Adaptive Data Confidence Using Cyclical Gaits On A Modular Snake Robot

Benjamin H. Morse and Howie Choset

***Abstract*— We are interested in gaining situational awareness and autonomy from modular snake robots. Modular snake robots exhibit undulatory motions which can cause odometry algorithms to be irrelevant due to direction of the snake's camera, or give bad data due to vibration or velocity of the camera. This paper presents a technique for leveraging internal shape data and cyclical gait information for data validation of sensors rigidly attached to the snake robot. This validation is described using a function that relates the reliability of sensor data to the internal shape of the robot within a gait cycle. We use a recently developed technique for choosing a "good" frame on such a robot to create a substitution for a motion model. We demonstrate the advantages of our technique using a Kinect^{TM}performing an open-source implementation of 6-DOF SLAM. The output of the SLAM algorithm with the data validation is compared qualitatively to the output of the SLAM algorithm without the data validation. Other potential applications of the technique are mentioned.**

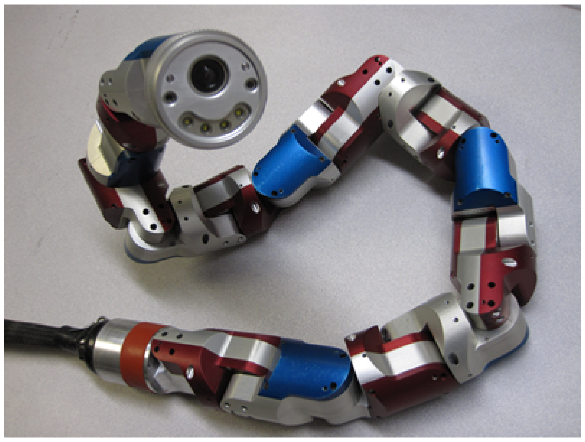


Fig. 1: 16-DOF snake robot.

I. Introduction

Modular snake robots equipped with sensors have shown potential for locomoting within difficult-to-access environments. These sensors allow the robots to send back data to accomplish various tasks, and also to increase the situational awareness of the robots and their operators. This situational awareness is paramount to the operation of the robots, as they move erratically and their movement depends largely on terrain. Periods of erratic motion happen predictably according to the gait cycle, but can affect the reliability of sensor data due to poor alignment of sensors with the world. Fast changes in the orientation of the sensor makes using the data for certain algorithms very unreliable. On wheeled robots, these problems are minimized due to a smooth motions and an obvious body frame of reference providing a simple static location for sensors.

We propose a technique that uses a function to map a particular shape of the robot within a gait to the validity of the data from a given sensor. We talk about how these functions apply to filtering techniques and we give example functions as they apply to gaits on a 16-DOF modular snake robot, showing how the functions might be generated. These functions also are informative for new methods of gait controls. We suggest applications that generalize beyond modular snake robots.

We use the Microsoft Kinect^{TM}sensor with an open-source implementation of 6-DOF Simultaneous Localization and Mapping (graphSLAM) to compare SLAM with no data validation to SLAM with data validation informed by the snake robot's control software. Several gaits are run and the pose graphs and maps produced by the graphSLAM algorithm are compared and explained qualitatively.

Benjamin H. Morse and Howie Choset are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213. Email: {bmorse@andrew, choset@cs}.cmu.edu.

II. Prior Work

This paper builds on extensive prior work on filtering algorithms and other methods of sensor fusion. The function based technique presented in this paper is based on applications using Information and Kalman Filters [1]. The Extended Kalman Filter [2] extends the Kalman Filter to nonlinear domains and is very widely used to model predictions and uncertainty in the face of imperfect sensing and motion models.

The robot featured in this paper has many gaits that are based on a common mathematical framework [3] [4]. Several of these gaits are combined to form a set of motion primitives used to move the robot in a controllable manner. These gaits lack motion models, because the motion realized from the internal shape change varies with the terrain that the gaits are executed on, and the dynamics are complex.

