Time-to-Contact Control for Safety and Reliability of Self-driving Cars

Liang Wang
CSAI Lab. MIT
Cambridge, MA 02139, USA
wangliang@csial.mit.edu

Berthold K.P. Horn
dept. EECS, MIT
Cambridge, MA 02139, USA
bkph@csial.mit.edu

***Abstract*—For safety and reliability of self-driving cars, their control system must use the relative velocity of vehicles. The relative velocity can be estimated by differentiating the measured distance between cars with respect to time. However, inevitable measurement noise in the distance measurements will be amplified by the derivative operation. In this paper, we provide, as an alternative, a time-to-contact based approach to estimating the relative velocity, which can also be fused with measurements from other sensors, if desired. The system is implemented on Android smart phones running in real-time.**

***Index Terms*—Self-driving cars, car-following control, time to contact, machine vision, android smartphones.**

I. Introduction

With the rapid development of sensors and wireless communications, driver assistance systems and self-driving cars come closer and closer to reality. Such systems can take into account more information about the environment of the car than a human driver can, and thus show promise of yielding attractive improvements relative to todays' traffic situation [1], [2], [3]. One problem that remains of interest is how to control a car based on the information from the sensors. Another interesting question is how much self-driving cars can improve the traffic situation, e.g. reduce the number of collisions, stabilize the traffic flow, and increase the throughput of a highway. In order to control a car automatically, the distance between the controlled car and the leading car needs to be available to the control system. However, this information is *not* enough to drive a car safely and stably. Human drivers also use information about the relative velocity of the leading car with respect to their car. We have shown that the relative-velocity based control term is *important* for stability [1], [3]. Thus, how to obtain stable and reliable measurements of relative velocity is an important issue for self-driving cars.

Relative velocity can be easily estimated by taking the time derivative of the measured distance to the leading car. However, any small noise in the distance measurement will be amplified by taking the time derivative. Thus, some other, more reliable, approach to measuring relative velocity is needed. Then, if multiple sources of information are available, their outputs can be fused using e.g. a Kalman filter, in order to obtain an even more accurate and reliable estimate.

In this paper, we provide a machine vision based approach to estimating the relative velocity. As is well known, at least two calibrated cameras are needed to obtain both depth and motion information about objects in the scene (using binocular stereo [4]). Finding distance and velocity monocularly is difficult. However, the *ratio* of distance to velocity — which is known as the **time to contact** (TTC) [5] — can be obtained relatively simply monocularly. The relative speed of the vehicles can then be calculated directly by multiplication of the distance measurement with the inverse of the TTC (Note that the natural output of the TTC algorithm below *is* the inverse of the TTC). If desired, the output of this new method for estimating the relative velocity can then be fused with the estimate obtained by taking the derivative of the distance. We implement the TTC algorithm (running in real time) on Android smartphones. The whole system has been tested on several smart phones (e.g. Samsung Note 3, LG Nexus 4, Motorola Nexus 6 and Huawei Nexus 6P).

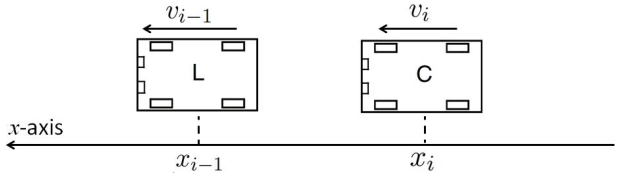


Fig. 1. Illustration of the car-following control. The blocks with "L" and "C" denote the leading car and current car.

