Accessible Aerial Autonomy

Nick Berezny, Lilian de Greef, Bradley Jensen, Kimberly Sheely, Malen Sok, David Lingenbrink, and Zachary Dodds
Computer Science Department; CS REU in Systems
Harvey Mudd College
Claremont, CA, USA
Contact: dodds@cs.hmc.edu

Abstract — This work presents a combination of software and hardware that makes aerial autonomy substantially more accessible both in terms of programmatic complexity and in terms of cost. We use the ARDrone quadrotor helicopter and Willow Garage's *Robot Operating System* software infrastructure to demonstrate several autonomous tasks. Using vision as the sole aerial sensor, we demonstrate point-to-point navigation, aerial support of a ground robots, and robot localization within image-based maps. In contrasting several variations of SURF-feature matching, we demonstrate that low-cost aerial platforms can support robust, landmark-free visual spatial reasoning. This evaluation shows that aerial platforms can be practical, cost- and time-effective components of task-performing systems. We argue that aerial autonomy should be considered a broadly accessible resource, within reach of any investigator or educator of AI robotics.

Keywords: aerial autonomy, quadcopter visual control, image-based localization

I. Introduction

Aerial platforms have a long history within the field of robotics. Some of today's most impressive remote systems are aerial vehicles acting under shared-autonomy with human operators [1,2]. Fully autonomous aerial vehicles demonstrate remarkably acrobatic maneuvers [3,4] and cooperative task-completion [5].

Yet to date, aerial autonomy has been a specialized subfield that required resources only a few institutions could sustain. Traditional fixed-wing aircraft need large staging areas for take-off, flight, and landing. They also require long-distance radio links to maintain communications with offboard computational resources – or, at the least, the opportunity for emergency human intervention. Even helicopters, though usable indoors in closer quarters, are often confined to specially-designed indoor spaces in which external motion-capture systems accurately estimate the pose of the vehicles. In short, autonomous aerial robots have not yet earned the description *practical*.

Recent software and hardware advances suggest that this exclusivity may be changing. Low-cost platforms that exploit the toy industry's economies of scale and freely available software scaffolding significantly lower the barriers to entry in investigating autonomous aerial robotics. As this paper shows, however, these very affordable and practical resources also change the *kinds* of tasks possible aerial In particular, the inevitable compromises in control make AI-type tasks the most promising foundation for aerial investigations. In short, we find that aerial robots are now as accessible as more traditional ground robots both as a platform for education, research and cooperative task-accomplishment.

We thank the NSF, the Rose Hills Foundation, and Harvey Mudd College.

A. Related work and this work's contributions

Long-distance vision-based aerial autonomy has a deep history [6], but far less work has been done in small-scale settings. Here, we follow the work of Bills et al [7], in which perspective cues from large line segments guide a quadrotor copter through an uncontrived indoor environment. We agree that computer vision is the most natural – and accessible – sensing modality for aerial platforms. Complementing the use of line segments, we investigate the effectiveness of local visual features including color blobs, contrived landmark patterns, and SURF-described textures for task support.

The tasks we investigate comprise (1) autonomous aerial sensing support for ground-vehicle navigation, (2) point-to-point navigation using artificial landmarks, and (3) landmark-free localization within a map of panoramas. By undertaking these tasks, we contribute

- a field evaluation of the hardware (the ARDrone) and software infrastructure (ROS) used, particularly for other researchers and educators considering aerial platforms
- software drivers that support the drone through ROS
- algorithms for vision-based localization and control of the quadcopter using only on-board sensing

We conclude with perspective on the benefits and drawbacks of aerial platforms as a basis for educational and research robotics. We hope that this work makes aerial robotics a more practical – and productive – area of investigation even under significant resource and time constraints.

II. Hardware and Software Overview

A. ***Hardware****: the ARDrone Parrot*

Beside an iRobot Create for scale, the $300 ARDrone Parrot quadrotor helicopter (the "drone") appears in Figure 1 (top). A remote-control toy, the drone operates by sending and receiving ASCII strings via 802.11b. The drone provides its own wireless hub to which any external wifi device can connect as a client. A user can fly the drone capably within ten minutes out of the box using one of the the freely available Android or iOS clients.

