Person-following UAVs

Francisca Vasconcelos
Torrey Pines High School
frannycisca@gmail.com

Nuno Vasconcelos
University of California San Diego
nuno@ece.ucsd.edu

Abstract

We consider the design of vision-based control algorithms for unmanned aerial vehicles (UAVs), so as to enable a UAV to autonomously follow a person. A new vision-based control architecture is proposed with the goals of 1) robustly following the user and 2) implementing following behaviors programmed by manipulation of visual patterns. This is achieved within a detection/tracking paradigm, where the target is a programmable badge worn by the user. This badge contains a visual pattern with two components. The first is fixed and used to locate the user. The second is variable and implements a code used to program the UAV behavior. A biologically inspired tracking/recognition architecture, combining bottom-up and top-down saliency mechanisms, a novel image similarity measure, and an affine validation procedure, is proposed to detect the badge in the scene. The badge location is used by a control algorithm to adjust the UAV flight parameters so as to maintain the user in the center of the field of view. The detected badge is further analyzed to extract the visual code that commands the UAV behavior. This is used to control the height and distance of the UAV relative to the user.

1. Introduction

Recent years have witnessed an explosion in consumer unmanned aerial vehicles (UAVs), known as drones. Particularly interesting is the problem of person-following UAVs, i.e. UAVs that can be programmed to follow a user. This has many interesting applications. For example, a UAV could track an athlete, acting as a "personal camera man." The resulting video could be used for entertainment (a football game shot from the quarterback's perspective), studying athletes' performance (recording a soccer player's dribbles and field position), etc. In search and rescue or firefighting, a drone could hover above a first responder or fireman, providing an expanded view of the scene. In the context of assisted living, it could follow an elderly person, producing an alarm if the person falls. Finally, in a child safety scenario, a drone could escort a child to or from school.

As drones decrease in size, many of these applications could be implemented almost seamlessly, e.g. by fly-sized UAVs. Currently, person following is only possible by tracking user coordinates with GPS and cellphones [1]. In addition to the well known unreliability of this solution indoors and in disaster relief scenarios, the applications above require precise control of the relative positions of drone and subject being tracked. The drone must sometimes stay in front, other times behind, and sometimes above. In each case, it may have to stay directly aligned with the subject, e.g. immediately above the firefighter, or at an angle, e.g. 45° above the soccer player, and track the subject from far or close. For example, the "personal cameraman" should stay close to record soccer dribbling technique and far to record player positioning. Such precision in the control of its position requires the UAV to precisely locate the user in the scene, understand which way the user is facing, etc. The UAV should also be able to figure out what to do by simple visual inspection of the scene, e.g. through analysis of user gestures or other form of visual input. Ideally, it would even be "programmable" by manipulation of physical world objects, as this would place the technology at the reach of users of all ages and technical skills.

While these goals could, in principle, be achieved by endowing drones with computer vision, previous work in robotic vision has emphasized autonomy, namely autonomous navigation based on simultaneous localization and mapping [11, 4], visual odometry [26, 23, 18], and obstacle avoidance [28, 24]. These, however, are not major requirements for the applications above where, rather than full-blown autonomy, the goal is to follow a person *robustly,* independently of the person's pose, time of day, etc. Similarly, complex user interaction, such as gesture [8] or emotion recognition [9] is not critical for person-following robotics. In this context, human-robot interaction reduces to simple "drone behavior programming" commands, such as specifying whether to collect video from the person's front or back, the person-following angle and distance, and simple "virtual fence" commands preventing drone access to restricted areas. This programming should be possible by manipulation of simple visual patterns.

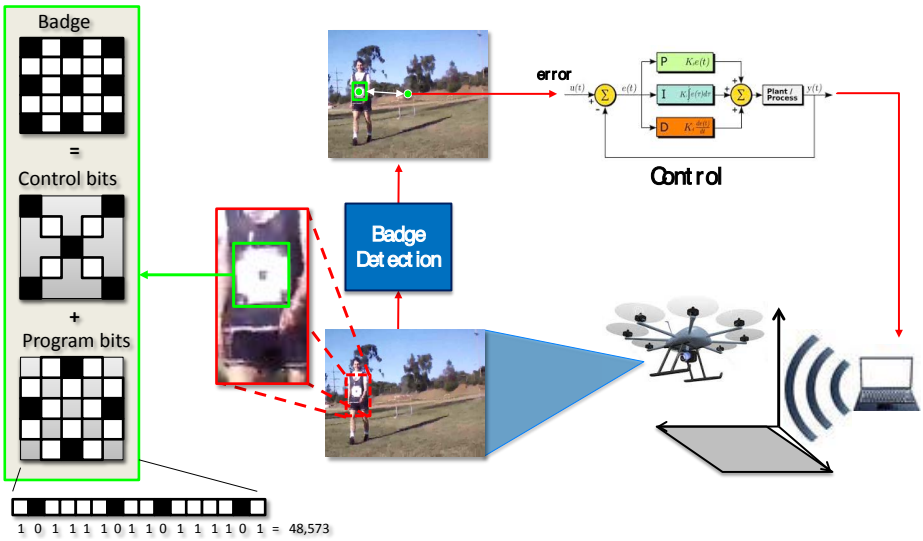

Figure 1: Vision-based UAV control architecture. The user wears a badge, which is detected and localized in real-time. The differences between the badge position/size and pre-specified target values act as error signals for a control algorithm. This issues drone flight commands, so as to align badge position and size with target values. The badge displays a control (control bits) and a behavior program (program bits) pattern. The former is used for badge detection, the latter (binary code for $48,573$ in this example) to specify drone behaviors.

