacles. For more sophisticated flight, such as flying in a cluttered environment like a forest, we must look beyond embedded optical flow techniques for solutions that provide greater resolution.

C. Monocular Vision

Monocular vision techniques are attractive because they require only a single, lightweight camera, are readily available, and easy to deploy. State of the art monocular depth estimation, however, is generally not fast and reliable enough for obstacle avoidance on fast-flying UAVs. Using expert demonstrations and learning, Ross et al. demonstrate a UAV flying at 1.5m/s through a forest [30]. Recently PTAM (parallel tracking and mapping) [19] has allowed systems with a downward facing camera to perform stable, drift-free hover and slow flight [31]. When integrating inertial measurement sensors and a barometric altimeter, stable flights in indoor and outdoor environments are possible [1]. With a full vison-aided inertial navigation system (VINS), Li et al. have shown remarkable tracking with commodity phone hardware, demonstrating tracking within 0.5-0.8% of distance traveled for significant distances [21], [22].

D. Stereo Vision

While we have seen substantial improvements in monocular vision systems recently, they are not yet fast or accurate enough for high-speed obstacle avoidance on small UAVs. Stereo systems suffer from a similar speed issue, with most modern systems running at or below 50 Hz [8], [35]. Recent work on an ultra lightweight flapping UAV demonstrated a 4.0 gram line-based stereo algorithm running at various rates from 11 Hz at 128x96 to 40 Hz at 128x32 on a 168 Mhz embedded processor [9]. Their algorithm also aims to significantly reduce computational requirements, but does so by matching along lines of images instead of reducing the number of disparities searched. In [12], Goldberg demonstrates significant low-level optimization and the use of a DSP to obtain stereo frame rates up to 46 Hz on a Gumstix processor with the same cameras we use below.

Plane-sweep stereo sweeps planes through images at different disparities, projecting the different camera views into the same space, allowing for photoconsistency checks at different disparities [33]. Our pushbroom stereo algorithm takes a similar approach, but focuses on computational efficiency through reduction of the number of disparities, or swept planes.

Multi-resolution approaches can reduce computation load by searching first in low-resolution versions of the input images and progressing to higher resolutions in local areas of interest [11]. While fast, these approaches can suffer from small obstacles that were missed in the low-resolution pass.

Honegger et al. recently demonstrated an FPGA (Field Programmable Gate Array) based system that can compute optical flow and depth from stereo on a 376x240 image pair at 127 fps or 752x480 at 60 fps [15], [16]. Unlike many other dedicated processors such as [26], [34], and [32], their system is compact and lightweight enough for use on a small UAV, but requires specialized hardware. By comparison, our approach performs less computation and can work easily on conventional hardware, but relies on the hypothesis that it is sufficient for high-speed flight to compute a subset of the stereo matches.

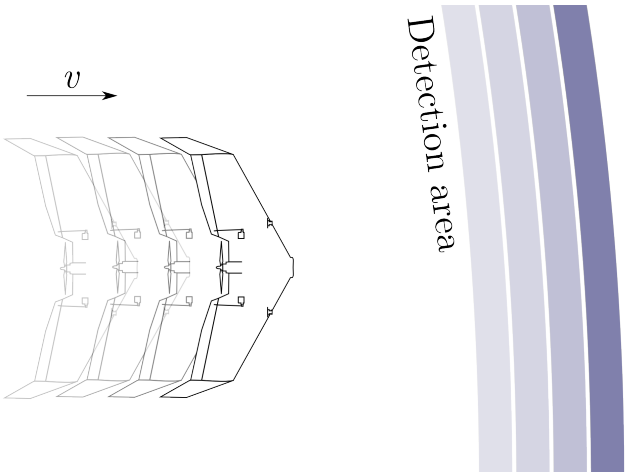


Fig. 2: By detecting at a single depth (dark blue) and integrating the aircraft's odometry and past detections (lighter blue), we can quickly build a full map of obstacles in front of our vehicle.