As a robot, the drone has significant advantages and disadvantages. In the latter category are (1) a limited battery life of about 15 minutes of flight, though its camera lasts for well over an hour on a single battery charge – a useful feature for debugging image-processing routines! (2) ARDrone publishes full specifications of the ASCII protocol it uses, but it provides no details of the quadcopter's internal firmware control nor its electrical/mechanical system. Short of fully reverse-engineering the device, the drone is most usefully considered a black box that accepts velocity commands and broadcasts a video stream and internal data such as battery charge. Thus, standard interaction with the drone is entirely off-board. (3) Partly because of the available API calls and even more because of the inherent difficulties of stable hovering, the most striking disadvantage of the drone is that it is not a precisely positionable device. Even in simple real-world envinronments such as indoor hallways, foyers, and laboratories, the downdraft from the rotors pushes the drone in unpredictable ways; neither the sensing nor the control bandwidth suffices to counteract this. In short, we quickly realized that the remarkable perching examples from [4,5] would not be the kinds of tasks that the drone could support.

Figure 1. (top) The ARDrone Parrot ("the drone") beside an iRobot Create for comparison. (bottom) An image taken from the drone's forward camera while under remote control. The student in the red shirt is using superman-like gestures, sensed by a Kinect, as input.

Rather than consider this limitation insurmountable, however, we feel that the drone's limitations motivate the investigation of new types of tasks that even low-cost aerial robots can support. The drone does have a number of strengths we can leverage. First, though not precisely *positionable*, the drone is reliably *controllable*, and will move in the direction commanded, relative to its current pose, until encountering an obstacle. (2) The drone offers both a downward and a forward camera: their image streams can be multiplexed in the four different combinations shown in Figure 3. It has a single range sensor, a downward-pointed sonar, and sensors for its internal electrical state. The drone also senses its orientation: the simplest method to stop it when something has gone wrong is to grab it mid-flight and flip it over, triggering an emergency shut off of all of its motors. (3) The drone is robust – ours have survived many crashes that we thought had ended their working life. What is more, it is easily repairable, with ample online documentation for the procedures involved.

B. ***Software**: Willow Garage's Robot Operating System*

The ARDrone offers a C++ software development kit (SDK) that provides source code through which users may programmatically access the ASCII protocol it uses. In experiementing with this SDK, we found it difficult to compile and use across multiple operating systems. That said, with enough effort, we succeeded in building applications that allowed alternative interfaces by which to remote-control the device. For example, Figure 1 (bottom) shows a still from a movie the drone took as its flight was being controlled by superman-like gestures made in front of a Kinect sensor. We experimented with third-party drivers, too, including pure-Java SDK and a pure Python interface, known as pydrone.

Because vision is the primary sensor available to the drone, we sought to leverage the powerful vision library OpenCV [8], stewarded by Willow Garage. To integrate the visual processing with control as efficiently as possible, we decided to use Willow Garage's robotics middleware, known as the *Robot Operating System (ROS)* [9]. ROS installed without a hitch under Ubuntu, though the integration with the ARDrone's SDK did not proceed as smoothly. Because of this, we investigated the pure-python pydrone package and adapted it to handle all four image-stream possibilities in Figure 3.

ROS organizes the diverse hardware and software components of a robotics system into *nodes*. Both message-passing and subscriber-based interactions are supported: the latter are called *services*. We implement a ROS wrapper around pydrone that provided services to control the drone and to grab the images from its cameras. That one-week effort provides a layer of abstraction that presents the drone in the same manner as any remote device with an image stream. This uniformity of interface is what has made ROS an effective resource for so many robotics efforts [10]. Figure 2 illustrates a block diagram of our system's software components (available in our repository, as noted below).