II. Stability condition of car-following control

In this section, we show why relative velocity information is important for controlling self-driving cars. Suppose a line of cars run on a road (See Fig. 1). The acceleration command a_n of car n is based *only* on its local measurements, i.e.

$$a_n = k_d(x_{n-1} - x_n - l - v_n T) + k_v(v_{n-1} - v_n) \quad (1)$$

where x_n and v_n denote the position and speed of car n, l denotes the car length, T is known as the **reaction time**, $k_d \geq 0$ and $k_v \geq 0$ are known as the **proportional gain** and **derivative gain**, respectively [1]. Note that $a_n = \ddot{x}_n$ and $v_n = \dot{x}_n$. Thus, eq. (1) leads to the following ODE

$$\ddot{x}_n + (k_v + T)\dot{x}_n + k_d(x_n + l) = k_v\dot{x}_{n-1} + k_d x_{n-1} \quad (2)$$

Taking **Laplace transform** of both sides in eq. (2), we find:

$$H(s) = \frac{X_n}{X_{n-1}} = \frac{k_v s + k_d}{s^2 + (k_v + T)s + k_d} \quad (3)$$

For stability, we must have $|H(j\omega)| \leq 1$ for sinusoidal excitation of *any* frequency ω (where $s = j\omega$). Note that

$$|H(j\omega)|^2 = \frac{k_d^2 + k_v^2\omega^2}{(k_d - \omega^2)^2 + (k_v + T)^2\omega^2} \tag{4}$$

The stability condition $|H(j\omega)| \leq 1$ corresponds then to $k_d^2 + k_v^2\omega^2 \leq (k_d - \omega^2)^2 + (k_v + T)^2\omega^2$ for all $\omega^2 < \infty$. That is,

$$k_d T^2 + 2k_v T \geq 2 \tag{5}$$

For human drivers, the "reaction time" T is usually taken to be about 1 second [1]. Without relative velocity control (i.e. $k_v = 0$), k_d will be too large to implement (and would lead to accelerations and decelerations quite unacceptable to passengers). The only way of setting reasonable gains, i.e. considerably less than one, is to use non-zero k_v. Further, for self-driving cars, T could be much smaller, e.g. 0.5 sec. In that case, the coefficient for k_v (i.e. $2T$), is much larger than the coefficient for k_d (i.e. T^2), so that stable operation can be achieved more esasily using a large gain on the relative velocity term. In summary, the relative-velocity control term is important for self-driving cars.

The distance $d_n = x_{n-1} - x_n - l$ can be measured directly using e.g Radar or Lidar. However, if the relative velocity $r_n = v_{n-1} - v_n$ is estimated by taking the derivative of d_n directly, i.e. $r_n = d/dt(d_n)$, then the noise in d_n will be amplified. If, for example, we model small components of perturbations in the measurements as waves of the form $\epsilon \sin(\omega t)$, then the derivative will be corrupted by $\omega\epsilon \cos(\omega t)$, and so higher frequency components of measurement noise will be amplified a lot[1].We conclude that it is non-trivial to obtain a reliable measurement of relative velocity for use in control of self-driving cars.

III. Time to Contact

As an object approaches you, its image on your retina will expand; conversely, as it moves further away, the image will become smaller and smaller. This observation gives us some intuition into how one might estimate the motion of an object from its time-varying image. As is well known, at least two calibrated cameras are needed to obtain the depth of objects in the scene. Thus, the absolute motion of the object, i.e. the speed, can not be estimated using just a single camera. However, the *ratio* of the object's depth and its speed, which is known as time to contact (TTC), *can* be estimated using a single camera [5]. Note that the distance d_n can be obtained reliably using some other sensor like Radar or Lidar. Thus, r_n can be estimated well by $d_n C$, where C is the 1/TTC.

A. Passive navigation

We chose to use the camera coordinate system shown in Fig. 2. The origin is at the pin-hole of the camera, and the Z axis is along the optical axis of the camera. The X and Y directions in 3-D are aligned with the x and y axes of the image sensor [4]. Suppose there is a planar surface perpendicular to the Z axis, and moving at the speed (U, V, W). The position of the image (x, y) of the point (X, Y, Z) is determined by the **perspective projection** equation [4], i.e.