III. Proposed Method

A. Block-Matching Stereo

A standard block-matching stereo system produces depth estimates by finding pixel-block matches between two images. Given a pixel block in the left image, for example, the system will search through the epipolar line[4] to find the best match. The position of the match relative to its coordinate on the left image, or the disparity, allows the user to compute the 3D position of the object in that pixel block.

B. Pushbroom Stereo

One can think of a standard block-matching stereo vision system as a search through depth. As we search along the epipolar line for a pixel group that matches our candidate block, we are exploring the space of distance away from the cameras. For example, given a pixel block in a left image, we might start searching through the right image with a large disparity, corresponding to an object close to the cameras. As we decrease disparity (changing where in the right image we are searching), we examine pixel blocks that correspond to objects further and further away, until reaching zero disparity, where the stereo base distance is insignificant compared to the distance away and we can no longer determine the obstacle's location.

Given that framework, it is easy to see that if we limit our search through distance to a single value, d meters away, we can substantially speed up our processing, at the cost of neglecting obstacles at distances other than d. While this might seem limiting, our cameras are on a moving platform (in this case, an aircraft), so we can quickly recover the

[4]Standard calibration and rectification techniques provide a line, called the epipolar line, on which the matching block is guaranteed to appear.

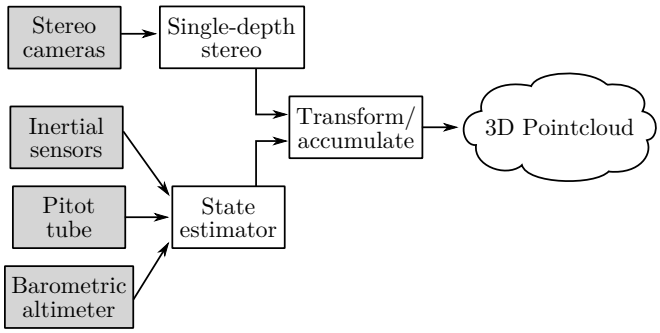


Fig. 3: Pushbroom stereo overview.

missing depth information by integrating our odometry and previous single-disparity results (Figure 2). The main thing we sacrifice is the ability to take the best-matching block as our stereo match; instead we must threshold for a possible match.

During flight, this algorithm does not increase obstacle detection latency over a full stereo system if d is set to the maximum resolvable distance for the stereo baseline. Once the platform has covered d meters, any new obstacles will be immediately identified at the maximum possible distance (excluding moving obstacles). We do sacrifice full information if the platform turns sharply or in place. In practice, on our aircraft with wide angle lenses, this is not a large concern.

We give this algorithm the name "pushbroom stereo" because we are "pushing" the detection region forward, sweeping up obstacles like a broom on a floor (and similar to pushbroom LIDAR systems [27]). We note that this is distinct from a "pushbroom camera," which is a one-dimensional array of pixels arranged perpendicular to the camera's motion [13]. These cameras are often found on satellites and can be used for stereo vision [14].

C. Odometry

Our system requires relatively accurate odometry over short time horizons. This requirement is not particularly onerous because we do not require long-term accuracy like many map-making algorithms. In our case, the odometry is only used until the aircraft catches up to its detection horizon, which on many platforms is 5-10 meters away. We discard obstacles that are behind the platform, since we are building only a local 3D map. We demonstrate that on aircraft, a wind-corrected airspeed measurement (from a pitot tube) is sufficient. On a ground robot, we expect that wheel odometry would be adequate.

IV. Implementation

A. Pushbroom Algorithm

Like other block-matching algorithms, we use sum of absolute differences (SAD) to detect pixel block similarity. In addition to detecting matching regions, we score blocks based on their abundance of edges. This allows us to disambiguate the situation where two pixel blocks might both be completely black, giving a good similarity score, but still not providing a valid stereo match. To generate an edge map, we use a Laplacian with an aperture size (`ksize`) of 3. We then take the summation of the 5x5 block in the edge map and reject any blocks below a threshold for edge-abundance.

