

Mobile Robot Path Drift Estimation Using Visual Streams

Helio Perroni Filho and Akihisa OHYA

Abstract—Differential Visual Streams (DiVS) is an image processing method to quantify changes in picture sequences. It works on monocular images captured by a single uncalibrated camera. Experiments show DiVS provides a sound basis for an appearance-based navigation system effective under a variety of lighting conditions (both controlled and natural), landmark occlusions and the presence of moving objects.

I. INTRODUCTION

Autonomous navigation is a central topic in mobile robotics, with a number of practical applications [1], [2]. One recurrent use case is *teach-replay*, where a robot is first guided through an environment (the teach step), and must later autonomously retrace the original route (the replay step) [3]. In the absence of measurement errors, this could be easily achieved by recording odometry data. In practice, however, any intrinsic localization method is subject to *drift*, the unrecoverable buildup of reading errors.

Navigation methods resilient to drift usually leverage distinct environment features, otherwise known as *landmarks*, as reference points relative to which more reliable localization is possible. These can be roughly divided into *map-based* and *mapless*, according to whether they depend on globally consistent world representations. Map-based methods can be further divided into *metric*, which represent the environment geometrically relative to a predefined coordinate system, and *topological*, which break down the world as a set of discrete nodes connected by relations of reachability [1]. Landmark detection was initially performed by range sensors such as laser and ultrasound, but recent advances and wider hardware availability have increased the appeal of video cameras for this task [1], [4].

Visual SLAM (Simultaneous Localization and Mapping) is a family of metric map-based navigation methods where selected landmarks are visually tracked (usually through feature extraction and matching) and changes in their apparent position are used to update a probabilistic representation of robot pose. These are computationally expensive operations, making such methods computation-driven [5]. Moreover, the need to keep an accurate map, and precisely track robot pose within it, have always placed an upper bound on the geographic range of SLAM systems [6].

The complexity and limitations of metric map-based navigation systems have motivated work in topological methods.

This work was supported by CNPq grant #201799/2012-0

Helio Perroni Filho is with Intelligent Robot Laboratory, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Japan helio@roboken.esys.tsukuba.ac.jp

Akihisa OHYA is with Intelligent Robot Laboratory, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Japan ohya@cs.tsukuba.ac.jp

TABLE I

SUMMARY OF SEVERAL TEACH-REPLAY VISUAL NAVIGATION METHODS PROPOSED IN THE LITERATURE, CHARACTERIZED IN TERMS OF EMPLOYED SENSORS, TEST ENVIRONMENTS, AND VARIATIONS ALLOWED BETWEEN TEACH AND REPLAY STEPS.

| Method | Sensors | Tested | Landscape Variations | | |
|---|---|----------------------|----------------------|-----------|----------|
| | | | Lighting | Occlusion | Movement |
| Correspondence of vertical lines [7] | Monocular camera | Indoors | No | No | No |
| Template matching [8] | Monocular camera | Indoors | No | No | No |
| Average Landmark Vector [9] | 360° panoramic camera, polarized light sensor, wheel encoders | Simulation, Outdoors | Yes | No | No |
| Block matching Optical Flow [10] | 360° panoramic camera, wheel encoders | Indoors | Yes | Yes | No |
| Feature point tracking [11] | Monocular camera | Indoors, Outdoors | Yes | No | Yes |
| Mutual information Optical Flow vectors [12] | Monocular camera | Indoors | No | Yes | Yes |
| Stereo feature matching and visual motion estimation [13] | Stereo camera | Indoors | Yes | Yes | Yes |
| Local best match and sequence recognition [14] | Monocular camera | Outdoors | Yes | Yes | Yes |

In particular, *appearance-based* navigation methods represent the world as picture sequences collected along a route, which are later matched to live sensory input in order to estimate current robot pose. Image comparison methods that don't require feature extraction and matching can be used, lowering computation costs; representing locations as recorded pictures is also simpler, and closer to how humans navigate our surroundings. On the other hand thorough environment recordings become necessary, making these methods more data-driven.

Table I relates a number of appearance-based navigation methods proposed over the years. They mostly differ in how images are compared – and this is where most fall short to wide applicability one way or another. Common limitations include poor performance in the presence of landmark occlusions, moving objects, or natural (outdoors) lighting. Some methods also require non-standard hardware (e.g. stereo vision rigs). Clearly there is still work to be done on methods that require simple hardware such as a single monocular camera, work well under a variety of lighting conditions, and are resilient to occlusion and object movement.

