Robotics and Biomimetics
a SpringerOpen Journal

RESEARCH Open Access

Terrain perception for a reconfigurable biomimetic robot using monocular vision

Arnab Sinha*, Ning Tan and Rajesh Elara Mohan

Abstract

This paper presents a reconfigurable biomimetic robot which is able to crawl and roll. The robot mimics the morphology of a huntsman spider that can transform between crawling and rolling by reconfiguring its legs. Terrain perception for reconfigurable biomimetic robots has not been studied in literature. This work tends to perceive and segment the terrain when the robot is crawling or in the steady state between rollings. A remote control system is designed with a server-client mechanism which can perform real-time image processing with GPGPU coding and develop a probabilistic framework for terrain perception. For validation, we test the system in both an indoor lab environment and a more uncontrolled outdoor environment. The results suggest that the system provides a trustable performance.

Keywords: Reconfigurable robot; Biomimetics; Terrain perception; Unsupervised terrain classification; Mean shift

Background

Reconfigurable robots capable of performing multi-state locomotion offer enormous potential with their versatility, fault tolerance, and efficiency for a variety of rugged missions in real world. There have been a lot of works on reconfigurable robotics [1-5].

The design of reconfigurable robots can be inspired by nature. Some attempts tried to develop reconfigurable biomimetic robotic platforms with rolling and crawling capabilities in the cases of BiLBIQ [6]. However, these efforts had been focused completely on mechanism design with almost no effort associated with perception or autonomous features. However, real-world deployments of these reconfigurable robots often require some intelligent capabilities, such as terrain perception and understanding, that go beyond purely mechanism design. Unfortunately, integrating complex reconfigurable design mechanisms with perception introduces a lot of new research challenges.

In terrain perception field, researchers choose in general two directions, namely, predictive and reactive terrain perceptions. Predictive terrain perception is done in a pre-entry manner, that is, the robot starts to perceive the terrain in front of itself. On the other hand, reactive terrain perception considers post-entry classification or recognition of terrain on which the robot perceives. In the case of reactive terrain perception, usually, researchers use inertial measurement unit (IMU) like sensors as described in [7]. The authors have modeled the vibration to classify terrain. Amongst other important works, [8-12] are notable. There are multiple sensors used in all these works, such as gyros, accelerometers, encoders, motor current and voltage sensors, multi-axis force sensor, tachometer etc. The basic disadvantage within reactive terrain perception methologies is that before entering a terrain, the robot cannot do the processing.

On the other hand, for path or trajectory planning, predictive terrain perception is required and usually an elevation map is constructed. Some important works using this approach are [10,13-15], etc. In predictive terrain perception, usually, the sensors used are stereo-camera, laser, IR sensor, etc. These sensors can provide a 3D point cloud or a 2D point cloud along with (or not) a camera color information.

Iagnemma et al. in [10] had used both reactive sensors and predictive sensors for the estimation of two key terrain parameters, cohesion, and internal friction angle. Whereas, Ojeda et al. in [11] used both types of sensors for both terrain characterization and classification. Every sensor modality was trained/learned with respect to different

*Correspondence: arnab_sinha@sutd.edu.sg
Singapore University of Technology and Design, 20 Dover Drive, Singapore, Singapore

kinds of terrains with a neural network and henceforth can be used with some error in terrain classification or characterization.

A particular work which is worth mentioning was done by Fukuoka et al. [16]. The authors have tried to emulate the muscle structure through a bio-inspired mechanical design and to provide flexibility with respect to the terrain characteristics.

In the case of predictive terrain perception algorithms, there are again two broad classes, namely, supervised and unsupervised. Mostly, the works, such as [17-19], etc., are supervised. Like our work, [19] also uses only one camera. However, we try to move our algorithm from supervised texture classification to unsupervised terrain perception. Moreover, we concentrate our terrain perception algorithm on a single camera-based algorithm, which do not have any localization information, no mapping information, and no geometry information (i.e., no 3D/2D point cloud).

Recently, a family of reconfigurable robotic platforms (i.e., Scorpio) capable of crawling and rolling locomotion have been developed. These robots mimic the morphology of a huntsman spider that can transform between crawling and rolling by reconfiguring their legs. In this paper, we present our effort to achieve terrain perception in the Scorpio robot using one monocular camera.

The rest of the paper is organized as follows. The 'Design of the robot' section describes the design specification of our robot. Thereafter, in the 'Terrain perception' section, we describe our methodology, whose results are described in the 'Results and discussion' section. Finally, the 'Conclusions' section concludes the paper with discussions and future research directions.

Huntsman spider

The robot to be discussed in this paper is inspired by a species of huntsman spider, *Cebrennus rechenbergi*. As shown in Figure 1, other than crawling, this kind of spider is also able to roll. The rolling locomotion of such spider was discovered by Ingo Rechenberg from TU Berlin [6]. The habitat of *C. rechenbergi* is the sand dunes of the Erg Chebbi desert in Southern Morocco, boundary to the Sahara Desert.

Normally, the spider crawls with eight legs. However, if provoked or threatened by an external stimulus, the spider can escape by doubling its normal crawling speed using forward or backward flips similar to acrobatic flic-flac movements used by gymnasts with the use of its eight legs simultaneously. What is the most curious thing is that the spider turns somersaults to move independently from surrounding conditions, which means that it does not need a slope to initiate the rolling process by using the gravitational force or does not need to walk a little first or perform a startup gesture to trigger the rolling locomotion.