In order to generate the functions used with the technique in this paper, a new method of finding intuitive body frames is used. The method will generate a body frame that separates the internal and external motion of the robot. This concept called virtual chassis [5] allows a shape stable method of determining the relative orientation of the sensor in question to the principal axis of the snake (which tends to be closely related to the direction of motion of each gait). This information helps to form a pseudo motion model, which informs the technique on what data to trust.

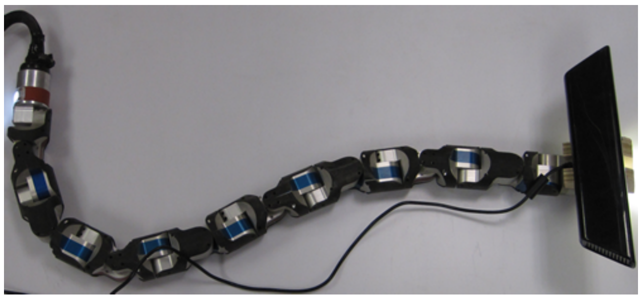


Fig. 2: Hardware setup for Kinect^{TM}sensor mounted on a snake robot

The benchmark that this paper uses to evaluate the effects of the technique is GraphSLAM from Thrun and Montemerlo [6]. It uses a sparse graph to represent all of the linear constraints on the robot poses and map feature positions, where each edge is a constraint, and each node is a pose or map feature. This allows for efficient optimization of past data, so that future readings can update the prediction on current and past pose and map estimates. This also scales relatively well with the number of features in the world, as long as the graph remains sparse (every feature should only be seen from a few pose estimate positions).

III. Approach

A. Normalized Phase

The first tool that is used to identify gait position in relation to sensor position is normalized phase. Normalized phase, denoted ϕ is a number from 0 to 1 that represents how far through a particular gait the robot is. The equation below shows the general equation for snake robot gaits.

$$\alpha(n,t) = \begin{cases} \beta_v + A_v \sin(\theta), \text{for lateral modules,} \\ \beta_h + A_h \sin(\theta + \delta), \text{for dorsal modules,} \end{cases}$$
$$\theta = \left(\frac{d\theta}{dn}n + \frac{d\theta}{dt}t\right) \tag{1}$$

where β, A, θ, and δ are respectively offset, amplitude, frequency, and lateral-dorsal phase shift. The spatial component $\frac{d\theta}{dn}$ describes the frequency of the shape of the robot and the temporal component $\frac{d\theta}{dt}$ determines the frequency of the actuator cycles. n and t are module number and time respectively. A zero value for $\frac{d\theta}{dn}$ causes identical angles on all of the modules on one axis, generating an arc (or possibly a straight line) along that axis. A zero value for $\frac{d\theta}{dt}$ has the effect of holding the robot in a static shape. δ is simply a phase shift to control the timing between the motions of the two orthogonal waves, producing a variety of gaits. The β_h term is generally used to steer the robot when locomoting on the ground.

In this case, the normalized phase of a gait using this model can be computed from $\frac{\theta}{2\pi}$.

B. Trust Function

The technique in this paper involves a function that the user generates called the trust function. This function is a mapping of normalized phase to the apparent reliability of the sensor. We denote the trust function $G(\phi)$

$$G : [0,1] \rightarrow [0,1] \tag{2}$$

This trust function's numbers can be simply transformed to have canonical meaning. A value of zero means the controller should ignore where a value of one means the controller should treat the data from the sensor with the standard sensor model for that sensor. Numbers in between represent a factor of uncertainty associated with that sensor.

