Fusing LIDAR and Vision for Autonomous Dirt Road Following

Incorporating a Visual Feature into the Tentacles Approach

Michael Manz[1], Michael Himmelsbach[1], Thorsten Luettel[1] and Hans-Joachim Wuensche[1]

University of the Bundeswehr Munich, Autonomous System Technology, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany

Abstract. In this paper we describe how visual features can be incorporated into the well known tentacles approach [1] which up to now has only used LIDAR and GPS data and was therefore limited to scenarios with significant obstacles or non-flat surfaces along roads. In addition we present a visual feature considering only color intensity which can be used to visually rate tentacles. The presented sensor fusion and color based feature were both applied with great success at the C-ELROB 2009 robotic competition.

1 Introduction

Autonomous driving on forest tracks and dirt roads with faulty GPS localization, more or less no geographical maps and complex lighting condition is still a challenging task for intelligent vehicles. In the past we used the tentacle approach, utilizing a LIDAR based occupancy grid and GPS-Waypoints to handle such difficult autonomous driving missions. But up to now the road following capabilities of the tentacles approach without using dense GPS-Points are limited to scenarios with significant obstacles or non-flat surfaces along what humans would otherwise easily recognize as roads. To overcome these limitations we extend the tentacles approach to also incorporate visual features. Compared to other work we do not fuse the sensors on the data level [2] or try to generate a fused world map [3]. Instead we rate the tentacles independently in a way adequate to each sensor. The gathered information is subsequently fused during the process of tentacle selection.

The proposed procedure for visual tentacle evaluation is inspired by the work of [4] and is able to handle weighted pixel images form various color or texture segmentation algorithms. In addition it utilizes camera gaze control as well as compensation of vehicle's pitch and roll movements in a model based manner.

In order to drive through forest and across fields we have developed a fast-to-compute visual feature based only on the saturation channel of the HSI color space and is therefore quite robust against difficult lighting conditions and shadows. In addition, the proposed feature does not rely on the knowledge of past

road locations and is therefore best suited to the reactive tentacle approach. Before we explain the visual tentacle evaluation and the proposed visual feature in more detail we briefly introduce the tentacles approach.

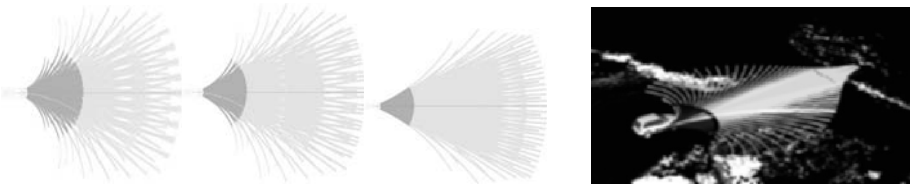
2 Driving with Tentacles

Our basic intention underlying the tentacles approach is to let our robot move within an unknown environment similarly to how a beetle would crawl around and use its antennae to avoid obstacles. Indeed, in our approach the analogue to an insects antennae are target trajectories representing the basic driving options of the vehicle given the current vehicle state. These driving options that we named "tentacles" then probe an occupancy grid (built from LIDAR data in a way similar to [5]) to determine drivability and other properties of the corresponding target trajectory. The main loop of the approach is as simple as follows: In each cycle, first select a subset of tentacles according to the vehicle state. Then, by inspecting the occupancy grid, assign properties to each tentacle and select one of the tentacles as the target trajectory for the vehicle to drive. Smooth transitions between selected tentacles can be achieved by temporal filtering or some kind of hysteresis.

2.1 Tentacle Evaluation and Selection

As said, a tentacle basically represents a target trajectory defined in the reference frame of the vehicle. These are represented as point samples p_i on arbitrarily shaped line sequences and denoted by $t_{lsq} = \{p_1, \cdots, p_N\}$. Fig. 1 shows the tentacle geometries we use at the C-ELROB 09. We will now give a brief description of the properties we evaluate for a tentacle and how a tentacle gets selected for execution based on these properties. The interested reader is referred to [1] for more details on this topic.

Drivability. The most important of all tentacle properties for sure is its drivability, $t_{drivable}$. Indeed, all other properties of a tentacle are only evaluated



(a) (b)

Fig. 1. (a) Some of the tentacles we used at C-ELROB 09. Every image shows one set of tentacles to be selected according to robot velocity. Every set contains 1000 tentacles, discretizing a subspace of curvatures, lateral offsets and headings. (b) Tentacles selected according to current vehicle state shown on top of the occupancy grid they relate to.

if the tentacle was considered drivable. For evaluating this property, we only have a look at the cells near to the skeleton line of a tentacle. If this set does not contain a cell that is within the crash distance of the vehicle and itself considered an obstacle, the tentacle is classified drivable.

