4th European Conference on Mobile Robots – ECMR'09, September 23–25, 2009, Mlini/Dubrovnik, Croatia

Time To Contact for Obstacle Avoidance

Guillem Alenyà* Amaury Nègre† James L. Crowley†
*Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas 4-6, 08028 Barcelona
†INRIA Grenoble Rhône-Alpes Research Centre, 655 Av. de l'Europe, Montbonnot, France

***Abstract*— Time to Contact (TTC) is a biologically inspired method for obstacle detection and reactive control of motion that does not require scene reconstruction or 3D depth estimation. TTC is a measure of distance expressed in time units. Our results show that TTC can be used to provide reactive obstacle avoidance for local navigation. In this paper we describe the principles of time to contact and show how time to contact can be measured from the rate of change of size of features. We show an algorithm for steering a vehicle using TTC to avoid obstacles while approaching a goal. We present the results of experiments for obstacle avoidance using TTC in static and dynamic environments.**

***Index Terms*— Time-To-Contact, obstacle avoidance, Bayesian driving.**

I. Introduction

Obstacle avoidance is commonly defined as the problem of computing a motion control sequence that is free of collisions. The design of an obstacle avoidance method is naturally conditioned by the dynamic behavior of the vehicle, whether obstacles are static or in motion, the availability of a map, and the degree to which the environment is structured. When obstacles are static and the structure of the environment is known in advance, a global approach based on planning is generally preferred. When the structure of the environment is unknown or dominated by moving obstacles, a local approach may be necessary. However, when a local approach is used, some form of global supervision is required to prevent the vehicle from being trapped.

Classical obstacle avoidance algorithms are based on heuristics that convert sensor readings to motion instructions [3, 1]. Naturally these methods can be neither exhaustively tested nor proved effective in all cases. Alternatively, analogies with physical fields have been used to derive methods for obstacle avoidance. The most popular of these is the Potential Field Method (PFM) [11] in which the vehicle is modeled as a particle under the attractive force of a target while obstacles exert repulsive forces. This approach has been explored by Borenstein through the Virtual Force Field (VFF) and Vector Field Histogram (VFH) methods, and their extensions: VFH+ and VFH* [23]. These methods compute a subset of motions and search for the best path among the possible safe paths using an intermediate representation. Other variations of this approach include the Obstacle Restriction Method, the Steering Angle Field [5] or the Dynamic Window Approach [8]. The Dynamic Window Approach uses velocity instead of motion direction, and is known to be a useful method for vehicles with at high speeds with limited acceleration. The Velocity Obstacles (VO) method [7] takes into account the velocity of the obstacles.

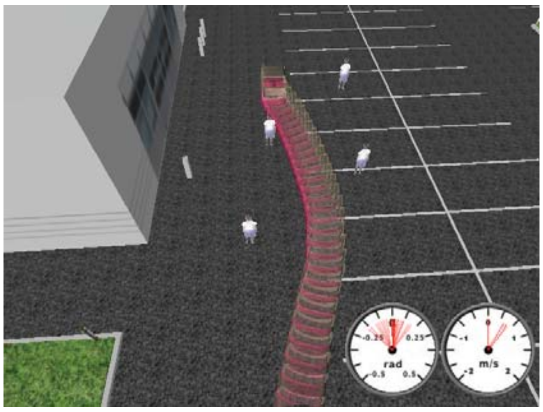


Fig. 1. Path of a vehicle avoiding obstacles automatically detected using monocular TTC estimations in a simulated environment.

Classically, obstacle avoidance strategies combine distance information with information about vehicle motion. Range sensors using time of flight of ultrasound or lasers are often used to provide distance information. Ultrasonic range sensors provide distance to the nearest reflecting surface within a relatively large field of view. A classical arrangement is to place several sensors in a ring such that the overlap of their fields of view provides a zone protecting the vehicle in the direction of travel. However with ultrasonic range sensors, if more than one sensor is operated at the same time cross talk problems can appear. As a result, the frequency of obstacle detection is limited by the number of sensors in use, and the time required for an echo to return from an obstacle. In addition, oblique surfaces, corners, and temperature variations can create artifacts. Infrared range sensors provide range measurements with a much faster measurement time, but a much smaller field of view. As a result, a much larger number of sensors are required to obtain full coverage of the vehicle path, but obstacle detection can be performed at higher temporal rates.