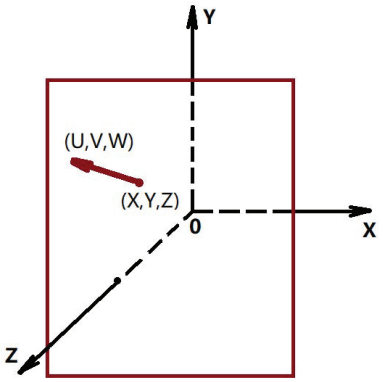


Fig. 2. We use the camera coordinate system, in which the Z axis is along the optical axis of the camera. A planner surface is perpendicular to the Z axis, and moving at the speed (U,V,W). The TTC is Z/(-W).

$$x = fX/Z \qquad \text{and} \qquad y = fY/Z \tag{6}$$

The motion (U, V, W) and the depth Z will generate movements of the image pattern, which is called the **optical flow** (u, v) [6]. The relationship between motion in the world and motion of corresponding image points is given by [7]:

$$u = (Uf - xW)/Z \qquad \text{and} \qquad v = (Vf - yW)/Z \tag{7}$$

The problem of estimating motion (U, V, W) and depth $Z(x, y)$ from (u, v) is known as **passive navigation** in the machine vision field [7], [8], [9]. Even for the very simple case shown in Fig. 2, there is *no* unique solution [4], [7].

Note that the **time to contact** (TTC) is $Z/(-W)$. Also, the **focus of expansion** (FOE) [10], denoted by (x_0, y_0), is $x_0 = fU/W$ and $y_0 = fV/W$. Eq. (7) can be written as:

$$u = (x - x_0)C \qquad \text{and} \qquad v = (y - y_0)C \tag{8}$$

where $C = 1/\text{TTC}$. The three parameters (x_0, y_0) and C can be estimated from the given optical flow (u, v).

B. Optical flow

The problem of estimating (u, v) from the changes in the image pattern — which is described by spatial variation (E_x, E_y) and temporal variation E_t of an image sequence $E(x, y, t)$ — is known as the **optical flow** problem [6]. Here, we face a very special case in which (u, v) is determined by only 3 parameters. Thus, it is more efficient to solve this problem using the **optical flow constraint** directly, i.e.

$$uE_x + vE_y + E_t = 0 \tag{9}$$

Substituting (8), we find

$$E_x A + E_y B + GC + E_t = 0 \tag{10}$$

where $G = xE_x + yE_y$, $A = -x_0 C$ and $B = -y_0 C$. Note that here E_x, E_y, E_t and G are calculated from the image sequence, while the parameters A, B, C are the unknowns to be determined.

[1]We could attempt to limit this effect by approximate low pass filtering the result, but that would introduce latency or time delays.

C. Least-square solution

We can solve for A, B and C in (10) by a **least squares** method. That is, we minimize the following objective function

$$\min_{A,B,C} \iint \left(E_x A + E_y B + GC + E_t\right)^2 \, dx\, dy \tag{11}$$

That coincides with solving a 3×3 linear system [5]:

$$\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & g \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{12}$$

with $a = \iint E_x^2 dxdy$, $b = \iint E_x E_y dxdy$, $c = \iint E_x G dxdy$, $d = \iint E_y^2 dxdy$, $e = \iint E_y G dxdy$, $g = \iint G^2 dxdy$ $p = \iint E_x E_t dxdy$, $q = \iint E_y E_t dxdy$ and $r = \iint G E_t dxdy$. Note that only multiplication and accumulation operations are involved in generating the nine quantities that appear in eq. (12), and that we can solve for (A, B, C) analytically. Thus, not surprisingly, this least-square approach is much faster than e.g. some method based on feature-detection and matching. If desired, the FOE can be calculated using$(x_0, y_0) = -(A/C, B/C)$. The relative velocity is $r_n = d_n C$. The remaining work is to implement the algorithm outlined above.

IV. Android-system oriented implementation

We implement the TTC algorithm (in JAVA) on Android smartphones. We describe the implementation in detail below. Moreover, In order to stabilize the estimation of TTC and FOE, a recursive filter is also used.