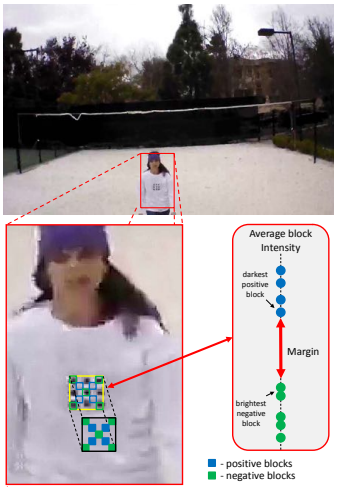

Figure 2: Margin similarity. The average intensity ξ_i of each badge square is computed. Blocks corresponding to white (black) squares are denoted positive (negative). The margin is the difference between the ξ_i of the darkest positive and brightest negative blocks.

In this work, we propose a computer vision-based UAV control architecture for robust person-following that supports behavior programming by manipulation of visual patterns. The architecture is illustrated in Figure 1. It achieves the two goals via the introduction of a programmable visual badge, to be worn by the user. The badge, which is inspired by QR codes, depicts a visual pattern with two components. The first (control bits) is fixed and used to locate the badge. The second (programming bits) is variable, implementing a code used to program the drone behavior.

A biologically-inspired computer vision architecture, composed of two saliency mechanisms, is then proposed to detect and localize the badge in the visual scene. This includes salient point detection, a novel image similarity measure, and an affine validation procedure. The badge location is fed to a control mechanism that adjusts the drone flight so as to maintain the badge in the center of the field of view. The located badge is further analyzed to extract the behavior programming code, which determines parameters such as drone altitude and distance to the user. Experiments show that the proposed architecture enables a drone to follow a user in real-time, over a range of scenes, distances, and lighting conditions. In fact, by displaying the badge on a cellphone screen, it is even possible to track the user when there is no ambient light. Furthermore, with the proposed behavior programming, the drone can follow and capture footage of the user over a range of distances and angles.

2. Related work

Robotic vision has mostly focused on autonomous navigation, namely problems such as simultaneous localization and mapping [11, 4], visual odometry [26, 23, 18], and obstacle avoidance [28, 24]. Autonomous navigation is not critical for person-following, which raises two other vision problems. The first, *vision for control,* addresses the localization and tracking of the person. The second, *vision for behavior programming,* uses visual properties of the tracked object to determine the person-following behavior.

Vision for control is closely related to object recognition [21, 32, 10, 14, 13]. Despite impressive recent gains in deep learning [19], this is still a difficult problem, especially in the presence of wild variations of pose and lighting and under real-time constraints. Object recognition can also place restrictive constraints on person-following. For example, face detection [32] does not support person-following from behind. Vision for behavior programming is less related to standard computer vision problems, although it could be framed as action recognition [30, 33, 20]. For example, a pedestrian detector [10, 12, 7] could be used for person-tracking and gesture analysis [35, 25] for behavior programming. However, these operations are difficult to implement in real-time, especially when the relative positions of user and drone vary significantly. Gesture vocabularies are also cumbersome to learn and it can be difficult to discriminate between more than a few gestures.

In summary, while computer vision methods could be applied to person-following, it is not clear that they could withstand the variations of lighting and pose of this problem or meet its real-time constraints. In this work, we address these problems by 1) restricting the object to follow to a badge and 2) proposing a novel badge following procedure inspired by biological vision, namely the *attention*

mechanisms that direct human vision to *salient* regions of the visual field [6]. This is inspired by psychology literature showing that saliency is a function of both the visual scene and the task executed by the human subject [36, 31, 34]. The scene-based component, known as *bottom-up saliency,* is due to the fact that regions of the visual field different from the background scene draw our attention. Mathematical models of saliency account for this with center-surround differences [17, 15, 6], modeling saliency as the difference between the feature responses of one region and those of the surrounding neighborhood. The task-dependent component, known as *top-down saliency,* reflects the fact that saliency computations take into account the goals of high-level areas of the brain. For example, while a subject asked to estimate people's ages will mostly fixate on their faces, he will instead fixate on their clothes when asked to estimate people's wealth [36]. We propose an attention mechanism based on the Harris corner detector [16] for bottom-up saliency and a procedure inspired by the Viola-Jones face detector [32] and the the local binary pattern (LBP) descriptor [29] for top-down saliency.