As laser range finder technology has matured, scanning laser range finders have become very popular for both vehicle navigation and obstacle avoidance. A laser range sensor usually transmits a single beam through a mirror that is rotated to obtain a complete line scan. Usually this sensor is placed parallel to the ground in the sense of direction of the robot. Obviously, lower obstacles or lumps not in the plane defined by the laser cannot be detected and thus obstacle avoidance can fail. To solve this the laser can be equipped with a tilt unit, at the cost of increasing the time required to scan the environment in an additional dimension. False range measurements can result when the laser beam reflects from more than one surface. Sensor readings may be erroneous because of specular reflections. Because of potential problems with eye-safety, many laser sensors can only be used in areas where no humans are present.

Time-of-Flight (TOF) cameras have recently emerged as a suitable alternative to scanning laser range sensors [4]. ToF cameras can typically provide 3D depth images at 25 fps. However, some calibration problems and sunlight artifacts are still present, and they have generally low resolution and can be plagued by depth ambiguity.

Computer vision can potentially overcome many of the problems of other obstacle sensors. However, obtaining reliable 3D depth information from 2D planar images can be computationally expensive and unreliable. Stereo vision remains the most common technique for 3D computer vision. Stereo vision uses the relative positions of image features in two or more images taken from different positions to geometrically estimate the 3D point position of the features. In practice, there are a number of open research issues related to computer vision. Open problems include the relative suitability of point features vs line features and region features, as well as the difficult and tedious methods required for camera calibration. In addition, substantial computing power can be required for stereo matching, particularly if high frame rates are required.

Other visual sources of range information include depth-from-focus depth-from-zoom, depth from image blur, structure from motion algorithm, and active triangulation methods that use projections of structured light. So far, none of these approaches have proven useful for obstacle detection.

In this paper we present a vision based approach for computing distance information from a moving monocular camera system. Distance is expressed as Time-To-Contact (TTC). Time-to-Contact can be defined as the time that an observer will take to make contact with a surface under unknown constant relative velocity. TTC can be estimated as the distance between two image points divided by the rate of change in that distance. The result is a form of relative distance to the object in temporal units that does not require camera calibration, 3D reconstruction or depth estimation. As such, TTC can potentially provide the basis for fast visual reflexes for obstacle avoidance and local navigation.

It is well known that the egomotion of a robot and its relative position with respect to the obstacle cannot be estimated with a single uncalibrated camera. Part of the attraction of TTC is that the calculation relies only on image measurements and does not require camera calibration or knowledge of the structure of the environment or the size of shape obstacles. Moreover, TTC naturally encodes the dynamics of the motion of the observer. As a consequence, TTC can be used to construct motion reflexes for collision avoidance, provided that a fast, reliable measure can be made of distance in the image. As we will see, this is valid also for dynamic environments.

The remainder of this article is structured as follows. In Section II the TTC principles are presented, and in Section III the feature detection and scale change estimation algorithm is depicted. Section IV describes briefly the driving algorithm we have implemented. Experiments are presented in Section V, including different practical experiences devoted to validate TTC in the context of obstacle avoidance. Finally, conclusions are presented in Section VI.

II. Time-To-Contact

The time to contact is usually expressed in terms of the speed and the distance of the considered obstacle. The classical equation to compute the TTC is

$$\tau = -\frac{Z}{\frac{dZ}{dt}}\,, \tag{1}$$

where Z is the distance between the camera and the obstacle, and $\frac{dZ}{dt}$ is the velocity of the camera with respect to the obstacle. However, with a monocular camera only, the distance is generally unknown. It's possible to derive (1) by using a characteristic size of the obstacle in the image [19] and by using the approximation that the obstacle is planar and parallel to the image plane :

$$\tau = \frac{\sigma}{\frac{d\sigma}{dt}}\,, \tag{2}$$

where σ is the size (or the scale) of the object in the image and $\frac{d\sigma}{dt}$ the time derivative of this scale. This equation is more appropriate as the size can be obtained directly in the image space. This reformulates the problem as a problem of estimating the obstacle size, as well as the rate of change of size.