This article presents an appearance-based navigation system built on *Differential Visual Streams* (DiVS), an image processing method to represent changes in successive landscape images as captured by an approaching observer. Remaining sections are organized as follows. First DiVS is described, then a procedure to quantify changes between

picture sequences, reminiscent of the multichannel neuron abstract machine [15], is constructed over it. Next a navigation control model is designed to generate steering commands in response to such differences. Experiments are reported, demonstrating the method’s resilience under a range of environmental variations. Directions for further research are discussed in the conclusion.

II. DIFFERENTIAL VISUAL STREAM

The *visual stream* function $S(t) = I_t$ returns a snapshot image $I^{m \times n}$ of the field of view at time t . For a source moving towards a visually heterogeneous landscape, the *frame selection* function $p(t, k) = t_k$ defines a sampling strategy for the visual stream, such that any two subsequent sample images are slightly different from each other. The precise design of $p(t, k)$ is discussed later on, but the following properties are assumed to hold:

$$p(t, 0) = t \quad (1)$$

$$p(t, k) < p(t, k + 1) \quad (2)$$

$$\sum_{i,j}^{m,n} |I_a(i, j) - I_b(i, j)| > 0 \quad \left| \begin{array}{l} I_a = S(p(t, k - 1)) \\ I_b = S(p(t, k)) \end{array} \right. \quad (3)$$

Combining $S(t)$ and $p(t, k)$ it’s possible to define a *difference image* function of instantaneous changes to the visual input:

$$D(t, k) = |S(p(t, k - 1)) - S(p(t, k))| \quad (4)$$

Where the subtraction operator is defined for images as pixel-wise subtraction (equivalent to how matrix subtraction is defined as cell-wise subtraction). Each difference image will likely contain regions of high difference (corresponding to the apparent movement of image discontinuities such as edges) and others significantly lower (corresponding to the inside of object surfaces). These regions can be separated into “change” and “no change” classes, by applying a threshold operation such as Otsu’s method [16] to the difference image. The resulting *binary difference image* function is defined as:

$$B(t, k) = \bar{o}(D(t, k)) \quad (5)$$

Where each pixel in the binary difference image $B(t, k)$ is 1 if the corresponding pixel in $D(t, k)$ was above the threshold automatically determined by Otsu’s method, and 0 otherwise.

How much each image discontinuity “moves” between snapshots is determined by several factors. Discounting moving objects, the larger and closer an object is, the bigger the change it will effect on the field of view as it is approached (conversely, the shorter and further away, the smaller the change). Over a prolonged travel, close-by objects – whether small or large – are soon passed by, unless they are sizable obstacles such as corridor walls. Meanwhile, distant objects and their discontinuities remain visible for a long time. Therefore, changes to field of view regions can be further characterized in terms of sustained change over a time

period, which enables limited inference over the landscape’s structure:

- Little to no change: very far away elements such as the sky or a distant horizon line;
- Moderate change: relatively distant, probably large objects such as buildings;
- High change: large, close obstacles such as walls.

These trends can be observed by calculating the pixel-wise average of binary difference images over a sequence window w :

$$DiVS(w, t, k) = \frac{1}{w} \sum_{l=k-w+1}^k B(t, l) \quad (6)$$

Where the *differential visual stream* function $DiVS(w, t, k) = C_k$ returns a *change image* $C^{m \times n}$, a quantitative representation of the changes observed in the field of view over the time window $[t, p(t, w)]$.

Because change images are calculated as the average over a range of binary difference images, they attenuate, or leave out entirely, many of the variations that complicate image processing tasks. Changes in lighting are mostly dealt with; moving objects are seldom registered, especially if they move fast and remain visible for a short period relative to the time range covered by the window; even changes to the landscape outline can have limited effect, if they don’t substantially affect the amount of observed change (as in the case of e.g. closed versus open doors).

III. LANDSCAPE SHIFT

In order to use change images to quantify differences between picture sequences, three problems still need to be addressed:

- 1) How to design frame selection function $p(t, k)$;
- 2) Given a teach step started at time $p(t_t, 0)$ and completed at $p(t_t, n)$, and a replay step started at $p(t_r, 0)$, how to select the change image from range $[DiVS(w, t_t, 0), \dots, DiVS(w, t_t, n)]$ that most closely corresponds to change image $DiVS(w, t_r, k)$;
- 3) Given replay step change image $DiVS(w, t_r, k)$, and its closest teach step correspondent $DiVS(w, t_t, k')$, how to define a difference measure between them that correlates well to robot pose.

