Real-Time On-Board Image Processing Using an Embedded GPU for Monocular Vision-Based Navigation

Matías Alejandro Nitsche and Pablo De Cristóforis

Buenos Aires University, Faculty of Exact and Natural Sciences,
Computer Science Department
{mnitsche,pdecris}@dc.uba.ar

Abstract. In this work we present a new image-based navigation method for guiding a mobile robot equipped only with a monocular camera through a naturally delimited path. The method is based on segmenting the image and classifying each super-pixel to infer a contour of navigable space. While image segmentation is a costly computation, in this case we use a low-power embedded GPU to obtain the necessary framerate in order to achieve a reactive control for the robot. Starting from an existing GPU implementation of the quick-shift segmentation algorithm, we introduce some simple optimizations which result in a speedup which makes real-time processing on board a mobile robot possible. Performed experiments using both a dataset of images and an online on-board execution of the system in an outdoor environment demonstrate the validity of this approach.

Keywords: monocular vision-based navigation, image segmentation, GPU.

1 Introduction

Vision-based robot navigation has long been a fundamental goal in both robotics and computer vision research in last years. Most vision-based navigation techniques assume that a sequence of images is acquired during a human-guided training step that allows the robot to build a map of the environment. However, this is a tedious process as it involves human intervention every time a robot moves to a new workspace. Therefore, to achieve a completely autonomous navigation system it is necessary to remove human intervention [1] [2].

The final aim of this work is to facilitate a mobile robot equipped only with a monocular camera to autonomously drive through a naturally delimited path. This robot should perform all processing on-board, given real-time constraints imposed by the robot motion.

While in previous works classification in navigable or non-navigable ground is done at the pixel level [3], in this work we follow more recent proposals that segment images into regions to perform a better classification [4] [5]. The advantage of the segmentation is that the pixels belonging to a certain region have

L. Alvarez et al. (Eds.): CIARP 2012, LNCS 7441, pp. 591–598, 2012.

low internal contrast. This facilitates the search of the navigable terrain, since it is fair to assume that this area is more or less uniformly colored or textured and the edges will most likely be of high-contrast. On the other side, the disadvantage is usually the associated high computational cost. For this reason, real-time image segmentation is difficult to achieve using low-power embedded processors on autonomous robots, which is the current target of this work.

In this work we present a new image-based navigation method for guiding a mobile robot equipped only with a monocular camera through a naturally delimited path. The method is based on segmenting the image and classifying each super-pixel to infer a contour of navigable space. We use a low-power embedded GPU to obtain an acceptable framerate in order to achieve a reactive control for the robot. Starting from an existing GPU implementation of the quick-shift segmentation algorithm [6], we introduce some optimizations which result in a speedup of approximately five times. This increased speed makes real-time processing on board a mobile robot possible.

2 Related Work

There are previous works that perform visual navigation using only monocular system and pixel classification. If we can assume that the robot is operating on a flat surface and all objects have their bases on the ground, the problem is reduced to classifying pixels into two classes: obstacle or traversable floor. This approach, that is suitable for robots that operate on benign flat terrains, has been used in in a variety of works. In [3] classification is done at the pixel level. First the image is filtered with a Gaussian filter. Second the RGB values are transformed into the HSV (hue, saturation, and value) color space. In the third step, an area in front of the mobile robot is used for reference and valid hue and intensity values inside this area are histogrammed. In the fourth step, all pixels of the input image are compared to the hue and intensity histograms. A pixel is classified as an obstacle if its hue or intensity histogram bin value is below some threshold. Otherwise the pixel is classified as belonging to the ground. This method is fast, as no complex computation is involved. The main drawback of this method is it is very unstable at the robot's movements and sensitive to noise.

Thus, the idea of segmenting the image into a number of super-pixels (i.e., contiguous regions with fairly uniform color and texture) arises. In [4] a graph-based image segmentation algorithm [7] is used. Once the image is over-segmented in super-pixels, each one is labeled as belonging to the ground or non-ground region using the HSV histogram approach as in [3]. While this method is more stable and robust, it is quite computational expensive, so the exploration algorithm that uses this method has to stop the robot periodically to capture and process images of the workspace. Using the same image segmentation algorithm, in [5] super-pixels that are likely be classified equal are grouped into constellations. Constellations can be the labeled as floor or non-floor with an estimator of confidence depending on whether all super-pixels in the constellation have the same

label. This is a more robust method, but computationally expensive. In contrast to these methods, this work proposes a real-time segmentation algorithm that allows a reactive control for guiding the robot.