Note that the TTC does not rely on the absolute size of the object in the image sequence, but in the relative change in scale from one frame to another. As a consequence, the TTC computation is not dependent on camera optics or the object size, only is dependent on the depth distance and the camera velocity.

Providing a fast, reliable distance measurement for TTC is a challenging task. Classical methods to compute TTC rely on the estimation of optical flow and its first derivative [16, 17]. However, optical flow methods are iterative and tend to be computationally expensive and relatively imprecise. Calculating the derivative of optical flow to estimate TTC further amplifies noise, generally leading to an unstable and unreliable estimate of TTC. Most demonstrations of this approach tend to use highly textured objects in order to obtain a dense velocity fields [22]. Such textured objects may provide an useful laboratory demonstration, but are not generally representative of the objects observed in real world scenes.

The use of the temporal derivative of the area of a closed active contour [21] has been proposed to avoid the problems associated with the computation of image velocity fields and their derivatives. This is an additional step in the tracking of active contours that can be avoided using the parameters of the deformation of the active contour [15]. Active contour initialization is usually performed manually, and is thus difficult to implement in real moving robots.

To overcome the problem of background segmentation a method has been proposed [10] based on derivatives of the whole image brightness. TTC obtained with this method is only valid when a large fraction of the image corresponds to the obstacle.

Some of these approaches restrict viewing conditions or allowed motions. When affine camera models are assumed [6, 21, 22], then affine image conditions are required Camera motion is sometimes restricted to planar motion [22, 13] or to not include vertical displacements [6] or cyclotorsion [13] An alternative approach is to compute TTC from scaled depth [16, 15]; however this adds complexity and has been shown that introduces new constraints and additional errors when these constrains are not fully satisfied [2].

In the next Section we present Scale Invariant Ridge Segments (SIRS) algorithm. Its interest consists in that several objects in the scene can be tracked at real time, obtaining at the same time several estimators of distance to potential obstacles. Compared to the other methods, obstacle tracking is automatically initialized and thus doesn't require human intervention.

III. Scale Invariant Ridge Segments (SIRS)

A characteristic size for an obstacle can be estimated from the characteristic scale using a normalized Laplacian scale space. Characteristic scale is estimated by computing the Laplacian (or second derivative) of the image for a given pixel over a range of scales. The scale at which the Laplacian is maximized is the "characteristic scale" for that pixel. A characteristic scale can be estimated at all image points except discontinuous boundaries, where the Laplacian is zero and thus has no maximum, and will return the same value for all orientations. A change of scale in the image results in a similar change in the characteristic scale.

A similar measure can be estimated using the Hessian of the gradient. As with the Laplacian, the Hessian can be computed for a given pixel over a range of scales. The scale at which the Hessian returns a maximal value is an invariant for changes of scale and rotation.

Because the characteristic scale at each pixel varies equally with changes in image scale, characteristic scale computed from the Laplacian or the Hessian can be used to estimate TTC at (nearly) all pixels in image image. However, estimating TTC from characteristic scale requires registering the images so that the rate of change in scale is measured for the same image feature.

Image registration is generally estimated by some form of tracking. A popular approach is to track interest points such as the maxima of the Laplacian, as used in the SIFT [14] detector or the maxima of the Hessian, as provided by the Harris [9] interest point detector. Unfortunately the Harris detector tends to respond edge and corner points where size is not meaningful. The SIFT detector detects scale-space maxima of the Laplacian and provides a stable estimate of scale. However, the position of SIFT interest points tends to become unstable along elongated shapes, as are common in many navigation scenes. The Scale Invariant Ridge Segment (SIRS) detector [20] extends the maximum of the Laplacian as used in the SIFT detector to detect elongated shapes.