So far, it could also be observed that only if very certain situations occur, the spider starts to switch from a normal crawling locomotion to this unique rolling locomotion in a somersaulting manner. Such situations might be the appearance of a predator, for example, the fennec

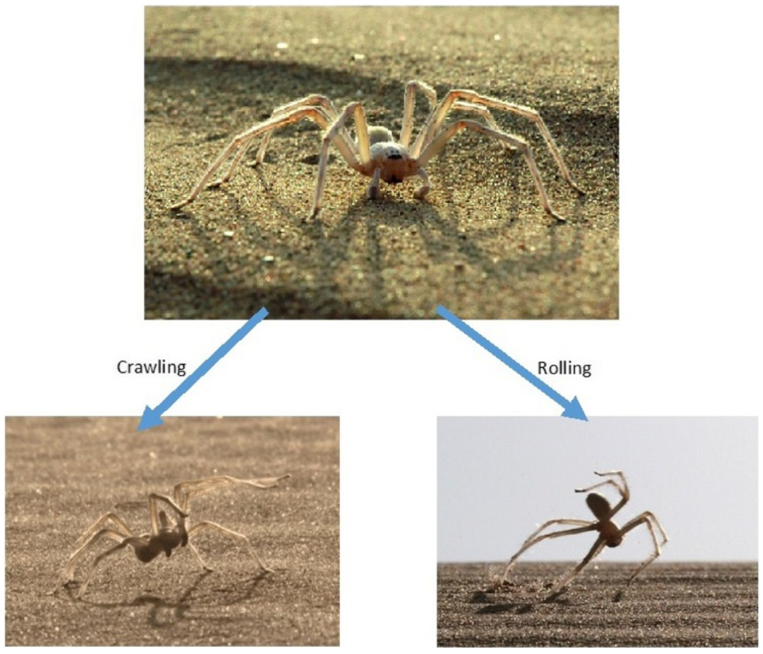


Figure 1 Huntsman spider performing crawling and rolling locomotion.

fox and sand cat, or meeting a conspecific. But it has not been researched that whether the sex of the conspecific plays a role or not. Besides, the circumstances that the spider makes use of the rolling locomotion, for instance, to change positions, to hunt down its prey, or to search for its tunnel, could not be observed either unfortunately.

Methods

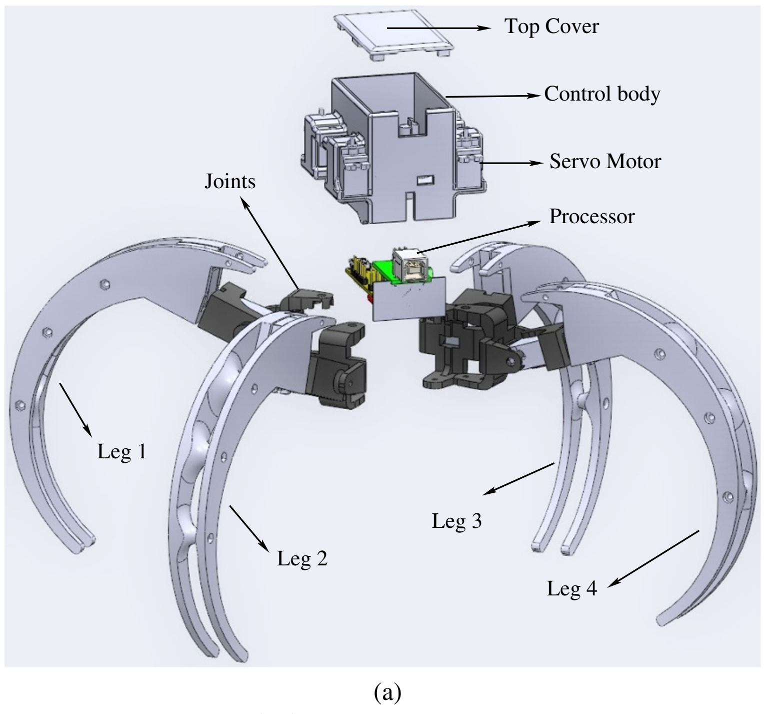
Design of the robot

The section presents the mechanical design and system architecture of the Scorpio robot.

Mechanical design

The design of Scorpio robot is based on the real huntsman spider introduced above, which is capable of crawling and rolling. The huntsman spider has eight legs, but we simplify the design to four legs which are sufficient to perform crawling and rolling.

Figure 2 contains a part-by-part view of Scorpio robot showing the assemblies. It is observed that the Scorpio robot consists of four legs (tibia), four servo covers and joints (femur), four main joints (coxa), and a body. The processor, controller, and sensors are placed inside



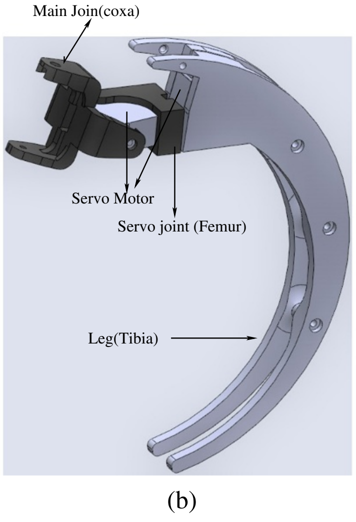


Figure 2 Design of Scorpio robot in exploded view with assembly parts. (a) Robot parts. **(b)** Robot leg parts.

the body which is made from PLA plastic. Twelve servo motors are used in this Scorpio robot to generate locomotion. Each leg is mounted with three servos, so it has 3 degrees of freedom. These legs are able to rotate and transform from crawling to rolling gaits. The specifications of the Scorpio robot are listed in Table 1.