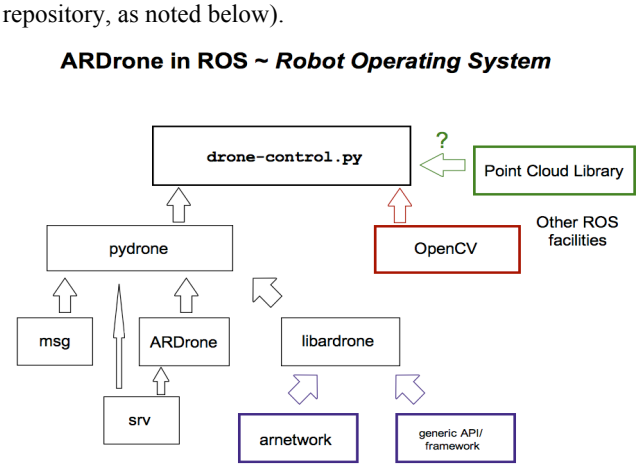

Figure 2. A block diagram of the drone platform's software components, provided and organized by Willow Garage's *Robot Operating* System, or ROS. OpenCV and pydrone are used by the **`drone-control.py`** file, providing state-machine control.

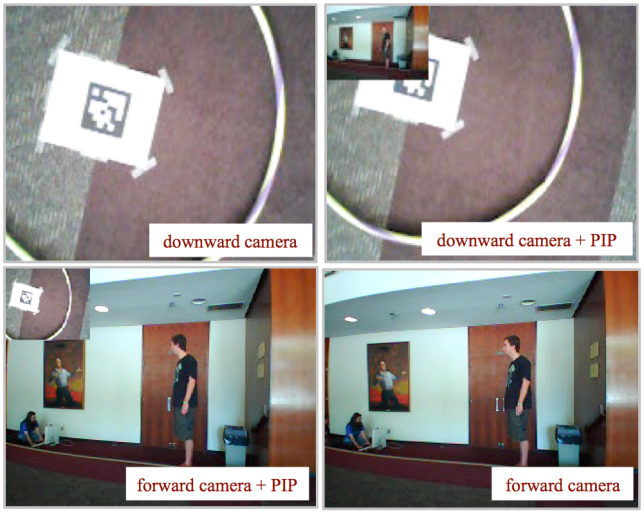

Figure 3. The four video-stream combinations available through our adapted pydrone drivers. In addition to images from the downward and forward cameras, each picture-in-picture (PIP) combination is also possible. Our tasks used either one or the other, instead of PIP.

It is noteworthy that no member of the team had prior experience with ROS or the drone. Even so, after two weeks of learning the software and hardware interfaces, the group felt comfortable enough with these resources to implement and investigate several aerial tasks, as the next section details.

III. Tasks Investigated

The drone's overhead point of view is one of its defining – and most compelling – characteristics. Thus, we first sought to use the drone to *support* the navigation of a ground robot by evaluating obstacles that the ground robot could not fully discern. We then evaluated the drone's ability to undertake point-to-point navigation using both its downward-pointing and forward-pointing cameras. Finally, we addressed a crucial limitation to aerial robots in uncontrived environments: their lack of odometry and precise localization. We show here that a vision-based approach to localization can provide excellent information about the drone's position using only the forward-facing onboard camera.

A. *Support of ground-vehicle navigation*

Figure 4 shows snapshots from a traditional ground-robot navigation task. An unmodified iRobot Create can only detect obstacles with its front bumper: as a result, if it encounters a wall (the bright brinks), it can not know which direction to turn in order to more efficiently continue its path. The default programming of the Create takes a wall-following approach, i.e., it hugs the obstacle until it is clear or until another directive supersedes the wall-following.

We used the drone, hovering overhead, to take images of this ground robot and any obstacles it encountered. The walls were distinctly colored to facilitate the image processing, and the Create carried an exclamation-point-shaped landmark to enable the computation of its position and orientation in the groundplane. Figure 4 shows the interacting state-machine controllers for each of the Create and the drone. Briefly, the Create no longer requires a wall-following subroutine: it simply moves forward until encountering an obstacle, at which time it waits for direction from the drone. The drone, meanwhile, follows the Create while it travels. When the overhead view is needed, the drone's images are segmented to determine whether the Create has more space to navigate on its left or its right. This message is passed on to the Create, which again takes the lead for the team. Figure 4's bottom panels illustrate the image-processing used in locating the Create and determining which direction to send the Create, when blocked.