3. Proposed approach

We now discuss the proposed UAV control architecture.

3.1. Person-following architecture

The proposed person-following architecture is summarized in Figure 1. Video is captured by the UAV-mounted camera and transmitted, via Wifi, to a laptop[1]. A computer vision module is used to detect a badge worn by the user. The badge's location in the image is then determined, a bounding box placed around the badge, and two measurements made: 1) the distance from the center of the box to the center of the image and 2) the size of the box. They are fed to a control module which issues motion commands to the UAV. The UAV responds with movements that align the bounding box position and size with a reference target. This target is programmable by varying the appearance of the badge, with different badge appearances corresponding to different drone behaviors. Current drone behavior parameters are target location and size and drone altitude, controlling drone position relative to the user.

Figure 1 shows the badge used in this work. This is a distinctive object, which can be affixed to different locations of the human body to support different drone following behaviors. As shown in the figure, it enables the simultaneous solution of the two vision problems of person-following, through the combination of a fixed and a programmable visual pattern. The fixed component (denoted as "control bits") is used to solve the control problem. The programmable component ("program bits") instructs the drone on how to behave. By adopting a sufficiently rich visual pattern, the behavior vocabulary can be as large as desired. Finally, since the badge can be active, e.g. displayed through a cellphone screen, it is even possible to perform person-following in the absence of environment light (e.g. at night) or to change the behavior program in real-time.

In our implementation, the badge displays the *visual code* shown on the left of Figure 1. This consists of 25 bits, implemented with white or black squares. From top-left (bit b_0) to bottom-right (bit b_{24}), these bits specify the binary code $c = (b_{24}, \ldots, b_0)$. Bit b_i has value 0 (1) when the square is black (white). The two vision modes are supported by combining two *bit-patterns*. The 9 squares in the 'X' configuration shown in the middle of the badge inset are bits *reserved* for control purposes. The UAV detects this fixed pattern to follow the person. The remaining 16 squares form a variable binary code, used for programming UAV behavior, supporting $2^{16} = 65,536$ behaviors.

There are several technical challenges associated with the implementation of this architecture. First, the difficulty of controlling a dynamic system, such as a UAV, increases substantially as the number of scene measurements decreases and/or there is delay. To be effective, the computer vision system must operate in *real-time*, ideally at $30-40$ frames per second. Second, tracking the badge over a reasonable *range of distances* requires high-resolution images and badge detections over multiple image scales. In this work, we consider 720×1280 pixel images and 50 badge sizes. Under the standard sliding window paradigm, this implies the vision system needs to classify 1.8 trillion windows per second. Third, the UAV frequently operates in cluttered scenes, against complex backgrounds, under a *wide range of conditions*, from over-exposed outdoors scenes to low-lit indoor environments, or even in nocturnal scenes with little artificial light. Fourth, UAV movement can induce significant image blur. Due to all these factors, badge detection is far from trivial. For example, our preliminary experiments showed that the OpenCV implementations of the Viola-Jones face [32] and HoG pedestrian detector [10] were a few orders magnitude slower than needed for this application and too sensitive to motion blur. For deep learning [19], the real-time constraint would be an even larger challenge.

3.2. Biologically-inspired badge detection

We propose a biologically-inspired architecture, using an attention mechanism composed of bottom-up and top-down saliency modules.

3.2.1 Bottom-up saliency

A first attention stage implements a bottom-up saliency detector based on the Harris interest point operator [16]. This

[1] All processing could in principle be done by the UAV. However, we had no access to the CPU of the Parrot AR.Drone 2.0 used in this work.

exploits the fact that the badge is dissimilar from natural backgrounds because it contains an unusually large number of corners. Since the Harris operator responds strongly to corners, it elicits a strong response in the badge area. To measure the density of interest points, the score map $H(u, v)$ produced by the Harris detector, where (u, v) are the pixel coordinates, is thresholded. A square window $\mathcal{W}(u, v)$, with $1/20^{th}$ of the image width, is then placed around each image position and the number of salient points counted with

$$s(u, v) = \sum_{(p,q)\in\mathcal{W}(u,v)} u[H(p, q) - T_h], \quad (1)$$

where $u(x) = 0$ for $x < 0$ and $u(x) = 1$ otherwise. This operation is performed efficiently with integral images [32]. The threshold T_h is chosen adaptively so as to guarantee that the number of saliency peaks surviving the bottom-up stage is less than 0.5% of the image size. This is critical to achieve the desired frame rates. In general, the saliency peaks of (1) cover the badge region. However, there are also many peaks in the background, due to areas of clutter.

3.2.2 Top-down saliency

Salient regions are passed to the top-down attention mechanism. This is a classifier

$$h(x) = sgn[\phi(w, x) - T_a], \quad (2)$$

where x is the image patch extracted around an image location, w the control-bit template shown in the badge inset of Figure 1, $\phi(w, x)$ a measure of similarity between w and x, such as the dot-product $< w, x >$, and T_a a threshold.