For crawling motion, the Scorpio robot opens up its four legs as shown in Figure 3a. The crawling involves 2 degrees of freedom. Transformation from crawling pose to cylindrical exoskeleton for rolling requires a motion of 3 degrees of freedom. The Scorpio robot uses its legs to push from the ground and shift the center of gravity to achieve the rolling motion with 1 degree of freedom. The rolling speed of the Scorpio robot doubles the rate of crawling speed.

System architecture

For the autonomous movement and reconfiguration, we need to perceive a terrain. Since the platform in this work is a lightweight and small-sized (around 15 cm) robot, and the robot should reconfigure during movement, one cannot place a laser sensor or any kind of range sensors because usually they are too heavy. Moreover, ground-level sensors provide some good data with respect to physical interactions. Based on these perspectives, we choose to build up a vision system, which can provide reliable data for terrain perception. The vision system should be lightweight and small-sized to minimize the influence on the locomotion and control of the robot. Thus, we further alleviate this problem by restricting the vision system with only one camera situated on top of the robot.

In this work, we have chosen a wireless network camera to stream video to a local network computer. Thereafter, we process the frames on this remote processor. The Ai-Ball camera [20] that we use is shown in Figure 4 with the robot. When the robot is crawling (Figure 4a), the camera can always be looking ahead. Whereas in the case of rolling gait (Figure 4b), the view might be blocked when the camera is rotated underneath. The camera may rotate along yaw and pitch rotation axis when the robot is crawling and rolling. Therefore, we cannot have any *a priori* geometry assumption relating the image to the world. After one roll of 360°, the robot gets back to the *stand up* position and checks (for now the decision is coming from human operator) the terrain for further rolling or crawling decision.

Table 1 Specifications of the Scorpio robot

	Specifications
Controller	Arduino Mini Pro 328
Servo motor	JR ES 376
Servo controller	Pololu-Micro Maestro 18-channel USB servo controller
Camera	WiFi Ai-Ball Cam
Battery	LiPo 1,200 mah 7.4 v
ZigBee	XBee Pro S1, Digi International
Full body material	Polylactic acid or polyclactide (PLA)
Dia (while rolling) in mm	168 mm
L × W × H (while walking) in mm	230 mm × 230 mm × 175 mm
Weight (full weight) in g	430 g

In Figure 5, we show the overall architechture of the system. XBee [21] is used for communication between the robot and the remote computer for message passing. The robot has a Arduino Mini [22] controller to drive the servo motors for the creation of the gaits, for different motions and reconfigurations. This controller also sends its current state (according to its movement) to the remote desktop.

The remote desktop receives information from the robot and camera, respectively, as shown in green arrows in Figure 5. These two information are then managed within ROS [23], for time synchronization, and software stability. After considering all the inputs, if the system tries to take feedback, it asks the human operator and, thereafter, transfers the required morphological gait commands to the robot, as shown in blue arrow in Figure 5. The time synchronization within different inputs and related decision-making process of our semi-autonomous system (developed within ROS) is required for stable robot performance.

Terrain perception

In this section, we describe the image perception/segmentation based on the mean shift algorithm.

Mean shift image segmentation

In this work, the algorithm for semi-autonomy depends on image segmentation, and therefore, we describe earlier works with respect to mean shift-based image segmentation in the following.

Mean shift has been introduced by Yijong Cheng in 1995 [24]. Thereafter, Comaniciu and Meer in [25] popularized the mean shift algorithm by providing a theory for robust feature space analysis. Since then, this algorithm has been used, studied, and further developed by a number of researchers, such as in [26], Han et al. have used it for object tracking. Yang et al. further improved its performance by introducing a new similarity measure in [27]. Some other important works involve image segmentation [28], visual tracing [29], stereo matching [30], etc.

Here, we first describe the mean shift algorithm. Given n data points (or vectors), $\mathbf{x}_i; i = 1, 2, \ldots, n$, where each data vector $\mathbf{x_i}$ lies in a d-dimensional space say R^d. Here, R represents state space of each dimension, i.e., in case of a color channel of a RGB image, $R = \{0, 1, 2, \ldots, 255\}$. The