3 Proposed Method

The global idea of the proposed method can be summarized as follows. First, the image obtained from the camera is smoothed using a median blur filter[1]. Then, the image is segmented into a number of super-pixels, allowing the discrimination of individual chunks of pixels with low internal contrast. Given an example of what the navigable space looks like (by using a sub-region of the image corresponding to the area directly in front of the robot), each segment is classified as being similar to it or not. After individual classification, the resulting positively classified group of super-pixels nearest to the example area are assumed to correspond to the most likely navigable path. Finally, by computing the contour of this area, a motion line can be extracted and used for reactive control. A control law is defined that restricts the robot to remain inside this detected navigable area. From these steps, the most computationally demanding corresponds to the image segmentation. Therefore, we focus on optimizing this portion of the method to allow real-time execution.

3.1 Image Segmentation

The quick-shift segmentation algorithm is a simpler and faster alternative to other segmentation algorithm. In listing 1.1 a pseudo-code implementation is shown. The algorithm first computes the density ρ of each pixel, which is a measure of local contrast in the vicinity of size 3σ. Next, each pixel is linked to the nearest neighbor in a vicinity of τ which has a higher density than the current one. In this fashion, trees are formed where each root is a pixel with the highest local contrast in the vicinity. Each tree then represents a segment or super-pixel.

Since with this algorithm the computation can be performed in parallel for each pixel in the image, it is a good candidate for a GPU implementation. In fact, such an implementation already exists and reports considerable advantage when compared to the CPU-based version[8]. The code used in this work by Fulkerson and Soatto is available on line. Moreover, this code actually includes a further speedup proposed by James Fung where authors report a $2x$ relation to the original. This code was used in the present work, but introducing a series of simple optimizations, obtaining a speedup of up to four times when performing all experiments on a low-power embedded GPU (which is used in the experiments with the robot).

[1] In contrast to the commonly used Gaussian blur, the median blur preserves high contrast edges, which is important for road-detection.

```
function density
 for i in all pixels
   ρ_i = 0
   for j in neighborhood of size 3σ
     ρ_i += exp(−||RGB[i]−RGB[j]||^2 / (2∗σ^2))

function neighbors
 for i in all pixels
   for j in neighborhood of size τ
     if ρ_i > ρ_j and distance(i,j) is smallest among all neighbors
       d_i = distance(i,j)
       parent[i] = j
```

Listing 1.1. Quickshift algorithm pseudocode

The first optimizations that were introduced consist of careful tuning of the number of concurrent threads executed, the registers required for code compilation (which affects the efficiency of the thread scheduling and thus the paralellization level), taking into account the capabilities of the specific card to be used. The second main optimization that was introduced involves a simpler handling of out-of-bound accesses which arise when searching the neighborhood of pixels near the edges of the image. In the original implementation these were avoided explicitly, where in our case, the *clamping mode* of the texture memory is used. Here, these accesses simply return the nearest valid pixel in the image (effectively repeating pixels in the image outwards). By introducing this change, the code can be simpler and more efficient. Finally, memory accesses in general were reduced by delaying them up to the point where they were for certain to be required.

3.2 Super-Pixel Classification

Once a list of super-pixels has been obtained, the following task is to classify each as belonging to free or non-free space. For this classification, a positive example of the floor appearance is provided. As in [3], a sub-region of the image corresponding to the ground immediately ahead of the robot was chosen for this example. In this work a rectangular region is used, instead of a trapezoid, to simplify the computations.

In order to classify each super-pixel according to this example region, a measure is taken from each and compared. By establishing a certain threshold, the classification is performed. In this work the measure consists of a simple average of pixel values belonging to the segment and to the rectangular region. This measure is not as computationally demanding as other histogram-based methods generally used, and also produces satisfying results.

While pixel values are generally processed in RGB, alternative color spaces exist. In particular, the HSV color space is particularly useful since it separates the tonal and illumination aspects of the image. Even further, in most cases, the hue channel provides enough information to discern a path (for example, a gray

pavement road delimited by green vegetation). Since the RGB to HSV conversion can be computationally demanding if performed on all the pixels of each frame, in this work we reduce this cost by first computing the average in the RGB space and then converting the final results to HSV. Besides the computational aspect, this solves the problem of taking the average HSV value of a group of pixels, since the hue channel is actually a continuous measure that wraps around.

3.3 Control Law

After the super-pixels are classified, several contiguous unconnected regions may appear as possible ground. Among these, the candidate region which includes the center point of the rectangular example area is chosen.