For computational efficiency, the similarity function exploits the fact that the control pattern is a sum of 9 box-like Haar wavelets

$$w = \sum_l a_l h^{(l)} \quad (3)$$

where $a_l \in \{-1, 1\}$ is the coefficient of wavelet $h^{(l)}$ and $h^{(l)}$ a binary function that identifies one of the 9 control squares. The dot-products

$$\xi_l =< h^{(l)}, x > \quad (4)$$

between each of the Haar wavelets and the patch x are first computed. These are proportional to the average image intensity of x inside each square. As in [32], each is evaluated with four additions, using integral images. The similarity function then reduces to

$$\phi(w, x) = \gamma(a, \xi) \quad (5)$$

where $\gamma(a, \xi)$ is a measure of similarity between the vector a of wavelet coefficients a_l of the template w and the vector ξ of wavelet coefficients ξ_l of the patch x. The evaluation complexity of $\phi(\cdot, \cdot)$ is equal to that of $\gamma(\cdot, \cdot)$ plus 36 additions, *independently* of the sizes of x and w. This is critical to enable the computation of (2) for many badge sizes in real time. For example, when $\gamma(\cdot, \cdot)$ is the dot product, (2) requires 45 adds and 9 multiplies per patch x.

The detection is repeated for multiple sizes of the patch x. For each size, the threshold T_a is set to the maximum value of the score $\phi(w, x)$, if this is positive, and zero otherwise. This guarantees that at most one badge location is selected per size. The size of highest score is finally selected, if this score is positive. Otherwise, it is concluded that there is no badge in the video.

3.2.3 Similarity functions

Initial experiments, using the dot-product as similarity function $\gamma(., .)$ produced somewhat disappointing results. This is due to the fact that, in this case,

$$\gamma(a, \xi) = \gamma_{dp}(a, \xi) = \sum_i a_i \xi_i = \sum_{i\in\mathcal{I}^+} \xi_i - \sum_{i\in\mathcal{I}^-} \xi_i \quad (6)$$

where $\mathcal{I}^+$ ($\mathcal{I}^-$) is the set of indices of the white (black) badge squares. Consider a vector of ξ_i that produces a mildly negative score, due to poor consistency between the white/black squares of w and the image intensities of x. Increasing a single ξ_i in the first summation (e.g. a very bright image region that aligns with a white block) can be enough to produce a high similarity score, even though the similarity is weak for all remaining blocks. Noting that (6) is a measure of the difference between the average image intensities of the blocks in $\mathcal{I}^+$ and those in $\mathcal{I}^-$, suggests the more robust similarity function

$$\gamma_m(a, \xi) = \min_{i\in\mathcal{I}^+} \xi_i - \max_{i\in\mathcal{I}^-} \xi_i. \quad (7)$$

This measure is illustrated in Figure 2. Denote the blocks of $\mathcal{I}^-$ (black badge squares) as negative and the blocks of $\mathcal{I}^+$ (white squares) as positive. As shown in the figure, (7) equates similarity to the difference between the less positive block (the darker among positive blocks) and the less negative block (the brighter among negatives). Since, in machine learning, this is called "the margin" for the classification of the blocks, we denote (7) as the *margin similarity function*. Note that, if the margin score is greater than zero, there is a threshold T such that $\xi_i > T$, $\forall i \in \mathcal{I}^+$ and $\xi_i < T$, $\forall i \in \mathcal{I}^-$. This implies that the image patch x can be thresholded so as to produce an exact replica of w. This property does not hold for dot-product similarity. Furthermore, the margin measure is insensitive to variations of image intensity that do not affect the less positive and negative blocks. For example, the intensity of the brightest and darkest blocks play no role in this measure. This increases robustness to lighting variations. These properties are similar to those of LBP [29]. We exploit them to produce a robust template matcher, rather than a robust image descriptor.

3.3. Affine validation and behavior programming

While the largest detection score ϕ^* across patch sizes tends to occur at the badge location, when there is a badge in the image, it is more difficult to determine badge absence. This is because low positive values of ϕ^* can be caused by both 1) background regions that resemble the control pattern, or 2) poor badge alignment (camera-badge orientation), motion blur, and video compression noise. To overcome this ambiguity, the badge detection is post-processed by an affine validation procedure with three steps.

First, the detected image patch is cropped and intensity normalized to $[0, 255]$, to compensate for scene illumination. Second, the patch is resized to 64×64 pixels and an affine transformation τ is used to bring it into alignment with a reference pattern p. This is a synthetic 64×64 pixel badge replica, with intensities 0 or 255 (as defined by the badge code), smoothed with a 5-tap Gaussian filter. Finally, the similarity score $\phi(\tau(x), p)$ between the affine warped patch $\tau(x)$ and the reference pattern is computed. In our experience, this is a very robust indicator of badge presence, taking positive (negative) values when the badge is visible (not visible). A badge detection is declared if $\phi(\tau(x), p) > 0$. Since this procedure only requires the computation of an affine transformation of one 64×64 patch, it adds negligible complexity to the detection.