The SIRS detector consists in maximizing a score function in the 3D segment's space. We consider a ridge segment S parameterized by two vectors:

- $\vec{c} = (c_x, c_y, c_\sigma)$: center position in the image scale-space
- $\vec{s} = (s_x, s_y, 0) = \|\vec{s}\| \cdot \vec{u}$: half-edge (vector between an extremity and the center)

Then the score function correspond to the sum of the normalized Laplacian $\nabla^2 L$ combined with a symmetry detector

$$f(S) = \int_{l=-||\vec{r}||}^{||\vec{r}||} |\nabla^2 L(\vec{c} + l \cdot \vec{u})| - |\nabla^2 L(\vec{c} + l \cdot \vec{u}) - \nabla^2 L(\vec{c} - l \cdot \vec{u})| \, dl - \alpha \cdot ||\vec{r}|| \quad (3)$$

where α is a parameter that represent the minimum Laplacian value needed to detect a segment.

To speed-up the maxima search, we can compute a principal direction for any center position by using the eigen vectors of the Hessian Matrix

$$\mathcal{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial xy} \\ \frac{\partial^2 f}{\partial xy} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}.$$

Then, the score function is performed in a 4 dimensional space. A second reduction is also performed by eliminating segments where Laplacian value in the center is too low.

The algorithm is depicted in Alg. 1. Its result is illustrated on Fig. 2. We can see that the detected segments fit well most of visible elements and the segment's scale depends on the structure size.

For the registration, the detected segments can be tracked in the Scale-Space, so the scale automatically estimated at any time [20].

```
1: Computation of first and second normalized derivatives Scale-Space
2: Elimination of edge pixel using the ratio of Laplacian and Gradient values
3: for each pixel do
4:    Estimation the principal direction using the Hessian matrix
5:    Calculation of the score function and the length that maximize this function
6: end for
7: Search of local maxima
```

Algorithm 1: SIRS detector

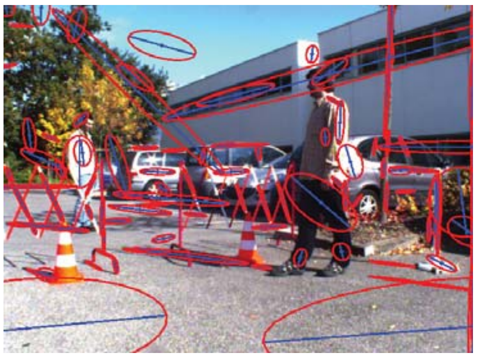
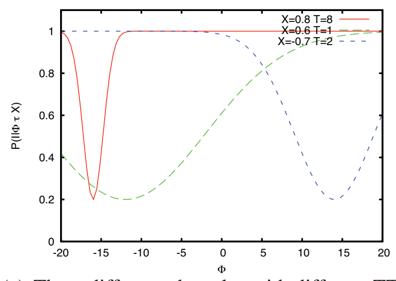
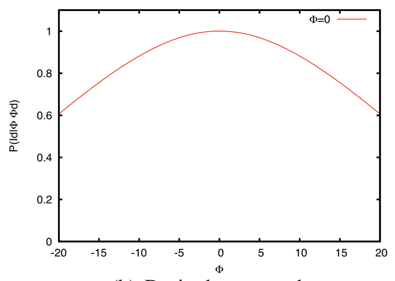


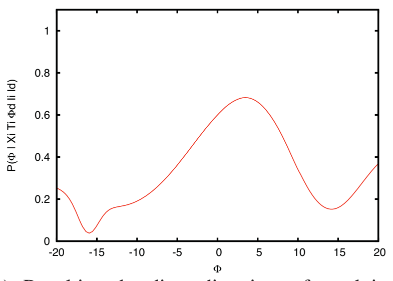
Fig. 2. An example of Scale Invariant Ridge Segments detection. Each detected segment is represented by an ellipse where the main axis represents the position of the segment and the second axis represents the scale.



(a) Three different obstacles with different TTC values



(b) Desired command



(c) Resulting heading direction of applying driving algorithm

Fig. 3. Example of the Gaussian Driving algorithm involving the computation of the heading direction in the presence of three different obstacles with different TTC values.

IV. Bayesian Driving

The control of the robot is done by using a Bayesian method described in [12]. This method consists in evaluate a probability distribution of the robot command functions of the sensor observations and the desired command.