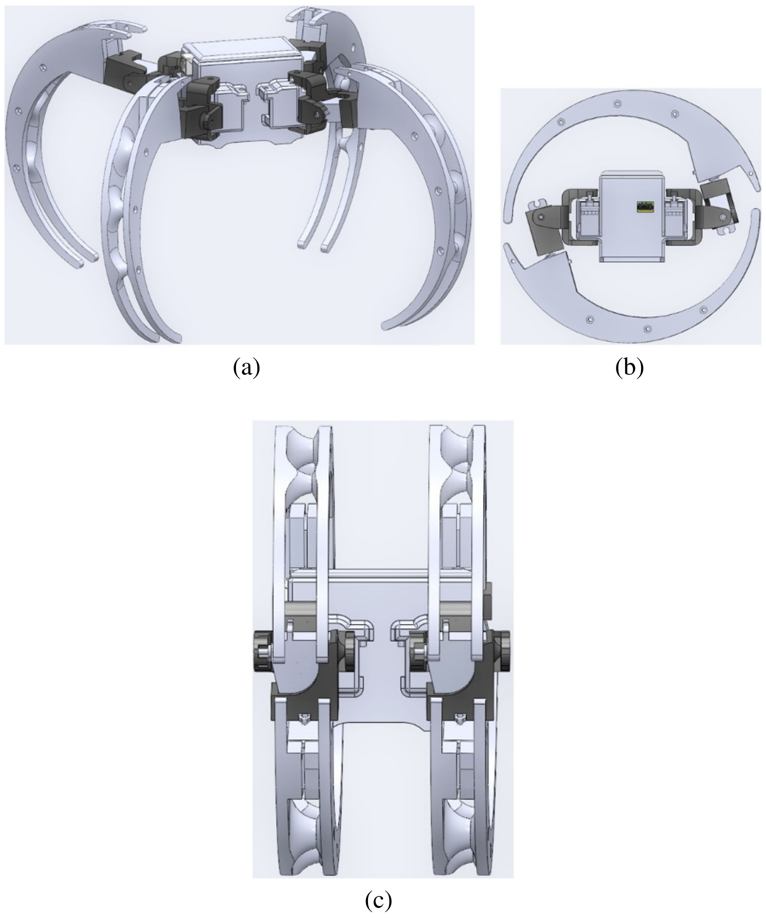


Figure 3 Design of Scorpio robot in crawling and rolling gestures. (a) Crawling configuration. **(b)** Rolling configuration - side view. **(c)** Rolling configuration - front view.

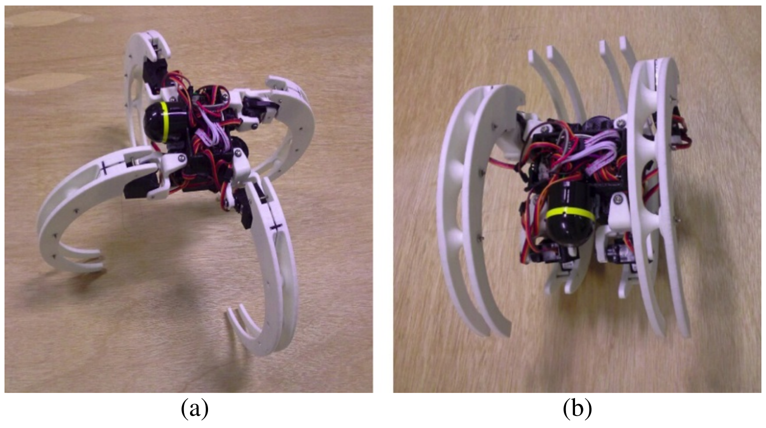


Figure 4 Scorpio robot with wireless network camera Ai-Ball in different gaits. (a) Crawling gait. **(b)** Rolling gait.

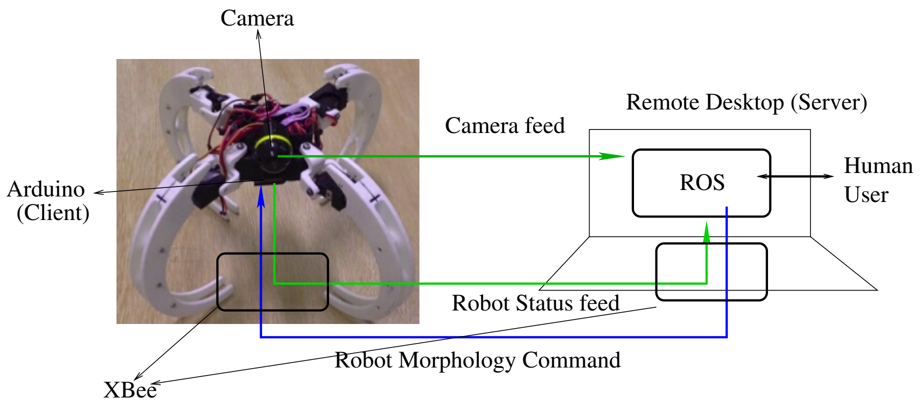


Figure 5 Overall system architecture. The arrows show the flow of information.

multi-variate kernel density estimator with kernel function $K(\mathbf{x})$ and a $d \times d$ bandwidth matrix H, at a point $\mathbf{x}$ can be computed as:

$$\hat{f(\mathbf{x})} = \frac{1}{n}\sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{x}_i) \tag{1}$$

where, $K_H(\mathbf{x}) = |H|^{-1/2}K(H^{-1/2}\mathbf{x})$. The kernel function $K(\mathbf{x})$ should satisfy the following conditions:

$$\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1 \quad \lim_{||\mathbf{x}|| \to \inf} ||\mathbf{x}||^d K(\mathbf{x}) = 0 \tag{2}$$

$$\int_{R^d} \mathbf{x}K(\mathbf{x})d\mathbf{x} = 0 \quad \int_{R^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x})d\mathbf{x} = c_k\mathbf{I} \tag{3}$$

where, c_k is a constant, $\mathbf{I}$ is the identity matrix, and $||.||$ is the norm defined on R^d space. In terms of popularity, product kernel are mostly used due to its mathematical and practical simplicity, it is defined in the following:

$$K^P(\mathbf{x}) = \prod_{j=1}^{d} K_1(x_j). \tag{4}$$

Moreover, we have used Gaussian kernel here, i.e.:

$$K_1(x_j) = \frac{1}{\sqrt{2\pi h_j^2}} \exp \frac{(x_j - x_{i,j})^2}{2\pi h_j^2}, \tag{5}$$

where $x_{i,j}$ is the jth element of data vector $\mathbf{x}_i$, and x_j is the jth element of data vector x, where we are estimating the density. Now, we define function $g(x) = -\frac{dk(x)}{dx}$ as the derivative of the single variable kernel profile. Here, we assume that $g(x)$ exists for all values of $x \in R$. Given the definition of $g(x)$, we define the mean shift in the following equation:

$$\mathbf{m}_{h,g}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(||\frac{(\mathbf{x}-\mathbf{x_i})}{h}||^2\right)}{\sum_{i=1}^{n} g\left(||\frac{(\mathbf{x}-\mathbf{x_i})}{h}||^2\right)} - \mathbf{x}. \tag{6}$$