This affine validation procedure can be extended, in a straightforward manner, to support behavior programming. In this case, a set of reference patterns $\{p_i\}$ are generated (one per badge programming code defined by the application) and the affine validation stage is repeated for each p_i. The pattern p_{i^*} of highest score

$$i^* = \arg\max_i \phi(\tau_i(x), p_i) \tag{8}$$

is finally selected and the corresponding behavior detected if $\phi(\tau_{i^*}(x), p_{i^*}) > 0$. Note that this operation requires the computation, per image, of a number of affine transformations equal to the number of programmable behaviors.

3.4. Backtracking to improve detection rate

The inclusion of affine validation makes the detector very robust. In all our experiments, it has not produced a single false positive. There can, however, be misses of two types. *Close misses* occur when the badge is detected at one patch size, but with a score weaker than a spurious detection at some other size. They tend to occur when the badge suffers moderate amounts of blurring or perspective distortion. *Far misses* are the cases where the true badge location is not detected for any patch size.

To correct for these misses, we implement two backtracking procedures. Close misses are addressed by an *affine backtracking* stage, which applies the validation procedure of Section 3.3 to the best patch of each size. This is usually sufficient to rerank the detections of different sizes so that the true badge location receives the top score. Far misses are addressed by discarding the detections of Section 3.2 altogether, and repeating the process for a *full badge detector*. This consists of repeating the steps of (2) to (7) for a template w that includes the complete badge (control + program bits) of Figure 1, rather than just the control pattern. In this case, there are 25 (rather than 9) blocks and the complexity increases to 125 adds and 25 multiplies per patch. If the applications involves behavior programming with C badge codes, the process must be repeated C times. Overall, this process has close to $3C$ times the complexity of the procedure of Section 3.2.

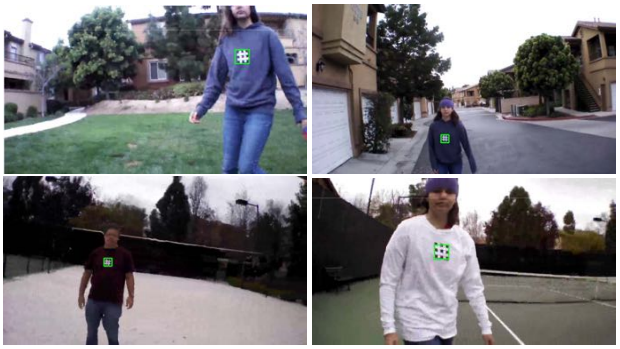

Figure 3: Example detections (green bounding boxes) in the badge detection dataset. Note the diversity of subject orientations, distance to the camera, and background clutter.

3.5. Badge tracking and drone control

Once detected, the badge is tracked in subsequent frames. The proposed tracker follows the tracking by detection paradigm [5, 22], applying the top-down saliency detector to each video frame to localize the badge. This is, however, constrained by a *focus of attention* (FoA) mechanism that limits the search area to a window around the previous detection. In our implementation, this window has roughly a quarter of the image size. The number of badge sizes is also reduced to 20, around the size of the last detection. The FoA mechanism exploits the fact that both drone and user are constrained by the laws of Newtonian physics, which limit badge motion between frames. Besides reducing tracking complexity, it prevents large jumps to false positives. It also leverages the robustness with which badge absence is detected. When a loss of track is declared, the FoA is disabled and the drone is ordered to hover until a new detection occurs, giving the user the opportunity to get back on frame.

The control loop is implemented with a proportional-integral-derivative (PID) controller [27, 3]. Given target $g(t)$ and measurement $m(t)$, at time t, the control is

$$\delta(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t), \tag{9}$$

Table 1: Average time (T) and detection rate (DR) of different detectors.

	1 code			
	dot product		margin	
	T (ms)	DR	T (ms)	DR
TD	407	68.5	369	68.1
TD + AB	422	87.2	396	86.6
FBD	736	58.6	922	86.6
FBD + AB	735	58.6	915	85.7
TD + FBD	601	66.5	572	90.1
TD + FBD + AB	591	66.5	589	**93.0**
	2 codes			
	dot product		margin	
	T (ms)	DR	T (ms)	DR
TD	393	68.1	360	68.2
TD + AB	447	82.8	412	87.3
FBD	875	46.0	1300	86.6
FBD + AB	853	46.0	1319	85.1
TD + FBD	685	64.7	700	89.8
TD + FBD + AB	682	64.7	741	**93.0**

where K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively, and $e(t) = g(t) - m(t)$ the measurement error. While the proportional term $K_p e(t)$ depends on the current error, the integral term $K_i \int_0^t e(\tau)d\tau$ depends on accumulated past errors, and the derivative term $K_d \frac{d}{dt} e(t)$ is a prediction of future errors, based on current rate of change. Three independent PID controllers were implemented, each acting on the drone speed along one of the 3 spatial directions - forward/backward motion, left/right motion, and elevation. Elevation was not controlled through visual measurements, using instead the drone altitude sensor and a target altitude set by the behavior program. The measurement signals for left-right and forward-backward motion are the location and size of the badge bounding box, respectively. The targets for these variables are specified by behavior programming. The control gains K_p, K_i, K_d were set by trial and error.