A simple example of the application of the trust function involves using notation from Information Filters [7]. When dealing with the sensor model for each sensor used in an Information Filter, a information value is used to determine information gain, which is the amount of certainty gained from a sensor reading. This is usually a static value that is tuned to fit the sensor under all circumstances. In our technique, the information value for the sensor at a given normalized phase could be expressed as the information gain specified by the sensor model multiplied by the trust value at that normalized phase.

$$I(\phi) = I_{Static}G(\phi) \tag{3}$$

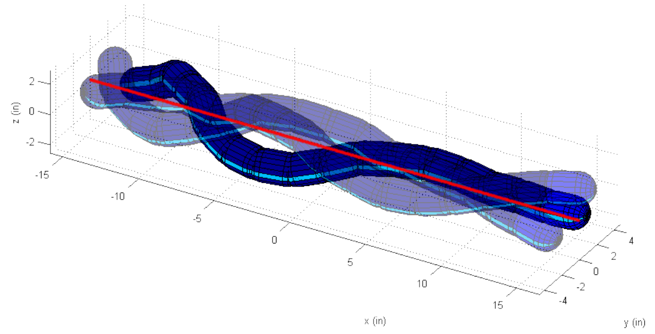
Thus when the trust is zero, the information is zero, which should be interpreted as ignoring the sensor data for that iteration. This translates in a less elegant fashion into the measurement covariance matrix R for the Kalman Filter, where a zero trust value would yield an infinite covariance.

$$R(\phi) = \frac{R_{Static}}{G(\phi)} \tag{4}$$

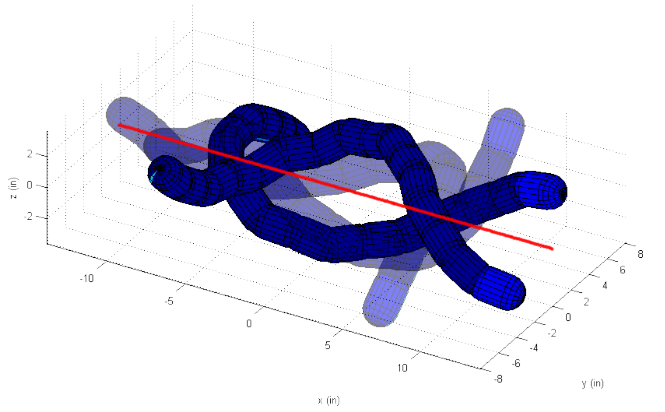
C. Function Generation

For this paper, visual odometry is the primary thrust considered in function generation. Here two factors can contribute to a sensor being unreliable: deviation of the sensor from the direction of motion and velocity (both angular and linear) of the sensor. These factors are necessary to consider as visual odometry requires large number of features to be shared between subsequent frames in order to get a reliable camera transformation estimate, and large velocities on the camera will cause motion blur which will completely invalidate frames. These factors might be used to generate smooth functions or step functions to represent trust, depending on the application and the desire amount of data to be processed. If throwing out data does not seem like a good option, a smooth function that distrusts but does not ignore data would be an appropriate solution, whereas an application where data can be discarded might allow for a step function approach. We used step functions for our trust functions so as to save memory and runtime.

For this paper, we assumed the snake robot would be moving through the trusted region of the gait at low enough



(a) Time Lapse Poses of Slither Gait



(b) Time Lapse Poses of Sidewind Gait

Fig. 3: Example of Slither and Sidewind gaits virtual chassis results. Here poses that align the head with the long axis of the virtual chassis are bold, and the poses with the most extreme angular deviation of the head from the long axis of the virtual chassis are faded to show the erratic motions within the gait. The red lines represent the long axis of the virtual chassis frame which is the first principal moment of inertia of the snake.

speeds so as to not cause motion blur, and therefore the only factor used in function generation was the alignment with the direction of motion. Examples of the explicit functions are generated and explained below.

IV. Robot Implementation

The modular snake robots utilize many gaits whose direction of motion roughly correspond to one of the principal moments of inertia of the system of all the snake modules. The principle moment of inertia of the snake tends to be stable to the shape of the robot. Therefore having the head align with the first principal moment of inertia of the snake will either cause the head to be looking in the direction of motion, or perpendicular to it. Both of these motions are easily accounted for by a visual odometry algorithm.