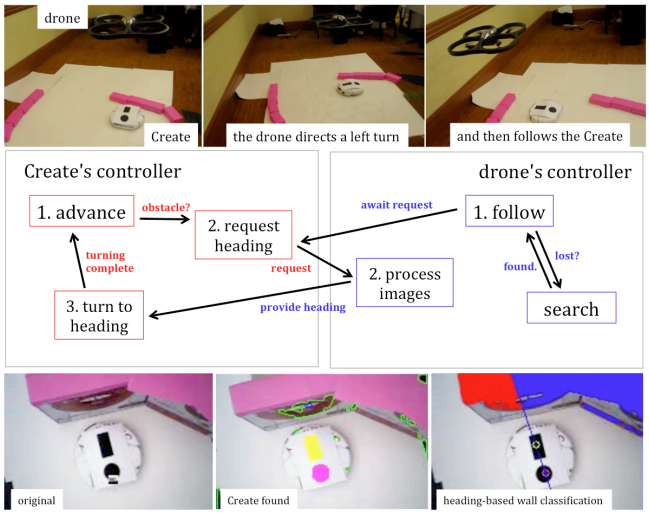

Figure 4. (top) snapshots from a task in which the drone provides aerial sensing support for ground-vehicle navigation: when the Create encounters an obstacle, the drone determines the heading of the closer edge of the wall. (middle) the interacting state-machine controllers for the drone and Create (bottom) the task-specific image processing in which the Create's landmark is found in order to find its pose; pose-specific segmentation of the wall then determines the correct heading.

Results In addition to success in our lab, this proof-of-concept task was demonstrated at 2011's Global Conference on Educational Robotics [11], the venue of the *BotBall* competition's finals. There, several handoffs of control from the drone to the Create were exhibited, including a three-minute run in which the walls had to be repositioned several times in order to keep both robots within the exhibition area. Figure 4 uses several snapshots from those runs.

Lessons learned The most challenging portions of this navigation-support task were (1) ensuring the drone could maintain its downward view of the Create as it hovered or followed behind it and (2) extracting and reasoning about the exclamation-point landmark designed for this task. Both of these challenges stem, in part, from the limited field of view and limited resolution of the downward camera. In response to these problems, the following navigation tasks first tried more distinctive landmarks and then used the forward camera in support of the drone's own navigation.

B. *Point-to-point navigation tasks*

Because the drone does not offer even rough relative odometry, point-to-point navigation is far more challenging on the drone than on ground robots such as the Create. To test

whether point-to-point tasks were feasible at all, we implemented two instances: for one, an autonomous room-and-hallway navigation routine that used landmarks and the drone's forward camera. Figure 5 (top) summarizes this routine. The second was a one-segment "hop" from the hula hoop at right in Figure 5 (middle) to the one at left in that image.

The hoops are only for human observers: to mitigate the difficulties with the exclamation-point landmark in the previous task, this point-to-point hop used April Tags [12] in the middle of each hoop as its basis for localization. Integrating the AprilTag library, which is in Java, into our ROS-based C++ and Python codebase was handled without a problem by making external calls. Figure 5 depicts the image processing, the state machine, and stills from one of the hops.