In mean shift segmentation, a new data vector is created according to the following equation:

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{m}_{h,g}. \tag{7}$$

Through iterations of Equation 7, the data vectors converge to its mode, i.e., $\mathbf{x}_i \to M_i$, within the original density function, which signifies a cluster. The convergence depends solely upon the bandwidth parameter h, which is also known as the smoothness parameter. Since, as one increases the value of this smoothness parameter, the convergent mode becomes more global, that is, it smoothes out local perturbations. In this work, we have considered three color channels (RGB) and two spatial channels (XY). Therefore, in our case, $d = 5$, and the state space of color channels is $L = \{0, 1, \ldots 255\}$ and spatial channel is $\Re$. The smoothness parameters for color channels are equal to each other and controlled by one parameter h_C, similarly, the smoothness parameter for each spatial channels is h_S. After convergence, we color the given image according to the average color value of the local mode or cluster. In the next section, we use these identified modes for final terrain perception.

Computational complexity of mean shift algorithm is quite extensive, and hence, it is not so popular within real-time algorithms. Recently, researchers have identified the parallel power of mean shift algorithm, and with the advance of general purpose graphics processing unit (GPGPU) architecture, one can have almost real-time performance with CUDA coding, [31]. In this work, we use this GPGPU mode for computing. The actual computational complexity of this approach comes from the calculation of kernel weight according to given input, i.e., Equation 1. Here, the computational complexity is $O(n)$, for each data $\mathbf{x}$. Therefore at each iteration, Equation 7 also has a computational complexity of $O(n)$ for each data point $\mathbf{x}$. In case of a serial computational algorithm, this computational complexity is scaled up by the number of

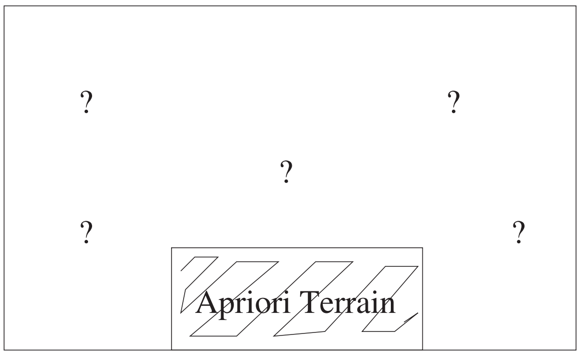


Figure 6 Camera image with a priori terrain region assumption.

data points, whereas in the case of parallel computing, it remains the same. Within an image, the number of pixels represent the number of data points, and henceforth with parallel computing architecture, one can reduce the huge computational burden.

Terrain perception

Unsupervised terrain perception is one of the fundamental requirements of future robotics. For the perception of terrain, we need some *a priori* assumptions on which the algorithm could be based. The two assumptions about terrain are listed in the following:

- The terrain is mostly flat at least locally.
- The robot is standing on a terrain, that is in its immediate front, there is no obstacle.

Given these two *a priori* assumption, we build up a probabilistic framework for the terrain estimation. As shown in Figure 6, we describe our *a priori* assumption about the terrain. Moreover, from sub-section, we know the modes of each pixel, say M_s, where $s = (i,j) \in \Omega$ is a pixel site. Now, we define our *a priori* terrain assumption region as $\Omega_T \subset \Omega$. We are now able to define probability of mode $M_s; s \in \Omega$, according to all the modes defined within Ω_T, as described in Equation 8. Thereafter, we use this likelihood measure to make a decision about whether a particular pixel is representing terrain or not. For this, one can use a hard threshold, i.e., if $P(M_s) \geq T$th, where Tth is the terrain likelihood threshold, we say $s = (i,j)$ represents a terrain pixel. This threshold is estimated according to Equation 9.

$$P(M_s) = \frac{1}{|\Omega_T|} \sum_{s_t \in \Omega_T} K_H(M_{s_t} - M_s) \tag{8}$$

$$T_{th} = \min P(M_{s_t}); s_t \in \Omega_T \tag{9}$$

Once we segment the image pixels in terrain and non-terrain classes, we can further segment non-terrain class into two classes using the above approach. It is assumed that there are regions which are probable terrain next to the terrain class. It is worth mentioning that the robot is not a wheeled robot; hence, the camera can rotate along yaw, pitch, and roll axis. That is, we cannot use vertical lines in the image as those in real world.

Semi-autonomous motion transformation

After terrain classification, we have three classes, namely terrain, probable terrain, and obstacle. Then, we can proceed to semi-autonomous motion transformation. Some terrains which have less friction best suit to robot rolling, whereas there are some rugged terrains where the robot cannot rotate and crawl. Such ruggedness is difficult to estimate just from an image; therefore, we leave this decision to the human operator. As the segmented terrain area in front of the robot becomes smaller, we notify the human operator for suggestions. The human operator then takes a decision to either change crawling direction or roll.

Results and discussion

In our experiments, we have considered two types of flat terrains, since our robot is not capable to move on a sloppy terrain or stair-like terrain or big pebble terrain. One terrain is in indoor environment, using normal carpet, and the other terrain is outside, where a cemented small pebble terrain and a much smoother marble terrain is situated.