To calculate the deviation from the first principal moment of inertia, we use the joint angles generated by the gait function at each phase to generate transforms between each module, then find the head module frame and transform that resulting frame into the virtual chassis frame of the snake robot. Then the resulting 3-dimensional homogeneous transform matrix is in the form:

$$H_{head} = \begin{bmatrix} r_{xx'} & r_{xy'} & r_{xz'} & t_x \\ r_{yx'} & r_{yy'} & r_{yz'} & t_y \\ r_{zx'} & r_{zy'} & r_{zz'} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The camera points along the z-axis of the head module, so we extract the direction vector of the head module.

$$dir_{head} = \begin{bmatrix} r_{xz'} \\ r_{yz'} \\ r_{zz'} \end{bmatrix} \tag{6}$$

To calculate the angle of deviation we use the direction vector for the first principal moment of inertia of the system, dir_{vc}.

$$\theta_{head} = acos(dir_{head} \cdot dir_{vc}) \tag{7}$$

Based on the way the virtual chassis is calculated, in this paper the first principal moment of inertia direction is always along the +x axis.

$$dir_{vc} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

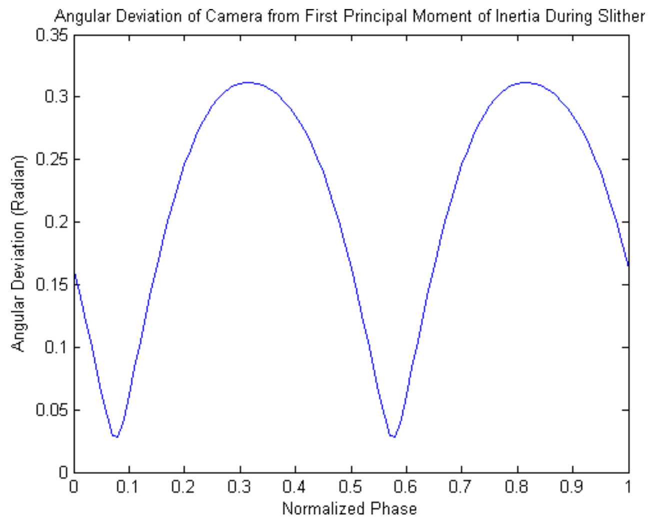
A. Slither

The modular snake robot uses the slither gait as its primary means of forward locomotion. This gait causes lots of erratic motion in the back half of the snake, while keeping the front of the snake more stable, however the head of the snake varies depending on the amplitude. Using an lateral and dorsal amplitude of $\frac{\pi}{5}$ and a phase per module of $\frac{1}{8}$ for x and $\frac{1}{14}$ for y, the head deviates as much as $18°$ in each direction from the first principal moment of inertia provided by the virtual chassis. Fig. 3a shows the centered and extreme poses in one cycle of the slither gait. Here the bold snake pose has the head centered, while the faded poses show the head at its extreme points.

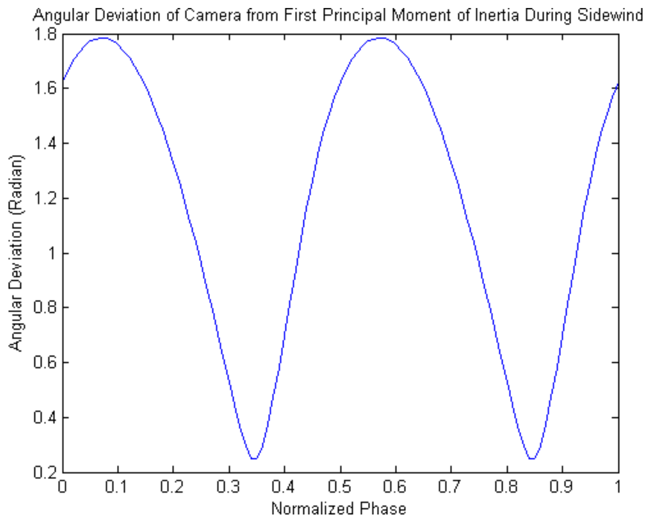
Fig. 4a shows the angular deviation of the head module with respect to the normalized phase. This leads to a corresponding trust function that is 1 from normalized phase 0.07 to 0.08 and 0.57 to 0.58 and is 0 everywhere else. It is also evident from this graph that the angular velocity of the head is highest at these times, so if this gait is being cycled at high speed, a small pause should be added at these valid regions.