Here we define the command as the braking angle, and the observation corresponds to the tracked ridge segments. Each observation i is characterized by the horizontal position X_i in the camera frame and its TTC T_i. The desired command (desired braking angle) is represented by the variable Φ_d.

A. Obstacle avoidance

The probability distribution to avoid obstacle is build by a proscriptive approach. The idea is to define a distribution which "avoids" some command that would move the robot closer to the obstacle. The probability of the command has to be close to 1 everywhere except in the direction of obstacles. For each obstacle i, we define a diagnostic variable I_i expressing the compatibility (in terms of security) between Φ and (X_i, T_i). The problem is now to express the distribution $P(I_i = 1|\Phi X_i T_i)$. This function will fit an inverse Gaussian function centered on the horizontal position of the target X_i and with a variance inversely proportional to the TTC T_i. This distribution can be explained by several reasons:

- Only a neighborhood of dangerous commands have a low probability, the rest have probability close to 1.
- The Gaussian's mean is chosen so that an obstacle on the right side prevents the robot to move in this direction and inversely. An obstacle in front of the robot lets the possibility to move on both direction.
- The lowest the TTC is, the most dangerous the obstacle is, so the Gaussian variance have to be larger to increase the number of dangerous commands.

Finally, the probability distribution function related to each obstacle have the following form :

$$P(I_i = 1|\Phi X_i \tau_i) = 1 - \alpha e^{-\frac{1}{2}(\Phi - \alpha_x X_i)^2 (\alpha_t T_i)^2} \tag{4}$$

where α, α_x and α_t are three parameter to defined. α represents the detection confidence while α_x and α_t depends on the camera field of view and the robot linear speed.

B. Desired command following

In order to move in accordance with the desire command, we add a new diagnostic variable I_d expressing the validity of the command knowing the desired command. The probability distribution $P(I_d = 1|\Phi\Phi_d)$ is then :

$$P(I_d = 1|\Phi\Phi_d) = e^{-\frac{1}{2}\left(\frac{\Phi - \Phi_d}{\sigma_\Phi}\right)^2} \tag{5}$$

where σ_Φ represents the importance of the desired command. The lowest σ_Φ, the most we follow the desired command but the less we will tend to avoid obstacles.

C. Fusion

To compute the final command distribution, it is necessary to merge the two models, the resulting distribution is then a product of the previous distributions :