4. Experiments

Several experiments were performed to evaluate the robustness of the proposed UAV control algorithm.

Detection experiments: Badge detection accuracy was evaluated with a dataset of images recorded by a drone flying in outdoors scenes with substantial clutter. Several data collection flights were performed, from which about 10,000 images were collected. From these, we selected 312 of the most challenging images, covering a diverse range of distances and angles between user and drone, background scenes with substantial clutter (trees, buildings, etc.), and sometimes exhibiting significant amounts of motion blur and compression artifacts[2]. Some of the images in this *badge detection dataset* are shown in Figure 3. Note that video quality can decrease substantially depending on battery level, distance between drone and laptop, wifi interference from nearby devices, etc. An example of a mildly degraded image is shown in Figure 2. The dataset was manually annotated with badge bounding boxes.

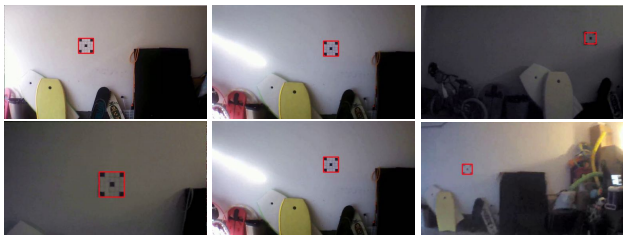

Figure 4: Badge detections over a set of images collected for different badge-drone distances and scene illuminations. Top: images collected at the same distance for different illuminations. Bottom: images collected at multiple distances.

Six detector configurations, involving combinations of the top-down detector (TD) of Section 3.2.2, the full badge detector (FBD) of Section 3.4, and the affine backtracking (AB) stage of the same section, were tested. These combinations are listed in Table 1. When more than one stage was used, the methods are listed in order of application. For example, the TD+FBD detector applied the TD detector first, relying on the FBD stage only when TD could not find the badge. The detections of all stages were affine validated, as discussed in Section 3.3. The detection performance of different algorithms was measured with the overlap criterion commonly used to evaluate object detectors [13]. This computes the ratio $O = \frac{A(G \cap D)}{A(G \cup D)}$, where $A(G \cap D)$ is the intersection area of ground truth and detection windows and $A(G \cup D)$ the area of their union, and accepts a detection if $O > 0.5$. Performance is summarized by the detection rate

$$DR = \frac{\#detections}{\#images}. \tag{10}$$

Table 1 summarizes the performance of the different methods in terms of both average run-time (T) per image[3] (in milliseconds) and average detection rate (DR) across the dataset. Results are shown for the two similarity functions (dot product and margin) of Section 3.2.3. The dataset includes images of badges with two different programming codes ($65,535$ as in Figure 1 and $48,573$ as in Figure 2). Performance was measured for two settings: 1) badge code known a priori, and 2) badge code unknown. In the first setting, the drone has a single behavior, which is preprogrammed. In the second, it can switch between two behaviors in real time. In this setting, the detection algorithm has to search over the two badge codes. We have not yet experimented with more complex scenarios, involving multiple codes. Code $65,535$ is, in our experience, the most challenging to detect, because it corresponds to a program bit pattern of all 1s. Since the associated visual badge program pattern consists of "all white" blocks, it is quite con-

[2] All datasets assembled for this work are publicly available at [2].

[3] All experiments were performed on a MacBook Pro 2015, 2.5GHz.

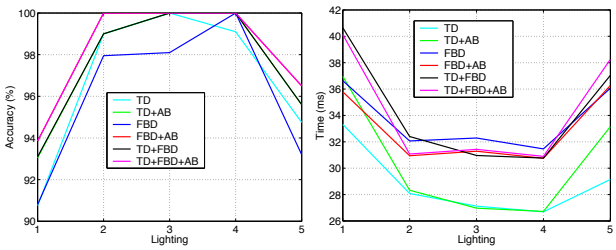

Figure 5: Tracking rate (left) and time (right) of the different detectors as a function of scene lighting condition.

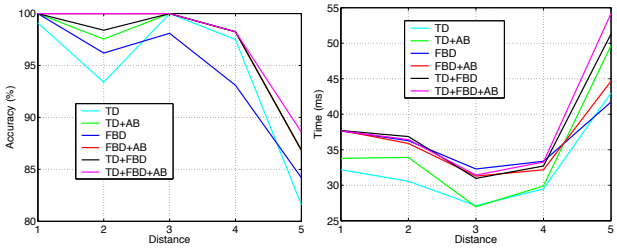

Figure 6: Tracking rate (left) and time (right) of the different detectors as a function of badge-drone distance.

fusable with large bright image regions, that appear in white walls, shirts, areas of sky, etc.