B. Sidewind

The modular snake robot uses the sidewind gait as a very efficient way of moving sideways and also up inclines. This gait moves erratically along the entire snake. Using an



(a) Angular Deviation of Slither Gait



(b) Angular Deviation of Sidewind Gait

Fig. 4: This graph shows the angular deviation of the head module from the first principal moment of inertia provided by the virtual chassis, plotted against the normalized phase for a) the slither gait and b) the sidewind gait. The trusted regions here are located around the local minima.

amplitude of 0.6 and a phase per module of 0.0465, the head deviates from the first principal moment of inertia by just over 103° in each direction. Fig. 3b shows the centered and extreme poses in one cycle of the slither gait. Here the bold snake poses have the head aligned with the first principal moment of inertia, while the faded poses show the head at its extreme points.

Fig. 4b shows the angular deviation of the head module with respect to the normalized phase. This leads to a corresponding trust function that is 1 from normalized phase 0.34 to 0.35 and 0.84 to 0.85 and is 0 everywhere else. It is also evident from this graph that the angular velocity of the head is highest at these times, so if this gait is being cycled at high speed, a small pause or slowdown should be added at these valid regions.

V. Results

A. Setup

Our method uses open-source code provided through the Robot Operating System (ROS) infrastructure. The code package is entitled RGBDSLAM and is provided by Felix Endres et al. from the University of Freiburg [1]. We interface the sensor to this package using open source drivers for the Kinect that are now integrated into a ROS node. The underlying algorithm is GraphSLAM, creating a posterior for full 6 DOF path and 3D colorized point cloud map of the environment.

The experiments for this paper were run on a 6-core 3.5 GHz desktop computer with 8 GB of RAM. We run the *diamondback* release of ROS. The RGBDSLAM code is from https://svn.openslam.org/data/svn/rgbdslam/trunk repository at revision 11. Fig. 2 shows the hardware configuration of the snake robot. Here the Kinect^{TM}is attached to the head of the snake robot, directly above the analog camera. It is important to note that the Kinect sensor is mounted to be at a 30° angle above the z-axis of the head module.

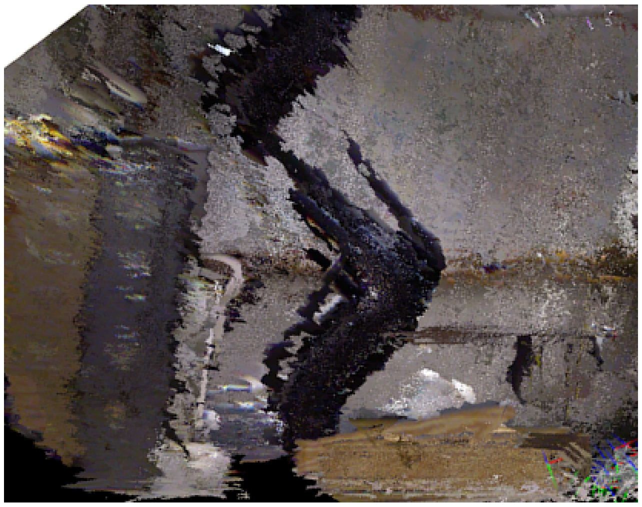
The SLAM software uses the grayscale images from the Kinect's camera for visual odometry. The software supports many different feature detectors through OpenCV (Intel Corporation, Santa Clara, CA), but for this paper the Speeded Up Robust Features (SURF) detector was used [8]. Feature matching and Random Sampling Consensus (RANSAC) [9] are used to estimate the transformation between frames. Feature matching is enhanced using the Fast Library for Approximate Nearest Neighbors (FLANN) [10]. There are several conditions that must be met for a node to be added to the graph. First, a frame must have sufficient features so that it can be matched robustly. To prevent unrelated frames from sharing an edge in the graph, the code compares the number of matches between nodes to a hit threshold.