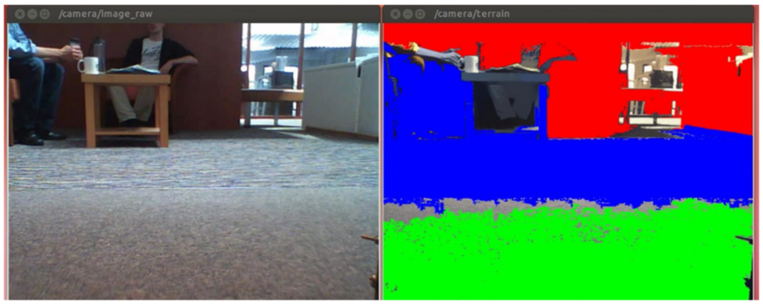


Figure 7 Terrain segmentation/perception in indoor environment. Green is a terrain, blue defines a probable terrain, and red signifies an obstacle.

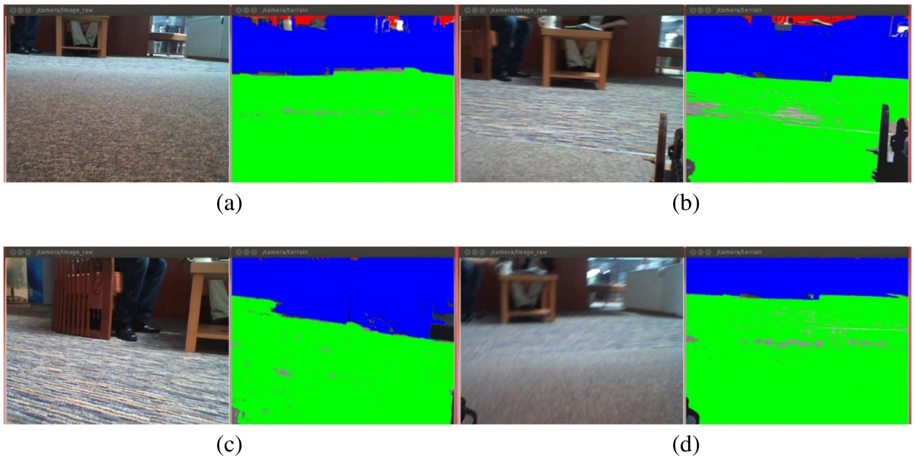


Figure 8 Terrain segmentation/perception results of a serial of motions. Green is a terrain, blue defines a probable terrain, and red signifies an obstacle. **(a)** Moving straight. **(b)** Robot leg is not detected as terrain. **(c)** Turned motion. **(d)** Again straight.

There are two main parameters within mean shift segmentation algorithm as mentioned in the 'Mean shift image segmentation' section, i.e., color smoothness parameter h_C and spatial smoothness parameter h_S. For our case of implementation, $h_C = 40$ and $h_S = 40$ are chosen. The image processing speed is less than 20 ms which can be considered to be real time.

The terrain perception can be performed during the crawling motion. When the robot is rolling, the camera is also in rolling state. Thus, the feedback from the camera cannot be considered. What we treat is that when the rolling motion is finished, the robot and camera are in a standing state, so the terrain perception is performed again.

With a single camera and without localization algorithm, it would be very difficult to have any geometrical information of the terrain. Moreover, the selection of the terrain also depends on the robot's capability. In this work,

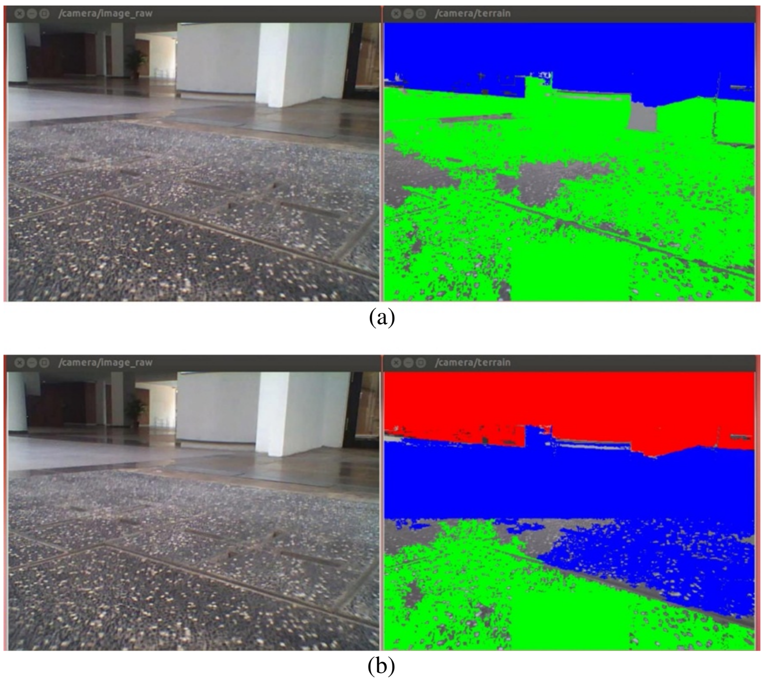


Figure 9 Initial results with outdoor terrain perception. Green is a terrain, blue defines a probable terrain, and red signifies an sobstacle. **(a)** Initial result - different terrain texture not segmented. **(b)** Initial result - different terrain texture not well segmented.

for the sake of simplification, the perception experiments are performed on flat terrains, with a rugged one and a smooth one.