Distance to Obstacles. This property is closely related to the drivability of a tentacle. However, of all drivable tentacles, we would prefer to select the one which can be driven for longer time, keeping the vehicle away from threatening obstacles. This is reflected by the property $t_{clearness}$, which is just the longitudinal length to the bin where an obstacle was detected (or infinity if there is no obstacle along the tentacle).

Flatness. Especially in offroad scenarios, a tentacle should be assigned a value reflecting the flatness of the ground it would lead the vehicle to. Assigning a flatness $t_{flatness}$ to a tentacle is a simple matter of computing the weighted sum of absolute z-coordinate differences of all cells within the tentacle support area.

With all drivable tentacles being assigned their properties, it now remains to select one of the drivable tentacles for execution. To be able to fine-tune the influence of individual properties, we decided to combine them all in a weighted sum, choosing the one tentacle minimizing this sum. As a prerequisite, we require all properties be normalized to uniform range. Again, we refer to [1] for details on how the properties are best normalized. Formally, selecting a tentacle is described by

$$t_{selected} = \underset{\{t|t_{drivable}=1\}}{argmin} \; a_1(1 - t_{clearness}) + a_2 t_{flatness} \tag{1}$$

where the a_i denote the weights allowing fine-tuning the resulting behavior.

Please refer to [1] for more details and an extensive description of how high-level knowledge based on GPS sensors and a geographical map can be incorporated into the tentacle approach. In the following of the paper, we will show how properties derived from a camera system can be incorporated via the introduction of new summands into this last equation.

3 Evaluating Tentacles by Visual Appearances

In this section we propose a possibility to incorporate the visual appearance of a tentacle into the tentacles rating process described in Sec. 2.1. The evaluation process deals with weighted pixel value images such as produced from road color (e.g. [6]) or road texture segmentation (e.g. [7]) algorithms. Furthermore it utilizes gaze control as well as vehicle pitch/roll stabilization in a model based manner. To avoid problems arising from fusing data from different times the camera device is triggered by the LIDAR sensor.

3.1 Perspective Mapping and Gaze Control

As described in Sec. 2.1 each tentacle is defined in the vehicle coordinate system and represents a possible road geometry given by its line sequence. Homogeneous

transformation matrices are used to do the perspective projection of the currently active tentacles into the camera frame coordinates. In order to compensate severe disturbances in vehicle pitch and roll angle while driving along dirt roads we consider the measurements of an on board IMU within the perspective projection equation. The IMU measurements of the vehicle's pitch and roll angle are split into two different frequency ranges. At the one hand the high frequencies are considered as disturbances due to rough terrain and directly compensated within the image with the aid of the mapping equations and the low frequencies on the other hand are interpreted as orientation changes of the road surface and therefore considered as a zero shift of the vehicle orientation.

As we want to drive along dirt roads with large horizontal curvatures we have to do gaze control. To this purpose we implemented to different algorithms. The first algorithm takes the current steering angle of the autonomous vehicle as well as a predefined lookahead distance and uses the Ackermann steering geometry and a pinhole camera model to calculate the gaze direction. The second algorithm takes the skeleton line of the last selected tentacle and also uses a predefined lookahead distance and the pinhole camera model to determine the camera platform yaw angle.

3.2 Determine Visual Quality

To determine the visual quality of a tentacle all that is needed is an image of weighted pixel values as input into the evaluation process. Weights should correspond to the probability of a pixel being a "'road"' or a "'non road"' pixel, and numerous algorithms for road color or texture segmentation can be used. In Ch. 4 we introduce such a color feature that is highly convenient for usage within the tentacle approach. A tentacle will be assigned individual properties t_{vis_i}, one by each different pixel weighting method used. Then, a tentacle's visual quality t_{visual} is the combination of all different color or texture features used, given by

$$t_{visual} = \sum_i b_i t_{vis_i} \tag{2}$$

contributing another summand to (1) when fusing LIDAR and vision for tentacle selection. Here, the b_i allow fine-tuning the influence of single visual features.

The evaluation process starts with first transforming the sample points of each tentacle's skeleton line $t_{lsq} = \{p_1, \cdots, p_N\}$ into the camera image coordinates. If 70% of the points in the line sequence are within the image boundaries, the tentacle is declared as visible. Otherwise, the tentacle is declared as non-visible and all single visual qualities t_{vis_i} are set to a fix value of 0.6. This is simply because we can not blame tentacles for getting out of sight of the camera, a point to which we return later.