We ran two experiments in the lab using objects along one wall to provide features for the visual odometry algorithm to use. Two different gaits were tested using the functions and technique designed above. These experiments were run in order to compare the map and pose graph outputs of the GraphSLAM algorithm between the filtered sensor data and the unfiltered data.

B. Slither

The slither gait was tested over multiple cycles moving mostly forward several feet. During the test the gait was cycled at low speed, to allow the unfiltered data to produce a map. Fig. 5a shows a map generated from unfiltered sensor data. Here the map has many errors and does not correctly rectify the position and size of the large black pvc pipe apparatus present in the center of the image. Fig. 5b shows

[1]http://www.ros.org/doc/api/rgbdslam/html/index.html


(a) Map Generated for Slither Gait without Filtering

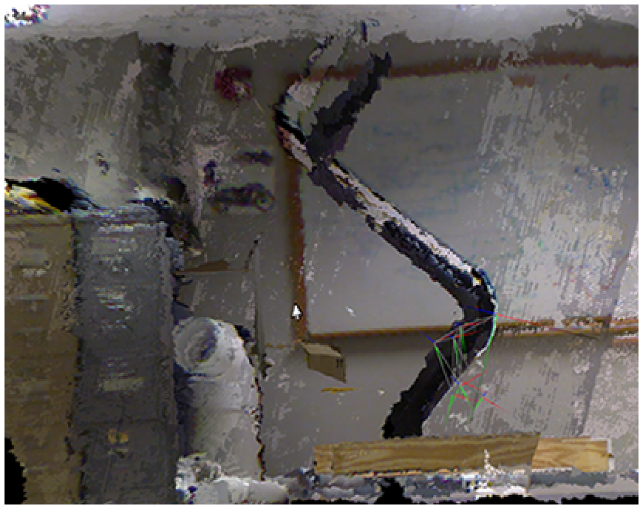
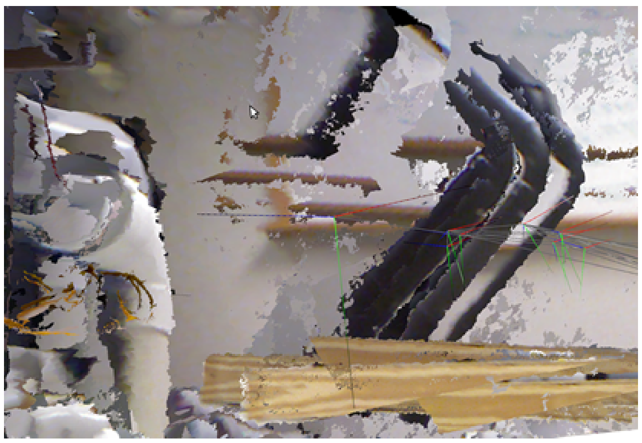

(b) Map Generated for Slither Gait with Filtering

Fig. 5: These maps were generated by the open-source SLAM software, both of the same area using the slither gait. These were generated a) without trust function filtering and b) with trust function filtering.