Several conclusions can be drawn from the table. First, the margin similarity function is vastly superior to the dot product. It improved the performance of nearly all detectors, with gains as large as 26.8% for one code, and 40.2% for two codes (FBD detector). Note that, for FBD, the dot-product substantially underperformed in the two-code scenario, where it tended to find regions of sky or walls as top badge matches. On the contrary, the margin measure proved very robust, achieving similar or identical results with one or two codes, for all detectors. Second, all components of Section 3.2 proved useful, with the TD+FBD+AB detector achieving the top DR of 93%. Examples of its badge detections are shown in Figure 3 and in [2]. For both 1 and 2 codes, the next best performance was the close to 90% of TD+FBD. The weakest performer was TD, although its performance increased substantially when combined with AB. Finally, FBD was the most complex method, almost tripling the complexity of TD. This was the fastest method, achieving speeds of 3 frames per second. TD+FBD+AB had intermediate complexity. Note that these are times for very high resolution images (720×1280) without the FoA mechanism of Section 3.5. They are not the processing times of the full system.

Tracking experiments: While rich in image artifacts, clutter, etc., the badge detection dataset does not allow a precise characterization of detection rate vs. variables such as distance and lighting (which are difficult to control outdoors). Furthermore, since it only contains images, it cannot be used to evaluate tracking performance, namely the impact of the FoA mechanism of Section 3.5 in tracking complexity. To overcome these problems, a *tracking dataset* was collected in a cluttered garage. The badge was hung on a wall with skateboards, wood panels, jump ropes, etc. lying below it. The drone was then moved in the direction parallel to the wall, at a nearly constant speed. To enable precise control of its distance to the wall, the drone was either rolled on an office chair or held by a person walking sideways. In the first case the badge was aligned with the camera, but in the second there was non-trivial in-plane rotation due to the relative motion of the person's hand and body. Figure 4 presents typical images from this dataset.

Data was collected to test tracking robustness to scene lighting and drone-badge distance. To evaluate robustness to lighting, the garage was illuminated by two light sources, sunlight from the left and a dim wall lamp to the right. Five lighting conditions were created by incrementally opening the garage door on the left. This is illustrated in the top row of Figure 4, which shows images collected under three of the lighting conditions. Under condition 1 (brightest, shown on the left) the garage door was fully open, allowing bright sunlight to cover the wall. This created a significant lighting gradient from left to right. Since the drone camera gain adjusts to the brightest scene patch, the image would at times be half dark and half bright. Under condition 5 (darkest, shown on the right), the only light source was the lamp. Intermediate conditions sometimes produced very uneven light patterns (e.g. the image in the center) due to flow of light through creaks in the semi open garage door. In these experiments, the drone was kept at a constant distance of 9 ft from the badge.

To evaluate robustness to distance, the drone was placed at five distances from the badge, ranging from $5-13$ ft with increments of 2 ft. This is illustrated in the bottom row of Figure 4, where the leftmost image shows the shortest (condition 1) and the rightmost image the largest (condition 5) distance. As distance increased, the badge appeared smaller and more clutter filled the frame. At smaller distances, the badge moved much faster between frames.

A set of 10 videos were collected for the 5 light conditions (one for chair, one for hand motion) and the 5 distance conditions. Since one of the conditions (intermediate light and distance) was shared by the two experiments, this lead to a total of 18 videos. These were divided into 132 video clips of 450 frames each. Bounding boxes were manually produced for keyframes 60 frames apart (8 keyframes per video), for a total of 1056 ground truth frames. Tracking accuracy was measured by using the DR criterion of (10) on these images. As before, we compared the performance of the six detectors of Table 1, all using the FoA mechanism of Section 3.5 and the margin similarity measure.

Figure 5 shows the detection rate (left) and time (right) of the detectors as a function of scene lighting. As before, TD+FBD+AB achieved the best performance. The combi-

Figure 7: Experiments on the overall performance of the UAV control algorithm. Top: the UAV follows a user as she moves in 3D space. Bottom: The user successively displays two badges that command the UAV to move closer or farther away. In all images, a green (yellow) ellipse is used to indicate the position of the drone (badge). Videos of these experiments are available at [2].

nations TD+AB and TD+FBD were the next best performers. Again, FBD was the weakest detector. Note however that, with a tracking rate always above 90%, all detectors are quite insensitive to scene lighting. Detection examples are shown in the top row of Figure 4. Close inspection revealed that tracking was only lost for extreme lighting conditions, e.g. when the camera gain control renders the badge (located on the darker half of the wall) nearly imperceptible. Figure 6 shows detection rate (left) and time (right) as a function of drone-badge distance. It confirms the previous observations, with TD+FBD+AB achieving the top performance, nearly matched by TD+AB and TD+FDB, and FDB the weakest detector. However, the differences are now more significant. All methods are less robust when the badge is farthest away, because much more background clutter is visible and the tracker sometimes locks on background objects. The gains of TD+FBD+AB over FBD are now substantial (from 82% to 89%).