(a) Without horizontal invariance filter. (b) Horizontal invariance filter enabled.

Fig. 4: All stereo systems suffer from repeating textures which cannot be disambiguated with only two cameras. Here, we demonstrate our filter for removing self-similarity. In this case, the horizon is not uniformly flat, so horizontal invariance causes false positives even when the aircraft is at an angle to the horizon. Detected pixel groups are marked with squares.

After rejecting blocks for lack of edges, we score the remaining blocks based on SAD match divided by the summation of edge-values in the pixel block:

$$S = \frac{\overbrace{\sum_{i=0}^{5x5} |p(i)_{left} - p(i)_{right}|}^{\text{Sum of absolute differences (SAD)}}}{\sum_{i=0}^{5x5} L\left(p(i)_{left}\right) + L\left(p(i)_{right}\right)}$$

where $p(i)$ denotes a pixel value in the 5x5 block and L is the Laplacian. We then threshold on the score, S, to determine if there is a match.

We have deliberately chosen a design and parameters to cause sparse detections with few false positives. For obstacle avoidance, we do not need to see every point on an obstacle but a false positive might cause the aircraft to take unnecessary risks to avoid a phantom obstacle.

All two-camera stereo systems suffer from some ambiguities. With horizontal cameras, we cannot disambiguate scenes with horizontal self-similarity, such as buildings with grid-like windows or an uninterrupted horizon. These horizontal repeating patterns can fool stereo into thinking that it has found an obstacle when it has not.

While we cannot correct these blocks without more sophisticated processing or additional cameras, we can detect and eliminate them. To do this, we perform additional block-matching searches in the right image near our candidate obstacle. If we find that one block in the left image matches blocks in the right image at different disparities, we conclude that the pixel block exhibits local self-similarity and reject it. While this search may seem expensive, in practice the block-matching above reduces the search size so dramatically that we can run this filter online. Figure 4 demonstrates this filter running on flight data.

B. Hardware Platform

We implemented the pushbroom stereo algorithm on a quad-core 1.7Ghz ARM, commercially available in the

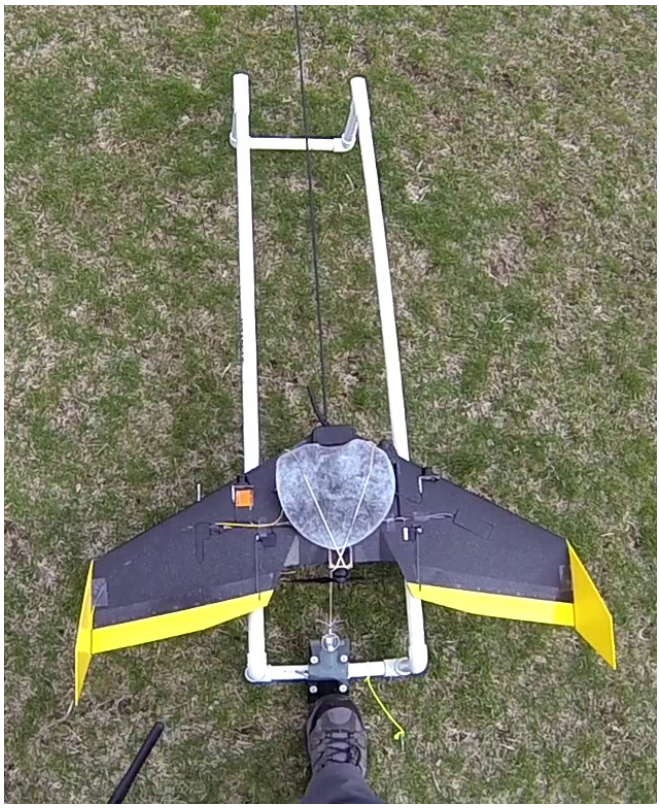


Fig. 5: Aircraft hardware in the field. We use a small catapult for consistent launches near obstacles.