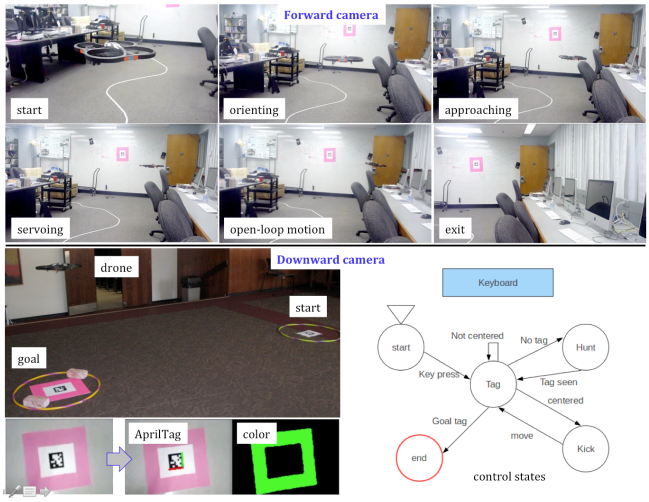

Figure 5. (top) stills from a point-to-point navigation task using the drone's forward camera and the color landmark indicating the door to the right (bottom) the point-to-point "hop" using AprilTags to designate locations. If it drifts away, he drone hunts for its start tag, and centers on it. The relative locations of the tags are known so that, once localized, the drone can "kick" in the correct direction toward the next tag. It then repeats the process until the goal tag is located.

Results and lessons learned The point-to-point hop turned out to be considerably more difficult than the navigation based on the forward camera. With this task, the drawbacks of using the downward camera for pose estimation became much clearer: both deliberate and accidental drone motions cause huge changes to the downward camera's field of view. With completely new image frames possible with each timestep, servoing a position or orientation error to zero is very difficult, and the system spends much of its time re-acquiring the landmark of interest: the state shown in the leftmost bottom frame of Figure 5.

In developing such applications, sliding-scale autonomy is crucial. Our interface allowed the human observer to change the robot's current state and, by default, required the drone to *ask* an operator before it was permitted to change state. This layer of control made interacting with the system – and debugging it – much more efficient.

These two examples made it clear that the forward camera would provide a better basis for long-distance, multiple-step navigation tasks. They also made clear than the crucial component needed to support such tasks is robust localization. The final investigation asked whether we could replace contrived landmarks such as Figure 5's in favor of the natural textures within the field of view of the drone's forward camera.

C. *Localization within image-based maps*

Without odometry, aerial autonomy is even more dependent on sensor-based localization than ground robots. The shared autonomy and point-to-point tasks, though successful, were *least* robust in their systems' pose-estimation of the drone relative to the task at hand. Thus, our final task focused on designing and evaluating a robust vision-based localization system. To avoid relying on contrived landmarks, we used SURF feature-matching as the basis for our approach [13], relying on OpenCV's implementation of SURF descriptors and approximate nearest-neighbor matching for the basic building blocks of the algorithm.

Appropriate to a vision-only platform, the drone's map of the environment was purely visual. At each point of interest (one node in a graph of locations) we took, by hand, 12 images in roughly 30° intervals. Because the field of view of the camera is 60°, the resulting images overlap to form a complete visual panorama making up the visual representation at that node. A four-location map thus consists of 48 distinct poses: 12 possible orientations from each of the four nodes in the map. Figure 6 shows an example graph, its images, and the image-matching using SURF features. Note that we did not stitch the 12 images together into a cylindrical panorama; rather, we extracted the SURF features from each image and allowed that set of features to represent the span of orientations within the image's field of view at that node in the map.

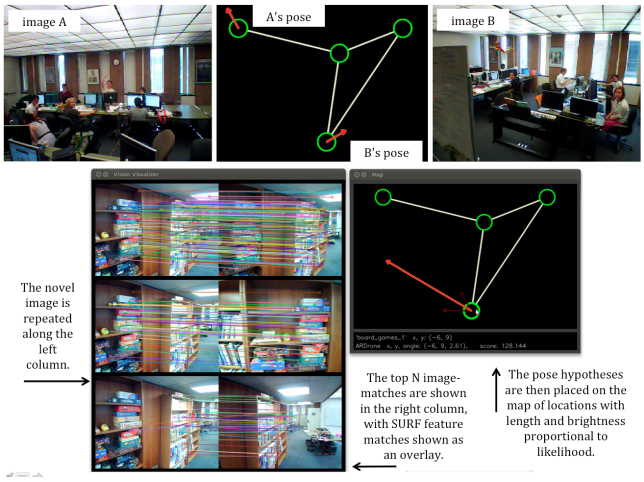

Figure 6. (top) two example images and their locations in the graph – note that these two have no overlapping field of view, but there are many similar features between them in this lab environment (bottom) A novel, unmapped image is shown adjacent to its three best map images with SURF-feature matches shown. The estimated poses appear with length and brightness proportional to their likelihood.