Overall, the TD+FBD+AB configuration achieved the best performance, proving to be quite robust to lighting variations and scene clutter. Figures 5 and 6 also show that the FoA mechanism is very effective at reducing complexity. While the detection times of Table 1 are on the order of 700 milliseconds, the tracking times of Figures 5 and 6 are between 25 and 50 milliseconds. This corresponds to frame rates between 20 and 40 fps, enabling precise drone control.

Drone control experiments: A final set of experiments was conducted to evaluate the overall performance of the UAV control system. Since this involves autonomous drone flight, we found it difficult to design a reproducible experiment in which all variables could be carefully controlled. Instead, we simply collected videos of autonomous flight, which are available at [2]. Two sets of experiments were performed. In the first, the drone simply had to follow the user, as she moved side to side. Keyframes from these experiments are shown in the top row of Figure 7. The videos in [2] show the successful completion of this task in the dark (night tests), using a tablet to display the badge. In the second experiment, the user successively displayed two badges with different patterns, which implemented two behavior programs, commanding the drone to fly at two different distances. The drone should thus approach or recede from the user, depending on the badge displayed. Keyframes from this experiment are shown in bottom row of Figure 7 (and video in [2]). In both sets of experiments, we observed that the UAV reliably executed the desired behavior and was successful in virtually all autonomous flights.

5. Conclusion

We have introduced a computer vision-based UAV control architecture for robust person-following that also supports behavior programming by manipulation of visual patterns. These goals were achieved by introduction of a programmable badge that depicts a visual pattern with two components. The first is fixed and used to locate the user, the second is variable and displays a behavior programming code. A biologically inspired tracking/recognition architecture, combining bottom-up and top-down saliency, a novel image similarity measure, and an affine validation procedure, was introduced to localize the badge. Badge locations were fed to a control mechanism that adjusts the drone flight so as to maintain a target drone altitude, distance, and angle to the user. These parameters are specified by the behavior programming code. Experiments have shown that the proposed architecture is a reliable person-following solution, enabling a drone to follow a user in real-time, over a range of scenes, distances, and lighting conditions.

Acknowledgements: this research was partially supported by NSF grant IIS-1208522. We also acknowledge the valuable ideas and feedback of Julia Newman.

References

[1] Hexo+ drone autonomously follows the action, for under $500. http://hexoplus.com/.

[2] Person-following UAVs. http://www.svcl.ucsd.edu/projects/dronefollow/.

[3] K. H. Ang, G. Chong, and Y. Li. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.

[4] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó. The SLAM problem: a survey. In *Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, pages 363–371, 2008.

[5] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007.

[6] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013.

[7] Z. Cai, M. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3361–3369, 2015.

[8] R. Cipolla and A. Pentland. *Computer vision for human-machine interaction*. Cambridge University Press, 1998.

[9] R. Cowie. Emotion recognition in human-computer interaction. *Signal Processing Magazine*, pages 32 – 80, 2001.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:886–893, 2005.

[11] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[12] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[15] D. Gao and N. Vasconcelos. Decision-theoretic saliency: computational principles, biological plausibility, and implications for neurophysiology and psychophysics. *Neural computation*, 21(1):239–271, 2009.

[16] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, 15:50, 1988.

[17] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[18] K. Konolige, M. Agrawal, and J. Sola. Large-scale visual odometry for rough terrain. In *Robotics Research*, pages 201–212. Springer, 2011.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[20] W. Li, Q. Yu, H. Sawhney, and N. Vasconcelos. Recognizing activities via bag of words for attribute dynamics. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2587–2594. IEEE, 2013.

[21] D. G. Lowe. Object recognition from local scale-invariant features. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157. Ieee, 1999.

[22] V. Mahadevan and N. Vasconcelos. Biologically inspired object tracking using center-surround saliency mechanisms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(3):541–554, 2013.

[23] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[24] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 593–600. ACM, 2005.

[25] S. Mitra and T. Acharya. Gesture recognition: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007.

[26] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–652. IEEE, 2004.

[27] K. Ogata and Y. Yang. Modern control engineering. 1970.

[28] A. Ohya, A. Kosaka., and A. Kak. Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE Transactions on Robotics and Automation*, pages 969 – 978, 1998.

[29] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[30] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[31] A. M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136, 1980.

[32] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[33] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3169–3176. IEEE, 2011.

[34] J. M. Wolfe. Visual search. *Attention*, 1:13–73, 1998.

[35] Y. Wu and T. S. Huang. Vision-based gesture recognition: a review. In *Gesture-based Communication in Human-Computer Interaction*, pages 103–115. Springer, 1999.

[36] A. L. Yarbus. *Eye movements and vision*. New York: Plenum Pres, 1967.