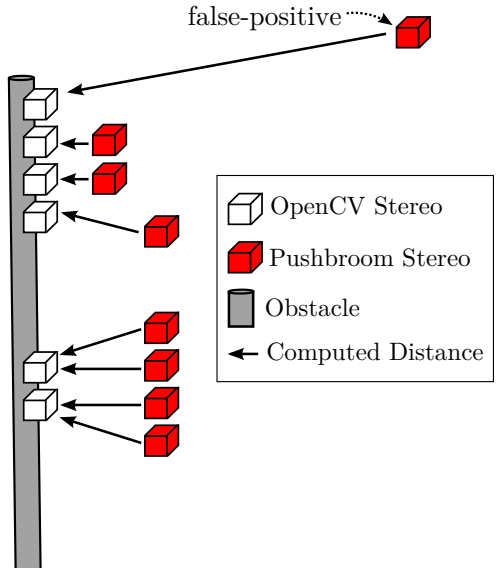


Fig. 6: Sketch of our evaluation strategy for single-disparity stereo. We detect false-positives by computing the distance from single-disparity stereo's output (red) to the nearest point from OpenCV's `StereoBM` (white). False positives stand out with large distances (labeled box).

ODROID-U2 package, weighing under 50 grams[5]. Our cameras' resolution and stereo baseline (34 cm / 14 in) can support reliable detections out to approximately 5 meters, so we use 4.8 meters as our single-disparity distance. We detect over 5x5 pixel blocks, iterating through the images with 8 parallel threads.

We use two Point Grey Firefly MV[6] cameras, configured for 8-bit grayscale with 2x2 pixel binning, running at 376x240 at 120 frames per second. A second ODROID-U2, communicating over LCM [18], runs our state-estimatator (a 12-state Kalman filter from [7]) and connects to our low-level interface, a firmware-modified APM 2.5[7], which provides access to our servo motors, barometric altimeter, pitot tube airspeed sensor, and 3-axis accelerometer, gyroscope, and magnetometer suite. The second computer also uses the state-estimate to transform the pushbroom stereo data into a global map frame and accumulate it using OctoMap [17], allowing us to build a local 3D pointcloud in realtime.

This platform sits aboard a modified Team Black Sheep Caipirinha[8], a 34 inch (86 cm) wingspan delta wing with a 2000kV brushless DC motor and 8-inch propeller[9] (Figure 5). The with the computational and sensor payload, the plane weighs 664 grams and has a stall speed of approximately 15 MPH (7 m/s). The aircraft's maximum speed is 55-65 MPH (24-29 m/s) and it has a roll-rate exceeding 300 degrees per second, allowing for fast obstacle avoidance. All outdoor flights are conducted with the aircraft under control of a passively-spooling non-conductive 250 meter safety tether.

[5]Hardkernel co., Ltd. `http://hardkernel.com`

[6]Point Grey Research, Inc. `http://www.ptgrey.com`

[7]3D Robotics, Inc. `http://3drobotics.com/`

[8]Team Black Sheep, `http://team-blacksheep.com/products/prod:tbscaipirinha`

[9]Graupner 8x5 propeller.

V. Results

A. Single-Disparity Stereo

To determine the cost of thresholding stereo points instead of using the best-matching block from a search through depth, we walked with our aircraft near obstacles and recorded the output of the onboard stereo system with the state-estimator disabled[10]. We then, offline, used OpenCV's [5] block-matching stereo implementation (`StereoBM`) to compute a full depth map at each frame. We then removed any 3D point that did not correspond to a match within 0.5 meters of our single-disparity depth to produce a comparison metric for the two systems.

With these data, we detected false-positives by computing the Euclidean distance from each single-disparity stereo coordinate to the nearest point produced by the depth-cropped StereoBM (Figure 6). Single-disparity stereo points that are far away from any StereoBM points may be false-positives introduced by our more limited computation technique. StereoBM produces a large number of false negatives, so we do not perform a false-negative comparison on this dataset (see Section V-B below.)