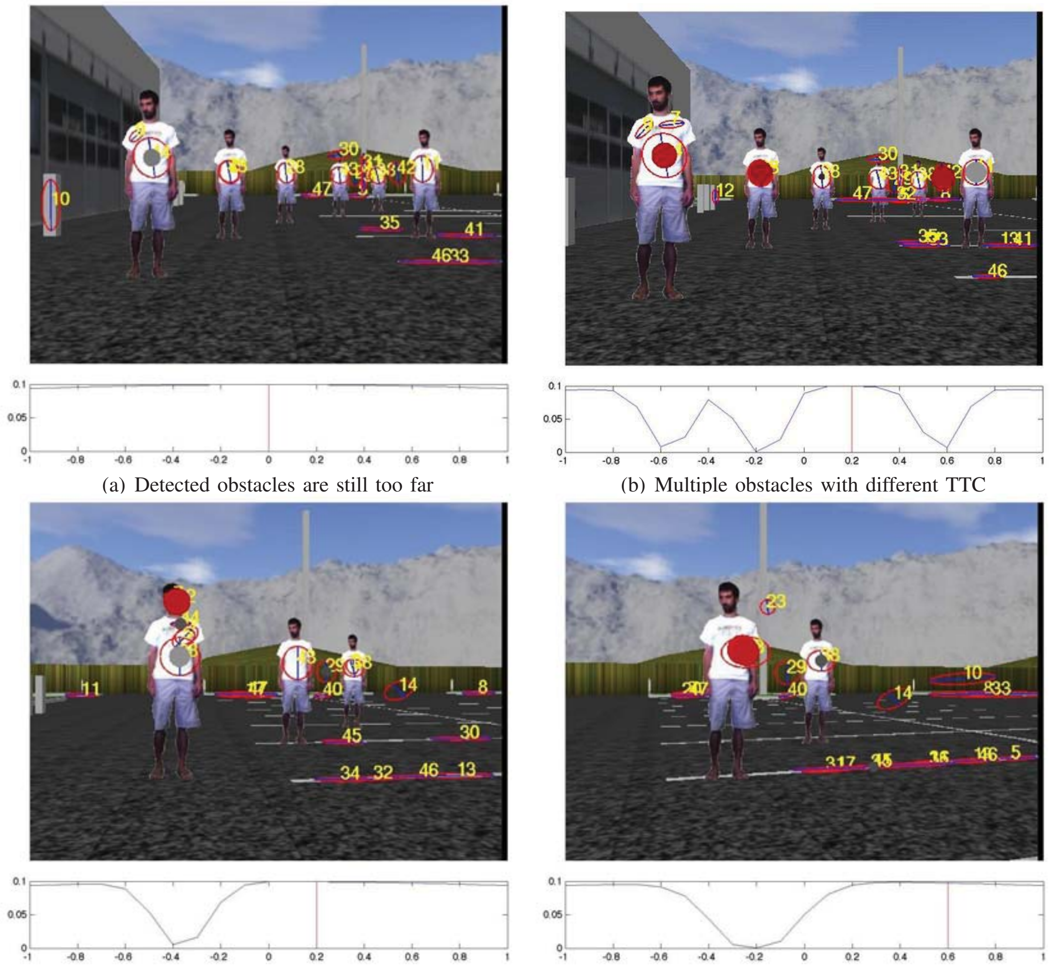
$$\begin{aligned} &P(\Phi|\Phi_d X_1..X_k T_1..T_k [I_i = 1]_{i=1..k} [I_d = 1]) \\ &\quad \propto P([I_d = 1]|\Phi\Phi_d) \prod_{i=1..k} P(I_i = 1|\Phi X_i T_i) \end{aligned} \tag{6}$$

To conclude, for each camera image, the method consists in evaluating this probability distribution and choosing the command Φ that maximizes this distribution. An example is shown in the Figure 3. We can see that with two obstacles in the right and one in the left, the vehicle will turn left to avoid the obstacle with the lowest TTC, but without braking two much to avoid the left obstacle.

V. Experiments

We present some of the results obtained in a simulated environment. In such environment we can control accurately the motion of the vehicle and thus obtain the ground truth of the motion of the robot and the obstacles, and the corresponding TTC values. Concerning our implementation, we can reach a framerate of 20 fps [18] comprising the unsupervised detection, tracking and TTC extraction of multiple potential obstacles in each image.

For the experiments presented here we have not taken into account the vehicle characteristics, like shape, kinematics or dynamics. Also, we have not imposed any environmental restrictions, like safety distances. Note that following our definition of the problem (Sec. I) we consider that these problems are associated with the integration of obstacle avoidance with a more elaborate planning strategy.



(a) Detected obstacles are still too far

(b) Multiple obstacles with different TTC

(c) Large TTC - slight right turn demanded

(d) Small TTC - large right turn demanded

Fig. 4. Some frames of one of the performed experiments. Automatically detected SIRS are marked as ellipsis with numbers and considered safe obstacles, while a grey circle appears when they are closer and a red circle indicates a potential danger. Bayesian Driving algorithm results are shown below each frame.

The task that the vehicle has to accomplish here is go ahead some meters in a straight line avoiding obstacles. Robot velocity is fixed to 1m/s and heading direction is controlled by the presented driving algorithm. Fig. 4 shows images of one of the experiments, in junction with a graphical representation of the final command distribution computed with the Bayesian Driving algorithm. Some of the automatically initialized SIRS trackers are shown in the figures as ellipsis with a number. When the observed TTC is below a first safety threshold it is indicated with a grey circle that grows while TTC decrease. When TTC is below a second threshold (here 10 seconds) the circle turns red and then is used in the Bayesian Driving algorithm. The demanded direction is indicated as a red vertical line below each figure.