In order to achieve a reactive control to guide the robot so that it maintains its position inside the chosen region, its contour in the image is obtained and processed in several steps. First, by going through every point of this contour from bottom to top (up to a predefined row, related to the horizon position), the horizontal middle position is computed using the left-most and right-most contour pixels. The resulting middle points are then fitted using a linear regression, which approximates the middle of the visible free space with a straight line called motion line. From this line, both its angle and its horizontal deviation with respect to the image's middle vertical are computed. These deviations are scaled in the range $[-1, 1]$ and are identified as ω and d, respectively.

Since these two values indicate the direction and position of the road with respect to the robot, a simple control law is used to compute turning (v_a) and forward (v_x) speeds which allow the robot to remain in the road:

$$v_a = \alpha \cdot \omega + \beta \cdot d$$
$$v_x = 1 - |v_a|$$

where α and β are constants between $[0, 1]$. In the tests performed, these constants were set to 0.5, giving equal impact by both values to the angular speed. In other words, when either the fitted road line deviates or turns, the robot will turn in the same direction. Regarding the forward speed, the control law generates a constant forward motion whenever the robot is not turning. Otherwise, the robot reduces (in an inverse proportion) its speed while turning.

4 Results

The system was tested using a previously recorded dataset and also on-line in an outdoor scenario. In the former case the road-detection capability of the system was evaluated. In the latter, the complete system was tested including the control-law that drives the robot, in order to ensure the capability of remaining inside the road limits.

The robot used in the experiments, the ExaBot[9], consists of a differential-drive base with an embedded Mini-ITX board (AT5ION-T Deluxe) and a firewire

camera (model 21F04, from *Imaging Source*) with a $3.5 - 8mm$ zoom lens. The embedded computer features a GPU capable of general purpose computing, using the CUDA technology from NVIDIA. This graphic card features 16 cores running at 1.1Ghz and with 256MB of memory.

While the camera is capable of capturing images at 640×480 px images at 15 fps, a smaller resolution of 320×240 px (at 30 fps) was chosen since it was enough for proper road detection. This smaller resolution also decreases computation times. The zoom lens was set-up at 3.5mm, providing a fairly wide-angle of view.

Additionally, an analysis of the computational times obtained by using the optimized segmentation step is included.

4.1 Performance Improvement

In this section we measure the execution speed of the segmentation algorithm, compared to the original GPU-targeted unmodified version. The segmentation algorithm depends on two parameters (σ and τ) which ultimately control segment size and affects computational speed. Through several tests with real-world images, it was experimentally found that the acceptable results were obtained with values $\sigma = 4$ and $\tau = 10$. Therefore, performance results are here presented with these fixed values.

The speedup obtained when compared to the original implementation running on the same low-power GPU is around 4.8 times. This relation does not change with different image resolutions since only an improved implementation of the same algorithm is used. For a 320×240 resolution (which was used in the on-line tests) the execution time of the segmentation step is around 145 milliseconds. This accounts for 60% of the total time required for one processing step, which is about 240 milliseconds. In the following experiments it can be seen that this it is enough for on-line processing on the robot.

4.2 Offline Tests

The dataset used for this test was produced by a *Pioneer* robot, on a several hundred meter long trajectory, on an outdoor park in Prague, Czech Republic[2]. This dataset was processed and the road contour along the middle line used for control was drawn. In figures 1(a) to 1(e) some example frames are presented[3]. While in some frames the system classified some distant parts as road (near the horizon), what is used for controlling the robot motion is the line fitted from the yellow middle points which are below of the last row configured by the user. This way, the road following is reduced to detecting the nearby edges and attempting to remain away from them.

[2] Dataset: `http://robotica.exp.dc.uba.ar/trac/exabot/export/85/trunk/src/gpu/floordetection/dataset/dataset.avi`

[3] Full video: `http://robotica.exp.dc.uba.ar/trac/exabot/export/85/trunk/src/gpu/floordetection/dataset/frontier.avi`

4.3 Online Tests

The system performance running on a robot was evaluated on an outdoor scenario, where the control-law and system reactiveness was put to test. The trials consisted of several initial positions were the robot was in the middle of the road and near the sides, both looking straight forward and also deviated towards a side. This allowed testing not only of the stability of a good initial configuration but also of extreme cases (which did not normally occur since the robot always remained on track) where the robot quickly returned to the middle of the road. Example frames from the complete video [4] are presented in figures 1(f) to 1(i).