Our ground dataset includes over 23,000 frames in four different locations with varying lighting conditions, types of

[10]Our state-estimator relies on the pitot-tube airspeed sensor for speed estimation, which does not perform well below flight speeds.

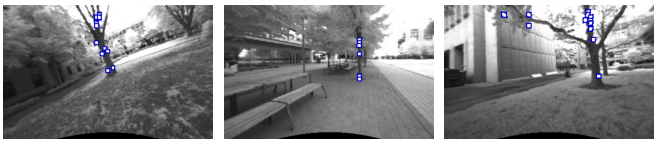


Fig. 7: Selected images from the ground dataset. Single-disparity detections at 4.8 meters are highlighted in blue.

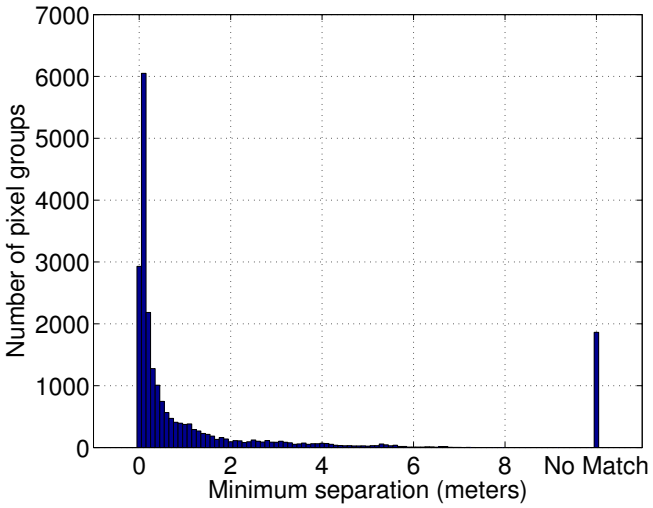


Fig. 8: Results of the false-positive benchmark described in Figure 6 on 23,000+ frames. *No Match* indicates single-disparity points where there was no matching StereoBM point on the frame. We find that only 8.2% of detected pixels fall into this category.

obstacles, and obstacle density. Over the entire dataset, we find that single-disparity stereo produces points within 0.5 meters of StereoBM 60.9% and within 1.0 meters 71.2% of the time (Figures 7 and 8). For context, the aircraft's wingspan is 0.86 meters and it covers 0.5 meters in 0.03 to 0.07 seconds.

B. Flight Experiments

To test the full system with an integrated state-estimator, we flew our platform close to obstacles (Figure 1) on three different flights, recorded control inputs, sensor data, camera images, and on-board stereo processing results. Figure 9 shows on-board stereo detections as the aircraft approaches an obstacle.

During each flight, we detected points on every obstacle in real time. Our state estimate was robust enough to provide online estimation of how the location of the obstacles evolved relative to the aircraft. While these flights were manually piloted, we are confident that the system could autonomously avoid the obstacles with these data. The integration of the planning and control system will be reported in future work.

To benchmark our system, we again used OpenCV's block-matching stereo as a coarse, offline, approximation of ground truth. At each frame, we ran full block-matching stereo, recorded all 3D points detected, and then hand-labeled regions in which there were obstacles to further increase StereoBM's accuracy.

We compared those data to pushbroom stereo's 3D data in two ways. First, we performed the same false-positive detection as in Section V-A, except we included *all 3D points seen and remembered* as we flew forward. Second, we searched for false-negatives, or obstacles pushbroom stereo missed, by computing the distance from each StereoBM coordinate to the nearest 3D coordinate seen and remembered by pushbroom stereo (Figure 12a).