Let’s solve the last problem first. Change images computed from sequences collected in the same environment, but different poses, can be compared in terms of *shift* – the apparent differences in landmark position (or rather, of the traces they left) in one change image relative to another. Given two picture sequences A and B , if A was recorded in a path “to the left” of the path that originated B , it is expected that change image C_A will be “right-shifted” relative to C_B – that is, features in B will be consistently found in A , but in positions further to the right. Therefore, shift between change images correlates to differences in robot pose. Furthermore, mobile robots are generally assumed to drive across flat surfaces, and in this context shift can be reduced to a 1D problem – change images can shift “left” or “right”, but

not “up” or “down”. Accordingly, it would be convenient to have a 1D representation of change images that simplified horizontal shift calculation.

Such a 1D representation can be computed from a change image as follows. First the change image is split vertically across the middle, and the lower half is discarded. This is done due to poor signal-to-noise ratio in the lower half of change images, which come mostly from pictures of the environment’s floor. Change half-images are then divided in d columns of equal width; pixels within each column are summed up, producing a *change vector*. These steps are summarized by the *change vector* function:

$$\hat{v}(C^{m \times n}, d) = (v_k = \sum_{i=1}^{\frac{m}{2}} \sum_{j=k \frac{n}{d}}^{(k+1) \frac{n}{d}} C(i, j) \mid 0 \leq k < d) \quad (7)$$

Cross-correlation is a popular technique for determining shift between two signals, robust to noise and quick to compute. A normalized variation of the method [15] is employed to compute shift between change vectors. For two change vectors v_t and v_r , the *windowed correlation function* $WINC(e, \hat{v}_t, \hat{v}_r)$ is defined as:

$$WINC(e, \hat{v}_t, \hat{v}_r) = \sum_{k=0}^{d-e} (\hat{0}^d \parallel \hat{v}_r \star \hat{v}_t[k : k + e]) \ll k \quad (8)$$

Where $\hat{0}^d$ is the zero vector of dimension d , the cross-correlation operator \star is defined as in [15], the concatenation operator \parallel is defined for any two vectors $\hat{a} = (a_1, \dots, a_m)$ and $\hat{b} = (b_1, \dots, b_n)$ such that $\hat{a} \parallel \hat{b} = (a_1, \dots, a_m, b_1, \dots, b_n)$, and the left-shift operator \ll is defined for a vector $\hat{v} = (v_1, \dots, v_k, v_{k+1}, \dots, v_n)$ such that $\hat{v} \ll k = (v_{k+1}, \dots, v_n) \parallel \hat{0}^k$.

In procedural terms, $WINC(e, v_t, v_r)$ places a window of length e over the teach vector, then calculates the normalized cross-correlation of the whole replay vector by the contents of the window. The window is then slid one position to the right, and the operation repeated. At every step a new cross-correlation coefficient vector of length d is computed. Finally, the $d - e$ vectors are zero-padded to dimension $2d$, left-shifted proportionally to the corresponding position of the window over the teach vector, and summed.

A definite output must still be selected from the shift vector. It’s possible to simply select the position of maximum coefficient at every turn, but this might result in shift values changing abruptly over time. A smoother alternative is to select a initial shift, then perform iterative hill climbing over successive shift vectors. Figure 1 illustrates the concept.

Let’s now turn to the problem of selecting a change image from the teach step to match against the current replay change image. First, a proper definition of what means for a change image in the replay step to “correspond” to an image in the teach step is required. Given the robot pose function $o(t) = (x, y, \theta)$, the *frame distance* function $q(t_r, k, t_t, k')$ can be defined as:

$$q(t_r, k, t_t, k') = \|o(p(t_t, k')) - o(p(t_r, k))\| \quad (9)$$

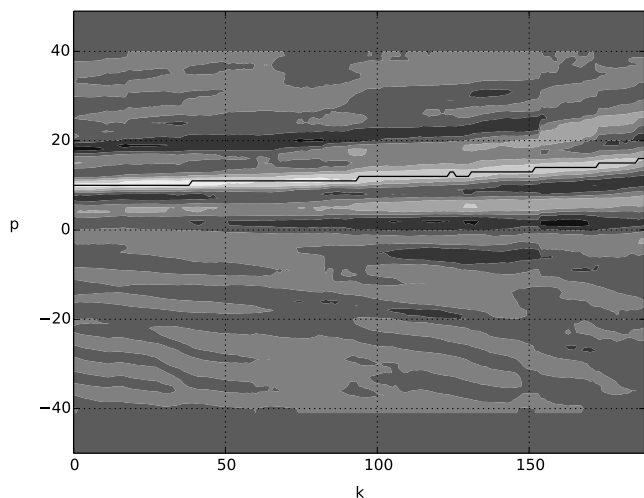


Fig. 1. Local search strategy for shift selection. The contour map shows shift likelihood across a sequence of shift vectors. Horizontal axis is shift vector index (k), while vertical axis is image shift in columns of width d (p). Brighter points indicate higher likelihood. Values at $p_0 = 0$ indicate the likelihood of no shift, values at $p_l > 0$ indicates likelihood of left shift by p_l columns, and $p_r < 0$, of right shift by $|p_r|$ columns. Starting from a global maximum at the first vector, hill climbing is performed to update the selected shift while avoiding too much of a deviation from previous values.

The *frame correspondence function* $c(t_t, t_r, k) = k'$ can then be defined as:

$$c(t_t, t_r, k) = \arg \min_{k' \in [1, n]} q(t_r, k, t_t, k') \quad (10)$$

That is, the replay change image $DiVS(w, t_r, k)$ “corresponds” to the teach change image in the range $[DiVS(w, t_t, 1), \dots, DiVS(w, t_t, n)]$ with the closest associated pose. Obviously, if $o(t)$ were reliably known at any time, there would be no need for visual navigation in the first place. Therefore what is required is an approximation to $q(t_r, k, t_t, k')$ that circumvents this requirement.

Several different strategies can be considered. The trivial solution is to correspond change images by index:

$$q_i(t_r, k, t_t, k') = |k - k'| \quad (11)$$

This may work reasonably well if the robot is driving at the same, constant speed in both teach and replay steps, and images are sampled at regular intervals (which might depend on implementation factors, such as hardware resources, or how the underlying operating system schedules processes and handles interruptions).

A slightly more elaborate option is to correspond change images by acquisition time, relative to step start time:

$$q_t(t_r, k, t_t, k') = |(p(t_t, k') - t_t) - (p(t_r, k) - t_r)| \quad (12)$$

This may work better than the previous option when regular sampling is not guaranteed, but it still assumes a constant speed shared across both steps.

It’s also possible to correspond change images by the distance $d(t_0, t) = s$ traveled since the start of the step:

$$q_d(t_r, k, t_t, k') = |d(t_t, p(t_t, k')) - d(t_r, p(t_r, k))| \quad (13)$$

It may seem contradictory to employ traveled distance on a system ostensibly meant to compensate for inaccuracies in odometry data, but it could work if the origin location is regularly reset – for example, every time the robot passes key landmarks.

In the implementation and tests detailed later on the distance-based matching strategy is used, but in fact after several tests performed on a range of $10m$ to $20m$, no significant performance difference was found among the three.

Finally, the frame selection function must be defined. The problem is how to ensure that any two subsequent images ($S(p(t, k)), S(p(t, k + 1))$) are different enough to provide useful information, but not too different, as this would increase noise. A simple solution is to again rely on traveled distance, defining $p(t, k)$ as:

$$p(t, k) = \arg \min_{t'} q(k - 1) \leq d(t') - d(t) \quad (14)$$

Where q is the minimal distance traveled by the robot between $p(t, k)$ and $p(t, k + 1)$.

IV. CONTROL MODEL

Given a non-zero shift between the current replay step change image and its teach step correspondent, steering towards the direction opposite to the shift should reduce and eventually eliminate it, as the robot returns to the teach path. Empirical tests indicate image shifts (which are measured in pixels) are numerically proportional enough to drifts in heading direction (measured in degrees) and sideways distance from the teach path (measured in meters), that the later can be computed from the former merely by application of a weighting term and cut-off limits. In more precise terms, given a shift s measured in pixels from the center, such that positive values represent shifts to the left and negative values, to the right, the heading direction drift $\Delta\theta$ and sideways drift Δy are defined as:

$$\Delta\theta = \begin{cases} -10^\circ & \text{if } s \leq -10 \\ 10^\circ & \text{if } s \geq 10 \\ s^\circ & \text{otherwise} \end{cases} \quad (15)$$

$$\Delta y = \begin{cases} -0.5m & \text{if } s \leq -50 \\ 0.5m & \text{if } s \geq 50 \\ \frac{s}{100}m & \text{otherwise} \end{cases} \quad (16)$$