The 3D support area boundaries of visible tentacles however are sampled and projected into the camera frame, as shown in Fig. 2 (a). The projected support area is split in two sub-areas on the left and on the right of the tentacle. This is done to account for grass strips in the middle of the road like they often

(a) (b)

Fig. 2. Image of a single tentacle's skeleton line (green) and support area placed between the blue lines on the right and left of the skeleton line) (a), Rating a sparse tentacle set according to the introduced visual feature (red encodes non drivable) (b)

appear on field paths. Subsequently if a tentacle is declared as visible we sum all weighted pixel within the projected support areas for a feature i (see Fig. 2) symbolised by $s_i(x_t, y_t)$ and normalize the sum by the number of considered pixels N_t.

$$w_{vis_i} = \frac{1}{N_t} \sum_{j}^{N_t} s_i(x_{tj}, y_{tj}) \tag{3}$$

To speed up the summing process we use line-oriented integral images like [4] to evaluate the weighted sum w_{vis_i} for each tentacles. In order to make the visual quality values t_{vis_i} comparable to other features, they are normalized to the range [0,...,1], where a value of 0 designates a preference for such a tentacle. The value is calculated by the sigmoid-like function

$$t_{vis_i}(w_{vis_i}) = \frac{2.0}{1.0 + e^{-c_{vis} \cdot w_{vis_i}}} - 1 \tag{4}$$

where the constant c_{vis_i} is calculated by (5) to yield $t_{vis_i}(w_{vis_i\,0.5}) = 0.5$. The value $w_{vis_i\,0.5}$ can be adjusted according to the considered visual feature:

$$c_{vis_i} = \frac{ln1/3}{-w_{vis_i\,0.5}} \tag{5}$$

4 Rating Tentacles by Color Intensity Feature

There is a vast number of works on road detection and following algorithms. Basically, all these works try to identify the road according to visual features special to roads, like road boundaries [8], texture [7] and color [6]. The visual feature we propose goes the other way around: Instead of trying to recognize a road in an image, we try to segment non-drivable pixels in an image.

In numerous video streams of non-urban scenarios, we discovered that one of the best and efficient ways to segment non-drivable terrain is to utilize the saturation channel of the HSI color space. The feature we propose makes use of

the knowledge we gained - that the road surface may show all kinds of colors but there will be only a vanishing amount of brightly colored pixels on a road's surface. In the opposite, this means that almost all brightly colored areas will most likely correspond to non-drivable area. This is especially true in dirt road and forest scenarios where the road is typically surrounded by some kind of vegetation. To further enhance this effect we transform the saturation value of each pixel $s(x, y)$ according to the dynamic weighting function,

$$s_w(x,y) = \begin{cases} 0, & s(x,y) \leq \mu_s \\ \frac{255(s(x,y)-\mu_s)}{s_{off}}, & \mu_s < s(x,y) < (\mu_s + s_{off}) \\ 255, & s(x,y) \geq (\mu_s + s_{off}) \end{cases} \quad (6)$$

where μ_s represents the temporally low-pass filtered mean saturation value in the lower image parts excluding the engine hood and s_{off} is a parameter to adapt the weighting transition. The mean value μ_s is subject to constraints to cope with special scenarios like paved areas (almost all image pixels are non colored pixels) or grassland (almost all pixels are colored). To speed up the process we use look up tables to evaluate equation (6). Fig. 3 shows the weighting of pixels according to the proposed visual feature.

One additional advantage of the feature is that, for a good segmentation result on country roads, no online learning algorithm is required. Thus the feature's performance does not depend on any earlier measurements of the road in front of the car. For example, the online learning approach based on the famous histogram back-projection algorithm used in [4] for road detection relies on the vague assumption that if the vehicle was well positioned on the road some successive steps of time before, then, at time t, exactly those pixels will belong to the road that show the same color statistics as those believed to belong to the road several successive time steps earlier. Obviously, such approaches can easily fail if the vehicle leaves the road without the algorithm taking notice of it or if the visual appearance of the road suddenly changes.

This is important because even if there are a thousand tentacles no tentacle might actually fit precisely into the road boundaries. This is due to the fact that tentacles only represent a limited set of driving primitives and will never cover all real-world road geometries. As a consequence, it is most likely that none of the pixel sets associated with any tentacle will fit the true pixel value distribution of

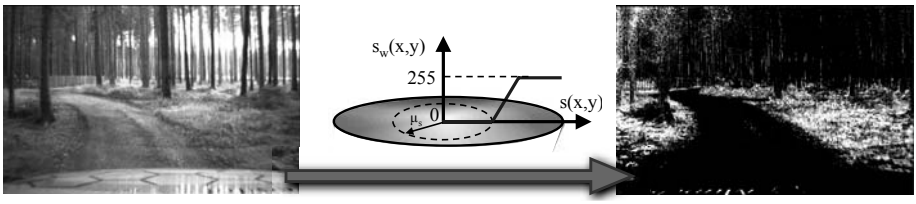


Fig. 3. RGB image of a dirt road scene (left), transformation by the weighting function for the saturation channel (middle) and resulting image (right).

the road. Therefore, knowledge about which pixels really belong to the road is even more incomplete compared to approaches based on more fine-grained road geometry hypotheses, such as the 4D approach for road tracking [8]. Hence, no serious statistics can be done.