Figures 11 and 12b show the results of the false-positive and false-negative benchmarks on all three flights respectively. Our system does not produce many false-positives, with 74.8% points falling within 0.5 meters and 92.6% falling less than one meter from OpenCV's StereoBM im-

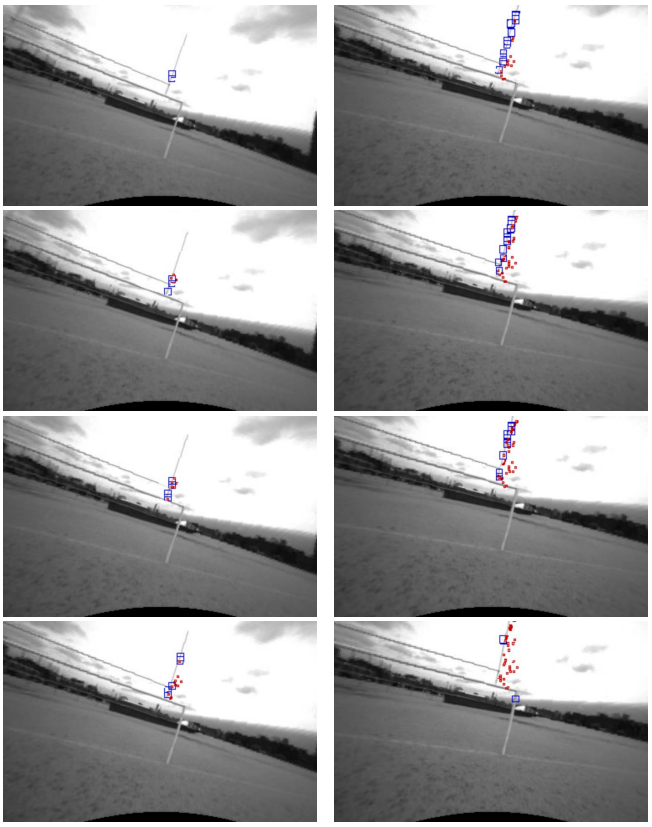


Fig. 9: Sequence of stills from an obstacle detection. Each image is 0.02 seconds (20ms) after the previous. The entire set captures 0.16 seconds. Here, the fieldgoal is detected in the first frames (blue boxes). Afterwards, the position of those detections is estimated via the state estimator and reprojected back onto the image (red dots).

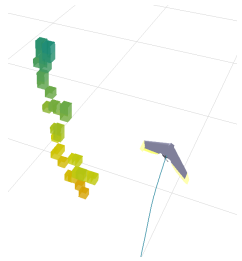


Fig. 10: Obstacle detection from Figure 9 rendered in a 3D visualizer. While we do not endeavor to build maps, our system outputs pointclouds providing compatibility with many existing planning, visualization, and control tools.

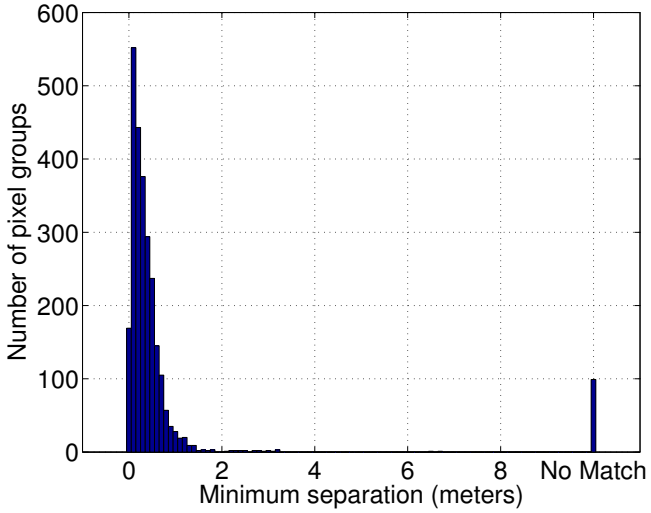
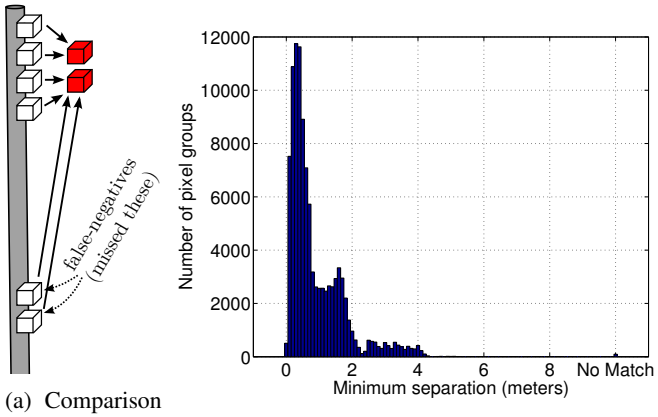