Unfortunately, the task of actually steering the robot in response to estimated drifts is complicated by the effects

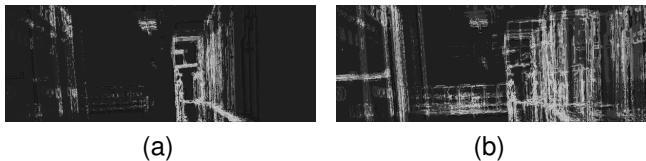


Fig. 2. Current change image cluttering caused by computation of difference images over a curved path. As the robot turns in response to a computed shift, the initially sparse replay change image (a) becomes cluttered with changing regions that fell out of alignment with the current sequence (b).

of curved paths on change images. As Figure 2 shows, the moment the robot starts turning in response to a shift estimate, the changing regions of the visual field fall out of alignment with the ongoing sequence, cluttering the change image. To account for this, every time the robot starts moving in a curved path – which it will do in order to compensate for a non-zero drift estimate – it immediately stops collecting images. After movement has stabilized in a new straight path, a new “time zero” for change image calculation is set, so any images collected before that moment are ignored. Obviously this implies the robot will “run blind” for a time, unable to further estimate shifts until heading direction stabilizes.

V. EXPERIMENTS

A differential drive robot with a stock web camera mounted to its front was employed in a series of experiments to evaluate the model described above. The model was implemented in the form of C++ library **Cight**. Built on top of OpenCV, its source code is available on the web under an Open Source license [17]. Runtime profiling shows change image computation under **Cight** is remarkably fast: after the first w difference images have been calculated, it takes a little over 40,000 CPU cycles (about 0.04 seconds in an $2.67GHz$ Intel Core i5 processor) to compute an additional difference image and recalculate the change image.

System parameter values ($w = 25, d = 50, e = 5$) were empirically determined and used throughout all experiments. Linear speed was always set to $0.3m/s$, with pictures and odometry recorded every $1.5cm$ on average. Two test environments were selected in the campus of the University of Tsukuba, henceforth referred to as “indoors” and “outdoors”: the corridor in front of laboratory room 3D402 (the “indoors” environment) and the paved way by the front entrance of faculty building 3L (the “outdoors” environment). The indoors environment featured a smooth floor; the outdoors environment’s floor was covered in tiles that added a noticeable amount of vibration to robot movement, but was otherwise flat.

In order to access shift measurement quality, shifts between teach step records were calculated off-line. Four different teach steps were recorded in each environment, all from the same start pose: one “reference” record, and three other “test” records. Test records occasionally include occlusions and the presence of moving elements such as people. Additionally, in two of the test records, the robot performed a 5° in-place turn (one to the left, another to the right) before setting off. Despite differences such as radical changes in lighting conditions, large bright saturation areas due to limitations in camera hardware, and the presence of pedestrians that further occlude landmarks seen in the reference teach step record, shift results were remarkably consistent with the actual relative orientation of the teach steps.

Figure 3 shows results for these tests for the outdoors environment. Given a reference teach step record collected in the morning (illustrated in Fig. 3a), and three test records collected in the afternoon, starting from exactly the same



(a)

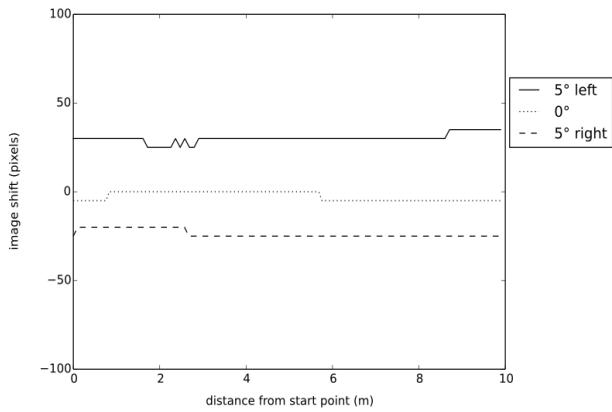
(b)



(c)