Observe that in Fig. 4(a) some potential obstacles have been encountered, but the obtained TTC is still in a safe range and accordingly no red circles appear. In the next figure (Fig. 4(b)) the robot is closer to the obstacles and then some of the detectors are returning a small TTC. Observe that closer obstacles return a small TTC but as was expected far obstacles with larger TTC values are still considered as safe. Below this figure the Bayesian Driving output is depicted including 3 TTC estimations in the computation of the heading direction.

Figs. 4(c) and 4(d) show two different frames with a potentially dangerous obstacle. Observe the effect of using TTC for weight the probability distribution that represents each obstacle. While in Fig. 4(c) obstacle is relatively far away and the variance of the Gaussian is small, in Fig. 4(d) the obstacle is closer, so TTC is small and accordingly the variance of the Gaussian representing the obstacle is larger.

The result of two other experiments can be seen in Figs 1 and 5 as a global view of the scene with traces of the different vehicle positions. We can observe that the distance between the vehicle and some of the obstacles is quite short. This is because the size of the vehicle and the field of view of the camera are not taken into account here into the driving strategy. Clearly, due to the size of the field of view, when the vehicle is approaching an obstacle it can get out the field of view before the vehicle surpass the obstacle position. It is easy to include this into the driving algorithm. In one case we can provide the size of the vehicle. On the other case, a *memory* factor can be included in the algorithm to remember the position of previous obstacles and include past readings into the next heading decision. Clearly, the amount of *memory* should be initialized using the TTC evaluated when the obstacle was visible.

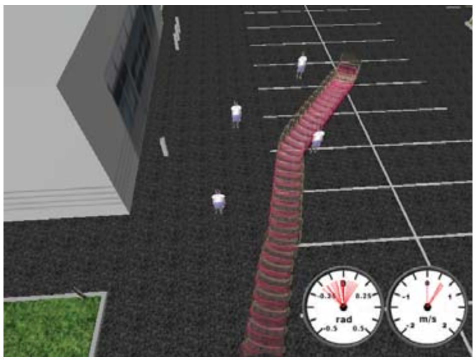


Fig. 5. Pursued path of a vehicle while avoiding obstacles automatically detected in a simulated environment.

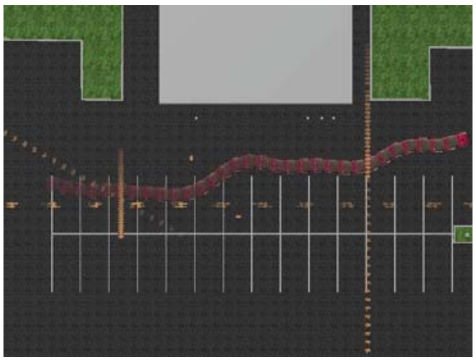


Fig. 6. Pursued path of a vehicle while avoiding obstacles automatically detected in a simulated environment.

We have also tested our algorithm in dynamic environments using moving obstacles (Fig. 6). As was expected the proposed TTC computation is able also to perform in this kind of scenarios. This is primarily because we don't need to reconstruct the environment, and also because of TTC encodes naturally the relative motion between the robot and each one of the different obstacles, even if they are also moving. Some of the planning algorithms that take into account moving obstacles consider that the obstacle position and velocity is either known or measurable [7]. We have shown here that this can be obtained using SIRS to obtain the TTC.

VI. Conclusions

In this article we have proposed Time-To-Contact as a valuable distance measure for obstacle avoidance applications. Contrary to other vision based techniques, TTC calculation relies only on image measurements and does not require camera calibration, knowledge about the robot velocity, knowledge of the structure of the environment or the size of shape obstacles. We have shown that TTC distance measure, expressed in seconds, can be effectively used in a reactive obstacle avoidance approach. For this purpose we have used a simple Bayesian driving algorithm.

We have proposed SIRS to estimate TTC. The interest above other common techniques to compute TTC is that in one hand initialization can be performed automatically, without human intervention, and in the other hand multiple potential obstacles can be tracked at the same time at real time. Some practical issues remain still open. False TTC readings appear sometimes in the floor, principally road signs. However, there are multiple floor detection algorithms and these false readings can be easily filtered.