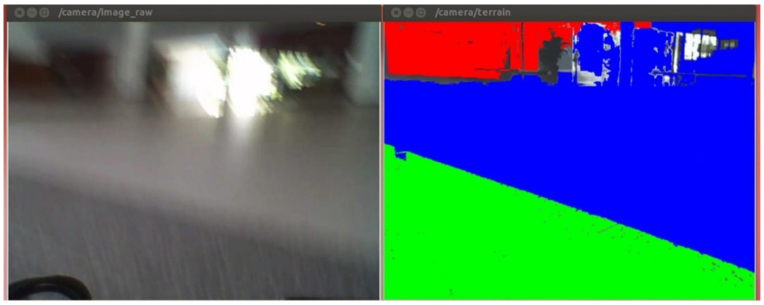
Indoor experiment

In an indoor environment, Figure 7 provides an example of the terrain perception result. We colorize the pixels in four classes: green signifies an identified terrain, blue represents a possible terrain, and red is an obstacle. Other pixels are shown with their mode color, since we could not decide whether they belong to any class or not. The same color representation is followed for all other results. As can be seen from the figure, the two different carpets are well segmented. Figure 8 shows the results as the robot is crawling in the indoor environment. Sometimes, the robot legs go in front of the camera. But they are not recognized as the terrain class. From the experiment results, we can observe that in indoor carpet case, the terrain perception works quite good.

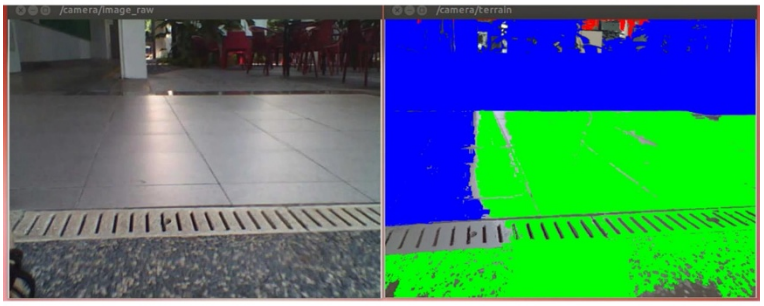
Outdoor experiment

Outdoor experiments are more difficult due to a lot of uncontrolled variations and disturbances. Figure 9 shows the results as the robot starts crawling on a more rugged terrain and approaching to a smoother terrain. Here, the segmentation, in between two terrains, are not so good, as depicted in the figure.

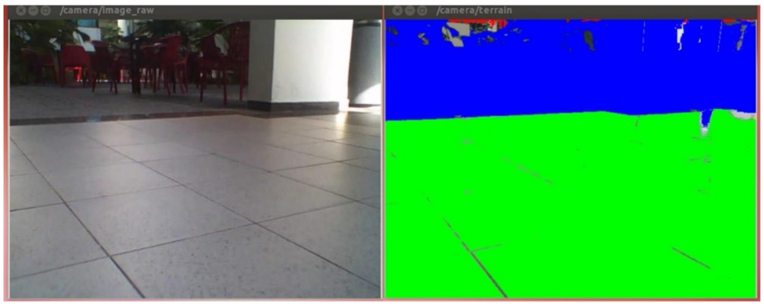
In Figure 10, we show the results of our algorithm as it is moving from rugged terrain to a smoother terrain from different angles. Figure 10a shows the motion blur, and most notably, our perception algorithm works quite well here. Figure 10b shows the effect of light reflection on



(a)



(b)



(c)

Figure 10 Effect of motion blur and reflection on terrain perception. Green is a terrain, blue defines a probable terrain, and red signifies an obstacle. **(a)** Effect of motion blur on terrain perception. **(b)** Effect of sunlight reflection on terrain perception. **(c)** Effect of sunlight on terrain perception - best result.

smoother surface. With respect to surface reflection, we show our best result in Figure 10c.

Normal execution time per frame of the video is less than 20 ms, which is real-time. One can also decrease the time of the algorithm, by changing the bandwidth parameter, required for mean shift-based image segmentation, but the results are not very good.

Conclusions

This work reports the study of real-time terrain perception for the reconfigurable biomimetic robot. The robot can mimic the huntsman spider to crawl and roll by its legs. Using the single camera situated on the robot, the robot can perceive different terrains. There are some demerits of our system, which are, our algorithm is not perfect all the time; but with a consensus over multiple test on same scenario, we can stabilize it. The advantages of our algorithm are it is fast, real-time, and especially un-supervised. That is, it needs not any training *a priori* for a particular unknown terrain. The experimental results show that in both indoor or outdoor condition, if the light is not reflected, our algorithm achieves good terrain perception, even when there is a motion blur.

This work is a fundamental step to build more advanced and lightweight reconfigurable biomimetic robots. In the future, we will try to include IMU sensor to extend the perception module with both reactive and predictive sensors.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

AS carried out the development of the 'Results and discussion' section, and NT has contributed in developing the section 'Mechanical design'. AS and NT has jointly written the 'Background' section. Finally, RME has guided the original motivation and sought through the development of the overall work presented within this paper. All authors read and approved the final manuscript.

Acknowledgement

We hereby acknowledge that this work is supported by the Temasek Laboratory and SUTD-MIT International Design Centre at Singapore University of Technology and Design.