Fig. 11: Results of the comparison as described in Figure 6 on flight data. Our system produces few outliers (74.8% and 92.6% within 0.5 and 1.0 meters respectively), even as we integrate our state estimate, and the obstacle positions, forward. *No Match* indicates points that pushbroom stereo detected but there were no block-matching stereo detections on that frame.



(a) Comparison technique.

(b) False-negative comparison.

Fig. 12: Results of the false-negative benchmark on flight data. In this comparison, we compute distance from each StereoBM point (white) to the nearest pushbroom stereo coordinate (red). False-negatives stand out with large distances. Pushbroom stereo performs well, detecting an obstacle within 2.0 meters of StereoBM 91.3% of the time.

plementation.

For comparison, a system producing random detections at random times resulted in 1.2% and 3.2% of the points falling within 0.5 meters and 1.0 meters of StereoBM respectively. We set this benchmark system to produce, on expectation, the same number of points per run as our pushbroom data and to generate those points at random in-image locations. It uses the same state-estimate and projections as our pushbroom stereo system.

As Figure 12 shows, pushbroom stereo detects most of the points on obstacles that StereoBM sees, missing by 1.0 meters or more 32.4% of the time. The random system misses approximately 86% of the time by the same metric. For context, the closest our skilled pilot ever flew to an obstacle was about two meters.

These metrics demonstrate that the pushbroom stereo system scarifies a limited amount of performance for a substantial reduction in computational cost, and thus a gain in speed. All thresholds were set experimentally by hand and could be optimized in the future. We note that all data in this paper used identical threshold, scoring, and camera calibration parameters.

VI. Conclusion

Here we describe a system that performs stereo detection with a single disparity. A natural extension would be to search at multiple disparities, perhaps enabling tracking of obstacles through two or more depths. As computational power increases, we can increase the number of depths we search, continuously improving our detection. Furthermore, given a faster CPU, pushbroom stereo can process higher resolution images than traditional methods, allowing us to scale to newer, higher resolution cameras.

We have demonstrated a novel stereo vision algorithm for high-framerate detections of obstacles. Our system is capable of quickly and accurately detecting obstacles at a single disparity and using a state-estimator to update the position of obstacles seen in the past, building a full, local, 3D map. It is capable of running at 120fps on a standard mobile-CPU and is lightweight and robust enough for flight experiments on small UAVs. This system will allow a new class of autonomous UAVs to fly in clutter with all perception and computation onboard.

Source Code Availability

Videos of flight experiments and our implementation of the algorithms described above are available: `https://github.com/andybarry/flight`.

References

[1] M. Achtelik, S. Weiss, and R. Siegwart. Onboard imu and monocular vision based control for mavs in unknown in-and outdoor environments. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 3056–3063. IEEE, 2011.

[2] D. B. Barber, S. R. Griffiths, T. W. McLain, and R. W. Beard. Autonomous landing of miniature aerial vehicles. *Journal of Aerospace Computing, Information, and Communication*, 4(5):770–784, 2007.

[3] A. Beyeler, J.-C. Zufferey, and D. Floreano. optiPilot: control of take-off and landing using optic flow. In *Proceedings of the 2009 European Micro Air Vehicle conference and competition (EMAV '09)*, 2009.