With the map constructed, the localization algorithm proceeds as follows:

(1) The drone acquires a new image from its forward camera, and the SURF features are extracted from that image

(2) For SURF feature in the acquired image, the closest match is found in each of the map's images

(3) Based on features' matches, each of the map images earns a "likelihood" score. We describe and contrast scoring approaches and filters for feature matching below.

(4) The map image with the highest overall score is declared the current location of the drone.

Runtime concerns Thus, each map image is being considered as a possible match for each novel image grabbed by the drone. On our maps of 36 or 48 images this image-matching ran at 2-3 frames per second, including visualization. As it stands, however, scoring *every* map image will not scale up efficiently to larger maps. However, a Markov- or Monte-Carlo-localization approach [14,15] will cull all but small subgraphs of possibilities out of a much larger map. These larger systems would then use the above image-matching as their innermost subroutine. Our timing results thus provide insight into *the extent* of pose-culling needed in order to run smoothly.

Scoring images Step (3), above, allows for several possibilities for defining the *score* of each stored map image with respect to a novel image of the environment. In addition, it is possible to filter the SURF feature-matches based on their image contexts. We tested all combinations of two scoring functions with the presence or absence of two filters in evaluating our localization algorithm. First, the *simple* scoring function returns the count of SURF features in the map image that had a match in the novel image within a Euclidean-distance of 0.1, measured in the 128-dimensional SURF-descriptor space (not in the image's pixel coordinates!) Because the simple scoring did not distinguish excellent SURF matches from borderline matches, we also implemented a *scaled* scoring metric that summed a value inversely proportional to the SURF-descriptor difference between the best matches.

In addition to using both scoring functions on all matches in each map image, we tested two filtering strategies: one using a ratio-distance threshold and one using only bidirectional best-matches. The ratio-distance, proposed in [16], keeps only matches in which the first-nearest neighbor (*fnn*) is significantly nearer to the query feature than the second nearest-neighbor (*snn*). We used a threshold of 0.4 in the ratio of fnn/snn. The bidirctional match test used only features in which the query feature and the map feature in question were each the best match of the other. In feature-poor images, this constrains each query feature and each map feature to *only one* best match. Figure 7 summarizes these scoring functions and filtering techniques. Figure 8 shows characteristic successful and unsuccessful results of the image-matching. The unexpectedly strong results there for some of the feature-poor images motivated the bidirectional best-matches filter.

Figure 7 also reports the accuracy of the localization results across all combinations of filter use and scoring strategies. The "position" data reflect the percentage of novel images whose node in the graph was correctly identified and the "orientation" data reflect the percentage whose absolute angle was within our resolution of 30°. The images come from two different data-collection passes for which ground-truth was known. The first pass involved taking images at 12 orientations in each of four nodes (those of the graph depicted). The second pass involved imaging the same environment at a different time of day and at orientations and locations that did not precisely match the originals (they were hand-collected). The data sets and the full source code tree are available at our project's repository at https://svn.cs.hmc.edu/svn/robotics-2011/.