(a) Map Generated for Sidewind Gait without Filtering

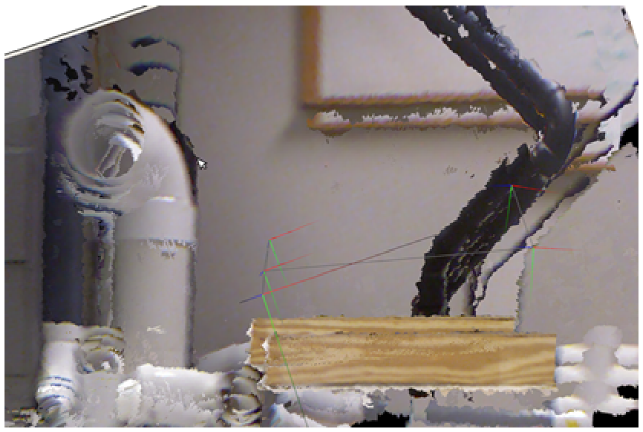

(b) Map Generated for Sidewind Gait with Filtering

Fig. 6: These maps were generated by the open-source SLAM software, both of the same area using the sidewind gait. These were generated a) without trust function filtering and b) with trust function filtering.

the map generated from a similar run using the trust-function approach. This map has minimal conflicts and rectifies that same pipe apparatus well.

Fig. 7a shows the pose graphs that correspond to the two maps, from an overhead view. The top of the figure shows the pose graph from the unfiltered data. This graph has many outliers and is cluttered, with many observations contributing to the inaccuracy in the map. The path generated from the filtered data clearly shows the gait cycles, and maintains a fairly straight path with no outliers.

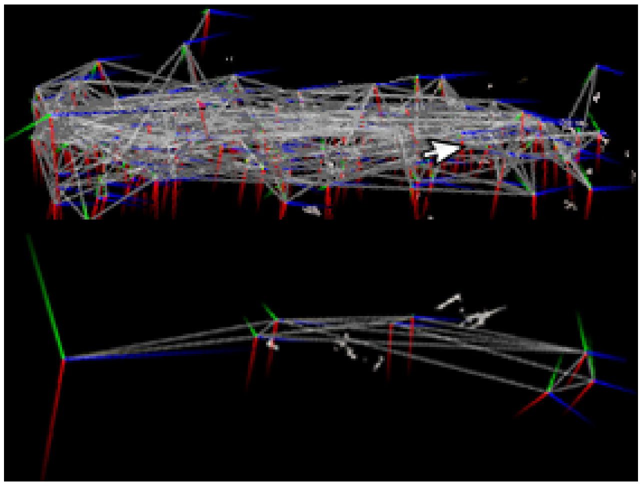
C. Sidewind

The sidewind gait was tested over multiple cycles moving sideways. Fig. 6a shows a map generated from unfiltered sensor data. Similarly to the slither experiment, the algorithm has trouble rectifying the position of the objects such as the pipes. Fig. 5b shows the map generated from a similar run using the trust-function approach. This map also has several conflicts due to motion artifacts from the sidewinding gait, but the artifacts are clearly reduced from the filtering.

Fig. 7b shows the pose graphs that correspond to the two maps, from an frontal view. The top of the figure shows the pose graph from the unfiltered data. The difference between the pose graphs for sidewind is much less pronounced than for slither, however the lower number of nodes makes the path simpler and helps the map stay more accurate.

VI. Extensions

The technique that has been presented in this paper applies beyond snake robots. Another example of a robot that might benefit from dynamic data confidence is the dynamic single actuator vertical climbing robot [11]. This robot uses its actuator and the friction from its environment to climb