We have also seen that our approach performs correctly in dynamic environments, so it can be used as the measure step in more elaborated planning with obstacle avoidance algorithms, such as Velocity Obstacles.

Acknowledgment

G. Alenyà was supported by the CSIC under a Jae-Doc Fellowship. This work was partially supported by Generalitat de Catalunya through a BE grant and by the Spanish Ministry of Science and Innovation under project DPI2008-06022.

References

[1] R. Chattergy. Some heuristics for the navigation of a robot. *Int. J. Robot. Res.*, 4(1):59–66, 1985.

[2] C. Colombo and A. Del Bimbo. Generalized bounds for time to collision from first-order image motion. In *Proc. IEEE Int. Conf. Comput. Vision*, pages 220–226, Corfu, Sep. 1999.

[3] J.L. Crowley. Navigation for an intelligent mobile robot. *IEEE J. Robot. Automat.*, 1(1):31–41, 1985.

[4] B. Dellen, G. Alenyà, S. Foix, and C. Torras. 3d object reconstruction from swissranger sensors data using a spring-mass model. In *International Conference on Computer Vision Theory and Applications*, 2009.

[5] W. Feiten, R. Bauer, and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. In *Proc. IEEE Int. Conf. Robot. Automat.*, page 2412?2417, 1994.

[6] F.G.Meyer. Time-to-collision from first-order models of the motion field. *IEEE Trans. Robot. Automat.*, 10(6):792–798, 1994.

[7] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robot. Res.*, 17:760–772, 1998.

[8] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robot. Automat. Mag.*, 4(1):23–33, Mar 1997.

[9] C. G. Harris and M. Stephens. A combined corner edge detector. In *Proc. Alvey Vision Conf.*, pages 189–192, Manchester, Aug. 1988.

[10] B.K.P. Horn, F. Yajun, and I. Masaki. Time to contact relative to a planar surface. In *Proc. Int. Vehicles Sym.*, pages 68–74, 2007.

[11] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.

[12] C. Koike, C. Pradalier, P. Bessiere, and E. Mazer. Proscriptive bayesian programming application for collision avoidance. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 394–399, Oct. 2003.

[13] M.I.A. Lourakis and S.C. Orphanoudakis. Using planar parallax to estimate the time-to-contact. In *Proc. 13th IEEE Conf. Comput. Vision Pattern Recog.*, volume 2, pages 640–645, Fort Collins, Jun. 1999.

[14] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[15] E. Martinez. *Recovery of 3D structure and motion from the deformation of an active contour in a sequence of monocular images*. PhD thesis, Universitat Ramon Llull, 2000.

[16] M.Tistarelli and G.Sandini. On the advantadge of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(4):401–411, 1993.

[17] N.Ancona and T.Poggio. Optical flow from 1d correlation: Aplication to a simple time-to-crash detector. *Int. J. Comput. Vision*, 14(2):131–146, 1995.

[18] A. Negre. *Evitement d'obstacles par invariants visuels*. PhD thesis, Institut National Polytechnique de Grenoble, 2009.

[19] A. Nègre, C. Braillon, J. L. Crowley, and C. Laugier. Real-time time-to-collision from variation of intrinsic scale. In *Proc. Int. Symp. Experimental Robotics*, pages 75–84, Rio de Janeiro, Jul. 2006.

[20] A. Nègre, J. L. Crowley, and C. Laugier. Scale invariant detection and tracking of elongated structures. In *Proc. Int. Symp. Experimental Robotics*, Athens, Jul. 2008.

[21] R.Cipolla and A.Blake. Surface orientation and time to contact from divergence and deformation. In *Proc. 4th European Conf. Comput. Vision*, pages 187–202, 1992.

[22] J. Santos-Victor and G. Sandini. Visual behaviors for docking. *Comput. Vis. Image Und.*, 67(3):223–238, 1997.

[23] I. Ulrich and J. Borenstein. Vfh*: local obstacle avoidance with look-ahead verification. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2505–2511 vol.3, San Francisco, Apr. 2000.