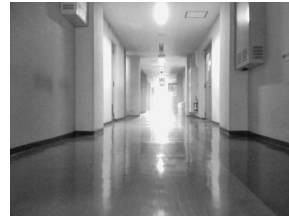
(d)



(e)

Fig. 3. Off-line shift estimate tests on outdoors environment. Given a reference teach step record collected in the morning (a), and three test records collected in the afternoon, starting from exactly the same pose as the reference record (b), or from the same pose plus a 5° in-place turn to the left (c) or right (d), plots of shift estimates over distances from the start (e) show that DiVS correctly estimates zero or negligible shift for the 0° case (dotted line), where the robot is treading much the same route as in the reference case; positive shifts for the 5° left case (solid line), where the robot is moving away from the original path in a leftward direction; and negative shifts for the 5° right case (dashed line).

pose as the reference record (Fig. 3b), or from the same pose plus a 5° in-place turn to the left (Fig. 3c) or right (Fig. 3d), plots of shift estimates over distances from the start (Fig. 3e) show that DiVS correctly estimates zero or negligible shift for the 0° case (where the robot is treading much the same route as in the reference case), positive shifts for the 5° left case (where the robot is moving away from the original path in a leftward direction) and negative shifts



(a)

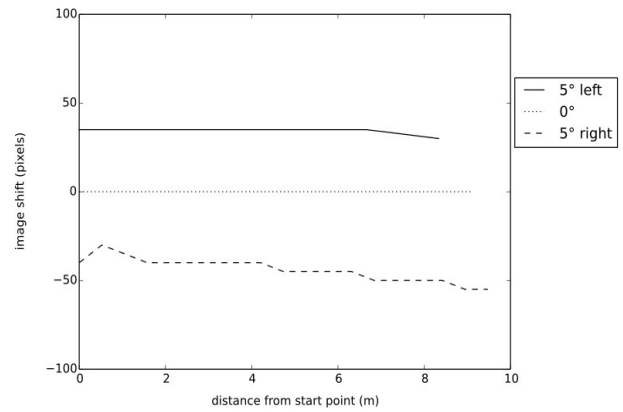
(b)



(c)



(d)



(e)

Fig. 4. Off-line shift estimate tests on indoors environment. Given a reference teach step record collected in the morning (a), and three test records collected in the afternoon, starting from exactly the same pose as the reference record (b), or from the same pose plus a 5° in-place turn to the left (c) or right (d), plots of shift estimates over distances from the start (e) show that DiVS correctly estimates zero shift for the 0° case (dotted line), where the robot is treading much the same route as in the reference case; positive shifts for the 5° left case (solid line), where the robot is moving away from the original path in a leftward direction; and negative shifts for the 5° right case (dashed line).

for the 5° right case. Likewise, Figure 4 shows test results for the indoors environment: in this case differences between the reference case (illustrated in Fig. 4a), and the three test records (Fig. 4b-d) were more subtle, but as Fig. 4e shows, overall results were not significantly different.

After the performance of the shift estimator was properly evaluated off-line, replay steps were also performed. As the plot in Figure 5 shows, after a teach step where the robot

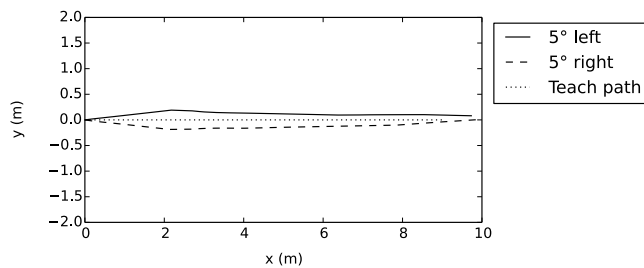


Fig. 5. Results for replay tests performed before the front entrance of building 3L. The plot shows a teach step path (the dotted line) where the robot ran in a straight line from pose $(x_0, y_0, \theta_0) = (0m, 0m, 0^\circ)$ to pose $(x_n, y_n, \theta_n) = (10m, 0m, 0^\circ)$, and two replay paths, with initial 5° left (solid line) and right (dashed line) in-place turns to simulate drift. Horizontal axis corresponds to the x component of robot pose, and vertical axis, to the y component. Heading direction component θ is not shown but can be inferred by the form of the path lines.

ran in a straight line from pose $(x_0, y_0, \theta_0) = (0m, 0m, 0^\circ)$ to pose $(x_n, y_n, \theta_n) = (10m, 0m, 0^\circ)$ (the dotted line), two replay steps were performed, one preceded by an initial 5° in-place turn to the left (solid line), and another by an initial 5° in-place turn to the right (dashed line). As can be seen, in both replay steps the robot is able to detect and counteract the differences in heading direction relative to the teach path.