(a) Pose Graphs for Slither Gait

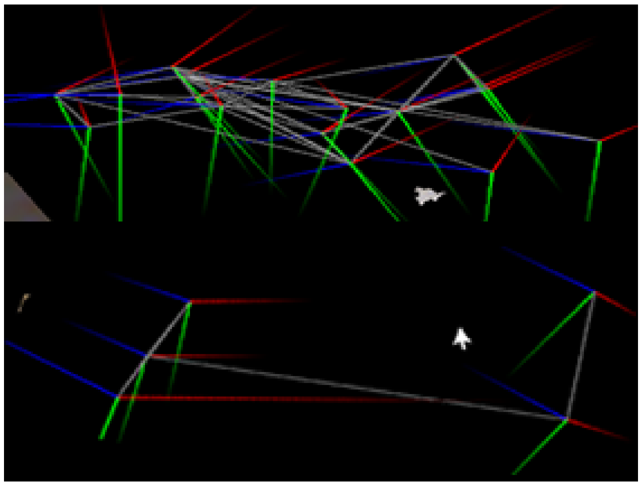

(b) Pose Graphs for Sidewind Gait

Fig. 7: These are the pose graphs generated by the open source SLAM algorithm, where each pose consists of a red, green and blue axis connected by white lines representing the graph edges. The unfiltered path is above and the filtered path is below. The gaits used for these paths are a) slither and b)sidewind.

vertically in a simplistic manner. This robot cycles between a power phase and a flight phase. This action can be modeled as a gait.

A sensor placed on this robot to determine distance from the ground could be filtered to provide useful information to the robot, but would become very unreliable during the powered phase of the robot's motion, as this phase consists of high impact with the environment, causing significant vibration in the robot's body. Therefore, it could be beneficial to create a function that exhibits high trust of the sensor during the flight phase, specifically when the sensor is pointing down at the apex of the flight arc, and low confidence close to the impact point of the cycle.

This function model also provides useful information in gait design. For the modular snake robots, cycling the gait at high speeds can help the robot locomote, but can cause the camera on the snake to have too much motion blur to be useful for odometry or intuitive remote operation. Given a reasonable motion model, or partial motion model using the virtual chassis, functions generated above will show when in the gait information will be most useful, and therefore when a gait designer might temporarily slow down the gait to gather sensor data. This way the majority of the gait can cycle at high speed, and the sensors will still be able to pick up data to perform visual odometry and/or SLAM.

VII. Conclusion

With both of the experiments run, the overall quality of the maps improved, and the pose graphs were much simpler with the filtering. In SLAM, the quality of the map and the quality of the path are directly related, since updates on the map are made based off estimations of the path and visa versa. Thus having static errors in the map will result in a less accurate pose graph.

Having a more sparse pose graph also allows the memory and processing power to remain manageable for longer runtimes. The filter also seems to make the pose graphs much more human readable, which is important especially for the modular snake robots which are teleoperated. The additional situational awareness gained from the filtering can greatly improve the ability of the operator to pilot the snake robots.

These results are expected to extend beyond the scope of SLAM and into other filtering applications.

Acknowledgment

The authors would like to thank the Biorobotics Lab at the Robotics Institute. In particular, special thanks go out to David Rollinson, Matthew Tesch, Austin Buchan, and Bradford Neuman.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problem," *Journal of Basic Engineering Transactions*, vol. 82, no. 1, pp. 34–45, 1960. [Online]. Available: http://ci.nii.ac.jp/naid/10006538775/en/

[2] R. Ohap and A. Stubberud, "A technique for estimating the state of a nonlinear system," *IEEE Transactions on Automatic Control*, vol. 10, no. 2, pp. 150 – 155, Apr 1965.

[3] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," *Advanced Robotics*, vol. 23, pp. 1131–1158(28), June 2009.

[4] K. Lipkin, I. Brown, A. Peck, H. Choset, J. Rembisz, P. Gianfortoni, and A. Naaktgeboren, "Differentiable and Piecewise Differentiable Gaits for Snake Robots," in *Proceedings of IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, San Diego, CA, USA, Oct 29 - Nov 2 2007, pp. 1864–1869.

[5] D. Rollinson and H. Choset, "Virtual chassis for snake robots," in *IEEE International Conference on Intelligent Robots and Systems (accepted)*, 2011.

[6] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[7] A. G. O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1998.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF),” *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[9] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[10] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *International Conference on Computer Vision Theory and Application VISSAPP’09)*. INSTICC Press, 2009, pp. 331–340.

[11] A. Degani, A. Shapiro, H. Choset, and M. Mason, “A dynamic single actuator vertical climbing robot,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-nov. 2 2007, pp. 2901 –2906.