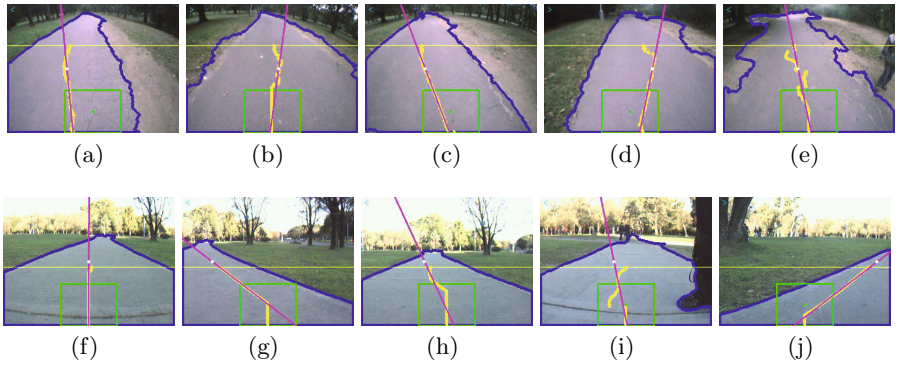

Fig. 1. Example frames from the processed images obtained during off-line (1(a)-1(e)) and on-line (1(f)-1(i)) testing. The green square represents the area defined as ground, the blue contour corresponds to the detected road, the red line is obtained by fitting the middle yellow points (which is then used for motion commands) and the horizontal yellow line is the last row that is processed for adding middle points.

5 Conclusions and Future Work

In this work we present a real-time implementation of a navigable floor detection algorithm for a mobile robot equipped only with a monocular camera.

The main computational requirement of this method is the image-segmentation step. By using an on-board low-power graphics card which is capable of general purpose computation, we achieve a reactive control for the robot. Based on an existing GPU implementation of the quick-shift segmentation algorithm, we introduce some simple optimizations which result in a speedup of approximately five times. This reduces computational times to a level that allows real-time execution on the robot.

The capability of the method to detect a road in an outdoor natural scenario was tested using a pre-recorded dataset. Also, several tests were performed on a

[4] `http://robotica.exp.dc.uba.ar/trac/exabot/export/85/trunk/src/gpu/floordetection/results-bien/frontier.avi`

mobile robot featuring the specified hardware. Besides starting the robot in the middle of the road and verifying its stability, several extreme initial positions and orientations were tested. In these conditions the robot quickly returned to the center line demonstrating the general stability of the road-detection.

After obtaining satisfying results, there are several aspects that can be improved. First, the classification of each segments as road or non-road (based on a single average measure in HSV color-space) could rely on the fact that a contiguous (i.e.: without holes) region is expected to be detected as road. Therefore, it could be better if the classification of a single segment could depend on the classification of nearby segments. Second, while framerate is already sufficient, the algorithm could possibly be sped up by attempting to utilize the GPU for other steps than image segmentation alone. Furthermore, robustness of this method in the long-term (and also to varying illumination) would have to be addressed by including some form of learning of positive (and possibly negative) examples of the area considered as free-space. Finally, the classification itself could be improved by considering not only pixel values (color) but also texture.

References

1. DeSouza, G., Kak, A.: Vision for mobile robot navigation: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(2), 237–267 (2002)
2. Bonin-Font, F., Ortiz, A., Oliver, G.: Visual navigation for mobile robots: a survey. Journal of Intelligent & Robotic Systems 53(3), 263–296 (2008)
3. Ulrich, I., Nourbakhsh, I.: Appearance-based obstacle detection with monocular color vision. In: Proceedings of the National Conference on Artificial Intelligence, pp. 866–871. AAAI Press, MIT Press, Menlo Park, Cambridge (2000)
4. Santosh, D., Achar, S., Jawahar, C.: Autonomous image-based exploration for mobile robot navigation. In: IEEE International Conference on Robotics and Automation, ICRA 2008, pp. 2717–2722. IEEE (2008)
5. Wang, Y., Fang, S., Cao, Y., Sun, H.: Image-based exploration obstacle avoidance for mobile robot. In: Control and Decision Conference, CCDC 2009, pp. 3019–3023. IEEE, Chinese (2009)
6. Vedaldi, A., Soatto, S.: Quick Shift and Kernel Methods for Mode Seeking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 705–718. Springer, Heidelberg (2008)
7. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. International Journal of Computer Vision 59(2), 167–181 (2004)
8. Fulkerson, B., Soatto, S.: Really quick shift: Image segmentation on a gpu. In: ECCV 2010 Workshop on Computer Vision on GPUs, CVGPU 2010 (2010)
9. Pedre, S., De Cristóforis, P., Caccavelli, J.: A mobile mini-robot architecture for research, education and popularization of science. Journal of Applied Computer Science Methods 2(1) (2010)