VI. CONCLUSION

This article described the *Differential Visual Stream* (DiVS) image processing method, which computes a graphical representation of changes observed in subsequent landscape images captured by an approaching observer. Such change images, when computed from sequences collected at similar positions in the same environment, can be compared in terms of horizontal shift. This in turn can be used to implement appearance-based navigation. Because change images attenuate or outright erase many of the transitory differences between images of a same landscape, robust navigation can be achieved against changing environmental conditions.

In order to evaluate the effectiveness of DiVS as the basis for an appearance-based navigation system, a procedure to compute shift between change images was introduced, as well as a control model that uses shift information to steer a mobile robot. Experiments demonstrated the method's resilience to a range of environmental variations, such as lighting changes, occlusion and the presence of moving elements, both indoors and outdoors. In its current form DiVS can only handle image sequences over straight paths, making it a better match for artificial environments where terrain is flat and turns are relatively infrequent (such as city landscapes). Further research is required to remove this limitation.

Tests under a wider variety of environments and visual conditions are still required to gauge the method's operational limits, particularly in relation to the presence of moving objects. The question of how to set parameters used across all three system stages (change image synthesis, shift computation, steering control) should be concurrently

addressed: at present these are set empirically, but with the right set of quality metrics they could be instead learned from the environment. For example, the parameter s used to convert image shift measurements to drift estimates could be initialized to an arbitrary value, and then iteratively adjusted as steering corrections either undershoot or overshoot current image shift.

Finally, change image correspondence by traveled distance requires a kind of sensor data (odometry) that is neither always available, nor reliable over long ranges. Correspondence by analysis of the change images themselves, or some other visual method, would be a more portable alternative, enabling the method to work in the absence of any sensor data other than an image stream.

REFERENCES

- [1] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296, nov 2008.
- [2] R. Arkin and R. Murphy, "Autonomous navigation in a manufacturing environment," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 4, pp. 445–454, 1990.
- [3] D. Burschka and G. Hager, "Vision-based control of mobile robots," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, 2001, pp. 1707–1713.
- [4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [5] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The slam problem: A survey," in *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*. IOS Press, 2008, pp. 363–371.
- [6] J. A. Castellanos, J. Neira, and J. D. Tardós, "Limits to the consistency of ekf-based slam," in *5th IFAC Symp. on Intelligent Autonomous Vehicles, IAV'04*, 2004.
- [7] T. Ohno, A. Ohya, and S. Yuta, "Autonomous navigation for mobile robots referring pre-recorded image sequence," in *IROS. IEEE*, 1996, pp. 672–679.
- [8] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," 1996, pp. 83–88.
- [9] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner, "A mobile robot employing insect strategies for navigation," *Robotics and Autonomous Systems*, vol. 30, no. 1–2, pp. 39 – 64, 2000.
- [10] A. Vardy and R. Möller, "Biologically plausible visual homing methods based on optical flow techniques," *Connection Science, Special Issue: Navigation*, vol. 17, pp. 47–90, 2005.
- [11] Z. Chen and S. T. Birchfield, "Qualitative vision-based mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.
- [12] R. Stewart, M. Mills, and H. Zhang, "Visual homing for a mobile robot using direction votes from flow vectors," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, Sept 2012, pp. 413–418.
- [13] J. Kim, Y. Bok, and I. Kweon, "Robust vision-based autonomous navigation against environment changes," in *IROS*, 2008, pp. 696–701.
- [14] M. Milford and G. Wyeth, "Seqslam : visual route-based navigation for sunny summer days and stormy winter nights," in *IEEE International Conference on Robotics and Automation (ICRA 2012)*. IEEE, 2012, pp. 1643–1649.
- [15] H. Perroni Filho and A. F. De Souza, "On multichannel neurons, with an application to template search," *Journal of Network and Innovative Computing*, vol. 2, no. 1, pp. 10–21, 2014.
- [16] N. Otsu, "A threshold selection method from gray-level histograms," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 9, no. 1, pp. 62–66, Jan 1979.
- [17] H. Perroni Filho, "Cight," 2014. [Online]. Available: <https://github.com/xperroni/Cight>