Received: 15 July 2014 Accepted: 6 November 2014
Published online: 26 November 2014

References

1. Nansai S, Rojas N, Elara MR, Sosa R (2013) Exploration of adaptive gait patterns with a reconfigurable linkage mechanism. In: Intelligent Robots and Systems (IROS) 2013 IEEE/RSJ International Conference On. IEEE, Japan. pp 4661–4668
2. Yim M, Shen W-M, Salemi B, Rus D, Moll M, Lipson H, Klavins E, Chirikjian GS (2007) Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot Autom Mag 14(1):43–52
3. Murata S, Kurokawa H (2007) Self-reconfigurable robots. IEEE Robot Autom Mag 14(1):71–78
4. Moubarak P, Ben-Tzvi P (2012) Modular and reconfigurable mobile robotics. Robot Autonom Syst 60(12):1648–1663
5. Gilpin K, Kotay K, Rus D (2008) Miche: modular shape formation by self-disassembly. Int J Robot Res 27:345–372
6. King RS (2013) BiLBIQ: a biologically inspired robot with walking and rolling locomotion, Biosystems & Biorobotics, vol. 2. Springer. http://www.springer.com/engineering/robotics/book/978-3-642-34681-1
7. Tick D, Rahman T, Busso C, Gans N (2012) Indoor robotic terrain classification via angular velocity based hierarchical classifier selection. In: Robotics and Automation (ICRA), 2012 IEEE International Conference On. IEEE, Minnesota, USA. pp 3594–3600
8. Garcia Bermudez FL, Julian RC, Haldane DW, Abbeel P, Fearing RS (2012) Performance analysis and terrain classification for a legged robot over rough terrain. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference On. IEEE, Portugal. pp 513–519
9. Rebula JR, Neuhaus PD, Bonnlander BV, Johnson MJ, Pratt JE (2007) A controller for the littledog quadruped walking on rough terrain. In: Robotics and Automation, 2007 IEEE International Conference On. IEEE, Roma, Italy. pp 1467–1473
10. Iagnemma K, Kang S, Shibly H, Dubowsky S (2004) Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. IEEE Trans Robot 20(5):921–927
11. Ojeda L, Borenstein J, Witus G, Karlsen R (2006) Terrain characterization and classification with a mobile robot. J Field Robot 23:103–122
12. Hoepflinger MA, Remy CD, Hutter M, Spinello L, Siegwart R (2010) Haptic terrain classification for legged robots. In: Robotics and Automation (ICRA), 2010 IEEE International Conference On. IEEE, Alaska, USA. pp 2828–2833
13. Belter D, Skrzypczynski P (2010) Rough terrain mapping and classification for foothold selection in a walking robot. In: Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop On. IEEE, Bremen, Germany. pp 1–6
14. Poppinga J, Birk A, Pathak K (2008) Hough based terrain classification for realtime detection of drivable ground: Research articles. J Field Robot 25(1–2):67–88
15. Manduchi R, Castano A, Talukder A, Matthies L (2005) Obstacle detection and terrain classification for autonomous off-road navigation. Autonomous Robots 18(1):81–102
16. Fukuoka Y, Kimura H, Hada Y, Takase K (2003) Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. Int J Robot Res 22(3–4):187–202
17. Best G, Moghadam P, Kottege N, Kleeman L (2013) Terrain classification using a hexapod robot. In: The Australasian Conference on Robotics and Automation (ACRA). ARAA, Sydney, Australia
18. Zenker S, Aksoy EE, Goldschmidt D, Worgotter F, Manoonpong P (2013) Visual terrain classification for selecting energy efficient gaits of a hexapod robot. In: Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME international conference on. IEEE, Wollongong, Australia. pp 577–584
19. Filitchkin P, Byl K (2012) Feature-based terrain classification for littledog. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference On. IEEE, Vilamoura-Algarve, Portugal. pp 1387–1392
20. LTD. T..I: Ai-Ball (2010). http://www.thumbdrive.com/aiball/index.html
21. http://www.digi.com: XBee®802.15.4. http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#overview
22. Banzi M (2008) Getting Started with Arduino, III edn. Make Books - Imprint of: O'Reilly Media, Sebastopol
23. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. In: ICRA workshop on open source software. IEEE, Kobe, Japan
24. Cheng Y (1995) Mean shift, mode seeking, and clustering. IEEE Trans Pattern Anal Mach Intell 17(8):790–799. doi:10.1109/34.400568
25. Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis. IEEE Trans Pattern Anal Mach Intell 24(5):603–619
26. Han B, Comaniciu D, Zhu Y, Davis L (2004) Incremental density approximation and kernel-based bayesian filtering for object tracking. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on, vol. 1. IEEE, Washington DC, USA. pp 638–6441. doi:10.1109/CVPR.2004.1315092,
27. Yang C, Duraiswami R, Davis L (2005) Efficient mean-shift tracking via a new similarity measure. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE computer society conference on, vol. 1. IEEE, San Diego, CA, USA. pp 176–1831. doi:10.1109/CVPR.2005.139

28. Sfikas G, Nikou C, Galatsanos N, Heinrich C (2011) Majorization-minimization mixture model determination in image segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE conference on. IEEE, Colorado, USA. pp 2169–2176. doi:10.1109/CVPR.2011.5995349
29. Lu L, Hager GD (2007) A nonparametric treatment for location/segmentation based visual tracking. In: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE conference on. IEEE, Minnesota, USA. pp 1–8. doi:10.1109/CVPR.2007.382976
30. Mei X, Sun X, Dong W, Wang H, Zhang X (2013) Computer Vision and Pattern Recognition (CVPR), 2013 IEEE conference on. IEEE, Ohio, USA. pp 313–320
31. Corporation N (2014) NVIDIA CUDA: Parallel Programming and Computing Platform. http://www.nvidia.com/object/cuda_home_new.html

doi:10.1186/s40638-014-0023-2
Cite this article as: Sinha *et al.*: Terrain perception for a reconfigurable biomimetic robot using monocular vision. *Robotics and Biomimetics* 2014 **1**:23.