[4] A. Beyeler, J.-C. Zufferey, and D. Floreano. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201–219, 2009.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] D. Brescianini, M. Hehn, and R. D'Andrea. Quadrocopter pole acrobatics. In *International Conference on Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ*, pages 3472–3479. IEEE, 2013.

[7] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in gps-denied environments using onboard sensing. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2012.

[8] J. Byrne, M. Cosgrove, and R. Mehra. Stereo based obstacle detection for an unmanned air vehicle. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2830–2835. IEEE, 2006.

[9] C. De Wagter, S. Tijmons, B. Remes, and G. de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *Proc. 2014 IEEE/RSJ Int. Conf. on Robotics and Autonomous Systems (ICRA), Hong Kong, China, 2–5 June 2014*, 2014.

[10] J. Engel, J. Sturm, and D. Cremers. Camera-based navigation of a low-cost quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2815–2821. IEEE, 2012.

[11] U. Franke and A. Joos. Real-time stereo vision for urban traffic scene understanding. In *IEEE Intelligent Vehicles Symposium (2000: Dearborn, Mich.). IV 2000: proceedings of the IEEE Intelligent Vehicles Symposium 2000*, 2000.

[12] S. B. Goldberg and L. Matthies. Stereo and imu assisted visual odometry on an omap3530 for small robots. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 169–176. IEEE, 2011.

[13] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(9):963–975, 1997.

[14] 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. IEEE, 2005.

[15] D. Honegger, P. Greisen, L. Meier, P. Tanskanen, and M. Pollefeys. Real-time velocity estimation based on optical flow and disparity matching. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5177–5182. IEEE, 2012.

[16] 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 *International Conference on Intelligent Robots and Systems*, Chicago, Illinois, USA, 2014. IEEE/RSJ.

[17] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.

[18] A. S. Huang, E. Olson, and D. C. Moore. LCM: Lightweight communications and marshalling. *International Conference on Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ*, pages 4057–4062, October 2010.

[19] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[20] A. Kushleyev, V. Kumar, and D. Mellinger. Towards a swarm of agile micro quadrotors. In *Robotics: Science and Systems*, 2012.

[21] M. Li, B. Kim, and A. Mourikis. Real-time motion tracking on a cellphone using inertial sensing and a rolling shutter camera. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[22] M. Li and A. Mourikis. 3-d motion estimation and online temporal calibration for camera-imu systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[23] Q. Lindsey, D. Mellinger, and V. Kumar. Construction with quadrotor teams. *Autonomous Robots*, 33(3):323–336, 2012.

[24] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Proceedings of the 12th International Symposium on Experimental Robotics (ISER 2010)*, 2010.

[25] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Cooperative grasping and transport using multiple quadrotors. In *Proceedings of the international symposium on distributed autonomous robotic systems*, 2010.

[26] C. Murphy, D. Lindquist, A. M. Rynning, T. Cecil, S. Leavitt, and M. L. Chang. Low-cost stereo vision on an fpga. In *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, pages 333–334. IEEE, 2007.

[27] A. Napier, P. Corke, and P. Newman. Cross-calibration of push-broom 2d lidars and cameras in natural scenes. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3679–3684. IEEE, 2013.

[28] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *International Symposium of Robotics Research (ISRR)*, Singapore, 2013.

[29] R. Ritz, M. W. Muller, M. Hehn, and R. D'Andrea. Cooperative quadrocopter ball throwing and catching. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4972–4978. IEEE, 2012.

[30] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[31] I. Sa, H. He, V. Huynh, and P. Corke. Monocular vision based autonomous navigation for a cost-effective mav in gps-denied environments. In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 1355–1360. IEEE, 2013.

[32] F. Schumacher and T. Greiner. Matching cost computation algorithm and high speed fpga architecture for high quality real-time semi global matching stereo vision for road scenes. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 3064–3069. IEEE, 2014.

[33] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[34] J. I. Woodfill, G. Gordon, and R. Buck. Tyzx deepsea high speed stereo vision system. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 41–41. IEEE, 2004.

[35] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–211. IEEE, 2003.

[36] 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.