Summarizing, compared to all approaches incorporating some kind of history, our feature for road segmentation allows the vehicle to recover from any wrong decision previously taken. Of course there may be more non-drivable terrain which is not recognized by the proposed visual cue (see upper part of Fig. 3), but in combination with a geometric road hypotheses and an obstacle avoidance system such as the tentacles approach it becomes a powerful additional feature. Evaluation of a sparse tentacle set according to the described color intensity feature and the rating process explained in Ch. 3 using $w_{vis0.5} = 70$ is shown in Fig. 2 (b). Incorporating the Intel Performance primitives libraries, we are able to evaluate as much as 1000 different tentacles within the cycle time of the system of 0.1s, corresponding to one LIDAR revolution.

5 Results and Conclusions

The proposed tentacle based LIDAR and vision fusing as well as the explained visual feature were used successfully during the C-ELROB 2009 in Oulu/Finland within the "Autonomous navigation" scenario. The scenario consisted of a round track with a length of approx. 2.6 km, which had to be passed twice, totaling 5.2 km (see Fig. 4 (b)). As most parts of the track led through a dense forest, reliable GPS information was not available. Despite the tough conditions (see also Fig. 4 (a)) the proposed enhanced tentacle approach enabled our autonomous vehicle MuCAR-3 to drive 95 % of the complete distance fully autonomously undercutting the time limit of 1 h by far with 43 min.

Problems due to fusion only arose at crossings, where both the LIDAR and the vision tentacle evaluation tend to go straight on even if the target track suggests to go left or right. With its 360 degree field of view, only the LIDAR is able to sense the right branch, whereas vision might fail sometimes. This is because of the limited field of view of the visual sensor and the fact that if there was no tentacle selected leading into the branch there will be no gaze control for visually focussing the branch. Thus, MuCAR-3 will go straight on and miss the branch even if non-visible tentacles are not rated as completely non drivable (see Sec. 3.2). Nevertheless there are by far more advantages then disadvantages using the described fusion process. An exemplary situation where tentacle navigation benefits from visual information is shown in Fig. 4 (b). Here, without using visual tentacle qualities, MuCAR-3 decided to go right in front of a narrow road surrounded by dense vegetation at both sides, and drove over the flat grassland on the right. In contrast, with the aid of the proposed tentacle evaluation process fusing LIDAR with vision, MuCAR-3 was able to stay on the road and drove through the dirt road section similarly to a human driver.

(a) (b) (c)

Fig. 4. Visual tentacle rating facing complex lighting conditions during C-ELROB 2009 (a), comparison between LIDAR based tentacle driving (red), fused LIDAR and vision based tentacle driving (blue) and a human driven trajectory (yellow)(b), aerial image of the C-ELROB track (c)(blue encodes autonomously driven parts)

6 Acknowledgements

The authors gratefully acknowledge funding by DFG excellence initiative research cluster CoTeSys, partial support of this work by DFG within the SFB TR 28 *Cognitive Automobiles*, as well as generous funding by the Federal Office of Defense Technology and Procurement (BWB).

References

1. F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Müller, and H.-J. Wünsche, "Driving with Tentacles - integral structures of sensing and motion," *International Journal of Field Robotics Research*, 2008.
2. C. Rasmussen, "Combining laser range, color, and texture cues for autonomous road following," *IEEE Inter. Conf. on Robotics and Automation*, 2002.
3. A. Broggi, S. Cattani, P. P. Porta, and P. Zani, "A laserscanner- vision fusion system implemented on the terramax autonomous vehicle," *IEEE Int. Conf. on Intelligent Robots and Systems*, 2006.
4. U. Franke, H. Loose, and C. Knöppel, "Lane Recognition on Country Roads," *Intelligent Vehicles Symposium*, pp. 99–104, 2007.
5. S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic terrain analysis for high-speed desert driving," in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.
6. C. Tan, T. Hong, T. Chang, and M. Shneier, "Color model-based real-time learning for road following," *IEEE Intelligent Transportation Systems Conference*, 2006.
7. J. Zhang and H.-H. Nagel, "Texture-based segmentation of road images," *IEEE Symposium on Intelligent Vehicles*, pp. 260–265, 1994.
8. E. D. Dickmanns and B. D. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 199–213, February 1992.