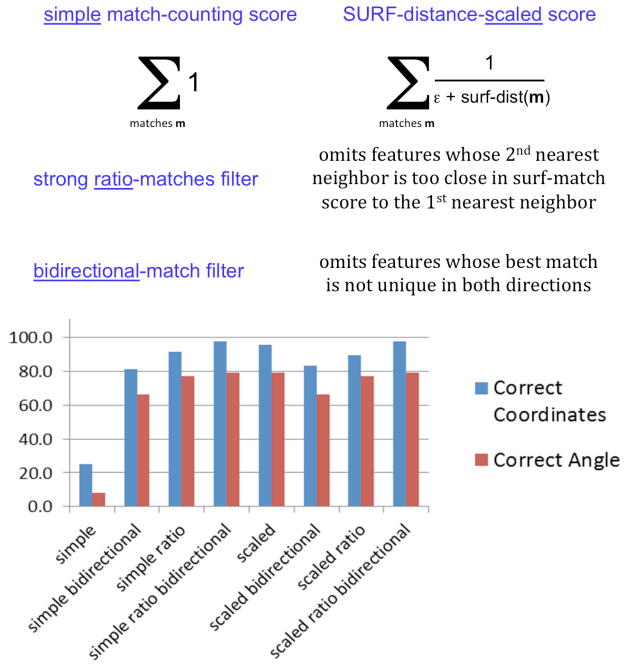

Figure 7. summary of the scoring functions and filters used in the markerless image-based localization using SURF features

Results and lessons learned It is noteworthy that the choice of scoring function was far less important than filtering the SURF features so that only the most likely correct matches were used. In fact, any one of the improvements over the simple match-counting scoring function yielded the greatest gains in localization accuracy. The 97.9% correct figure for location results from mis-localizing only one of the 48 images. The 80% accuracy in orientation estimation results from 10 mismatches from the correct orientation. The results are even better than this very conservative figure would suggest, however, because in 9 of the 10 missed cases, the orientation was estimated as a single 30° increment away from the correct orientation. In practice, this level of accuracy (a 90° range of possibilities, instead of a 30° range) is almost as useful: the point-to-point motions need to be able to handle even larger unmodeled displacements because of the jostling of the drone.

Thus, image-based localization without landmarks – at least within a small map of about 50 possibilities – offers a powerful foundation for implementing tasks that aerial autonomy can support: environmental surveillance, sensing support of ground robots, or safe, independent point-to-point navigation in advance of a shared-autonomy task at a distance.

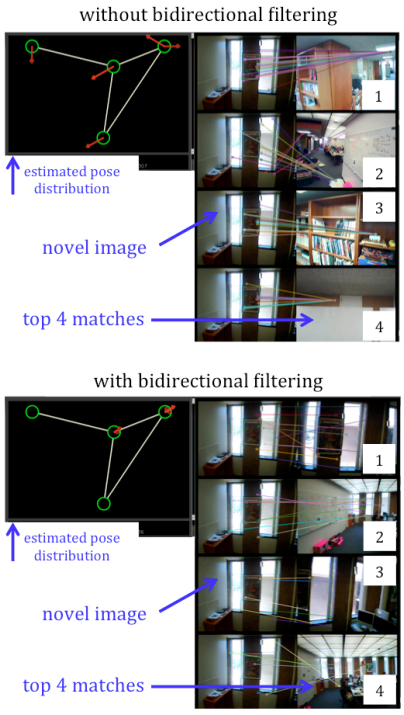

Figure 8. two more sets of results showing the improvement obtained with the bidirectional-match filter. Details appear in the text.

IV. PERSPECTIVE

It is remarkable that a complete set of resources for researching autonomous aerial robotics is now available for fo the cost of a $300 ARDrone (as long as a wireless-enabled netbook or desktop is already available). Indeed, equipping a Create with two cameras and wifi communications costs considerably more! Yet more important than the cost is the expense in *time*: the hardware, software, and algorithms presenting in this paper were integrated without the kinds of steep learning curves that require graduate-level, *i.e.*, full-time, investment. The undergraduate team that implemented the fundamental AI robotics of localization and navigation (with hand-built maps) came to the project with no prior experience in ROS, the drone, or the algorithms involved.

Thus, as a *practical* resource for AI and autonomous investigations, aerial robots are now accessible to an unprecedently large audience. As further evidence of this, the Kiss Institute for Practical Robotics has begun offering an *Autonomous Aerial Vehicle Contest* using the drone and its custom control computer, the CBC [17].

The accessibility of the hardware and software, in fact, presents an opportunity for educators: there are comparatively few resources that offer *curricular* support for aerial robotics. The ROS middleware, in fact, opens the possibility for even more: it serves equally well as an integrated interface to ground platforms, sensors such as the Kinect, and a wide variety of simulators and visualization tools. Thus, both educators and researchers of all stripes can now combine traditional ground-vehicle path-planning and state-machine control with vision-guided aerial vehicles in autonomous teams. We hope that with the algorithms presented here, we can help expand the role of aerial robots as captivating and accessible resources for both research and education in autonomous artificial intelligence.

ACKNOWLEDGMENTS

The authors acknowledge and thank the National Science Foundation (REU CNS-1063169 and CPATH #0939149), The Rose Hills Foundation, and Harvey Mudd College for their support of this work, along with our reviewers' suggestions!

REFERENCES

[1] D. Schneider. 2011. "Drone Aircraft: How the Drones Got Their Stingers," IEEE Spectrum 48(1):47-50 (January 2011)

[2] L. G. Weiss. 2011. "Autonomous robots in the fog of war," IEEE Spectrum 48(8):30-57 (August 2011)

[3] P. Abbeel, A. Coates and A. Y. Ng. 2010. "Autonomous Helicopter Aerobatics through Apprenticeship Learning," International Journal of Robotics Research 29(13) November, 2010, pp. 1608-1639.

[4] D. Mellinger and V. Kumar. 2011 "Minimum Snap Trajectory Generation and Control for Quadrotors," Proceedings, Int. Conf. on Robotics and Automation, Shanghai, China, May 2011.

[5] D. Mellinger, Q. Lindsey, M. Shomin, V. Kumar. 2011. "Design, Modeling, Estimation and Control for Aerial Grasping and Manipulation," Proceedings, Int. Conf. on Intelligent Robots and Systems (IROS '11), Sept. 2011.

[6] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts. 2005. "Combined optic-flow and stereo-based navigation of urban canyons for a UAV," Proceedings of the Int. Conf. on Intelligent Robots and Systems (IROS Aug. '05) pp.3309-3316.

[7] C. Bills, J. Chen, and A. Saxena. 2011 "Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues," Proceedings of the International Conference on Robotics and Automation (ICRA 2011).

[8] G. Bradski and A. Kaehler. 2008. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, publishers, Sebastopol, CA.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. 2009 ROS: an open-source Robot Operating System," Proceedings, ICRA '09 Workshop on Open Source Software, May 12-17, 2009 Kobe, Japan, IEEE Press.

[10] S. Cousins. 2011. "Exponential Growth of ROS," IEEE Robotics and Automation Magazine, pp. 19-20, March 2011.

[11] K. Sheely, S. Matsumoto, M. Sok, N. Berezny, L. de Greef, J. Vasquez, D. Lingenbrink, B. Jensen, and Z. Dodds. 2011 "Coordinated Navigation with Aerial/Ground Robots," *Proceedings of the 2011 Global Conference on Educational Robotics, Robot Exhbition*, July 9-13, Irvine, CA.

[12] E. Olson, 2010. "AprilTag: A robust and flexible multi-purpose fiducial system," Technical Report, University of Michigan APRIL Laboratory, May 2010.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. 2006. "Surf: Speeded up robust features," Proceedings of the European Conference on Computer Vision (ECCV '06), pp. 404-417.

[14] D. Fox, W. Burgard, and S. Thrun. 1999. "Markov Localization for Mobile Robots in Dynamic Environments," Journal of Artificial Intelligence Research, 11, pp. 391-427.

[15] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. 1999. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," Proceedings, Conference of the Association for the Advancement of Artificial Intelligence (AAAI '99), pp. 343-349.

[16] M. Brown, R. Szeliski, and S. Winder. 2005 "Multi-Image Matching using Multi-Scale Oriented Patches," *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition (CVPR05)*, June 20-26, 2005, San Diego, CA, pp. 510-517.

[17] KIPR autonomous aerial vehicle contest website, accessed 11/29/2011, at http://kipr.org/2011-kipr-autonomous-aerial-vehicle-contest