A. System specification

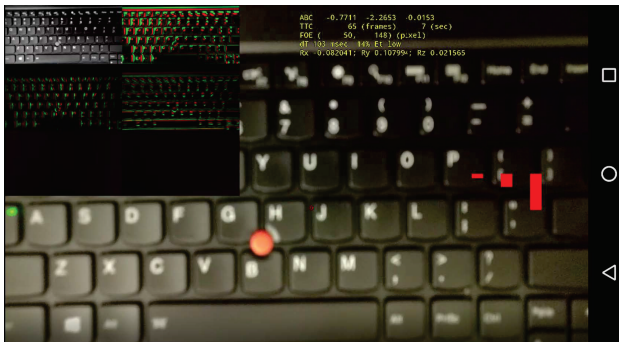
The Android TTC code contains three JAVA classes:

1) TTCCalculator: it extends the class "Activity", and calculate TTC using image frame data.
2) CameraPreview: it extends the class "SurfaceView", and displays the previewed image from the camera.
3) DrawOnTop: it extends the class "View", and shows the intermediate result for TTC calculation.

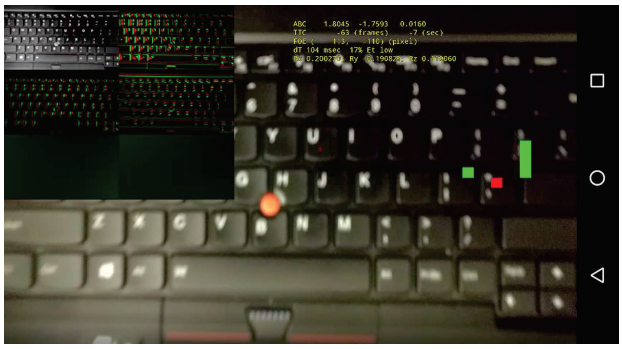
The sampling rate is set at 10 frame per second (fps). Thus the time interval is $\Delta t = 0.1$ sec. The resolution of the image frame is set to 720×1080 pixels. The image frame is downsampled by 4, i.e. to a 180×270 matrix, as the input data E. Fig. 3 shows some testing results running on Huawei Nexus 6P. The first four subimages in the top-left corner of Fig. 3(a) are the downsampled E, E_t, E_x, E_y, while the last two show the pixels' motion (in x and y direction, respectively) due to the phone's angular velocity — which is measured by the phone's gyroscope. The three "bars" in Fig. 3 show the computed parameters A, B, C, which indicate the motion (of the camera) in the X, Y and Z directions. Red means positive, while green means negative.

B. Two-registers recursive filtering

Note that the speed of both the current car and the leading car are continuous (velocity cannot change instantaneously; in fact, it's rate of change is limited by the maximum accelerations and decelerations possible for the vehicle). Thus,



(a) The third red "bar" indicate that the camera is approaching



(b) The third green "bar" indicate that the camera is leaving

Fig. 3. Some results on Nexus 6P. The top-left corner shows intermediate computational results. The three "bars" indicate the motions.

we expect the TTC ($1/C$) to also change smoothly. Due to noise in the image measurements, the estimates of C may be somewhat noisy also. In order to suppress noise we can use an approximate low-pass filter, e.g. $\overline{C}_k = \alpha C_k + (1-\alpha) C_{k-1}$ (with $0 < \alpha < 1$), to smooth the TTC output. One more register is needed to save the previous output C_{k-1}. The above is a simple finite impulse response (FIR) smoothing filter. To obtain better filtering we would have to remember additional old values. A more efficient approach is to use:

$$\overline{C}_k = \alpha C_k + (1-\alpha) \overline{C}_{k-1} \tag{13}$$

Only two registers are used. However, all previous outputs are used recursively. That is, eq. (13) can be rewritten as:

$$\overline{C}_k = (1-\alpha)^k C_0 + \sum_{l=0}^{k-1} \alpha (1-\alpha)^l C_{k-l} \tag{14}$$

This implements an infinite impulse response (IIR) smoothing filter. Here, the weights decay exponentially with time. This approach emphasizes new points over old ones, and does not require keeping a complete history of old values.

V. Experiments

We use a controllable robot car to test the TTC algorithm. Fig. 4 shows the experimental environment. The robot car is controlled to move at about 40 cm/sec. The screen of the smart phone is recorded by Android Studio IDE. Fig. 5 shows some experimental results. Fig. 5(a) is one frame from the video of

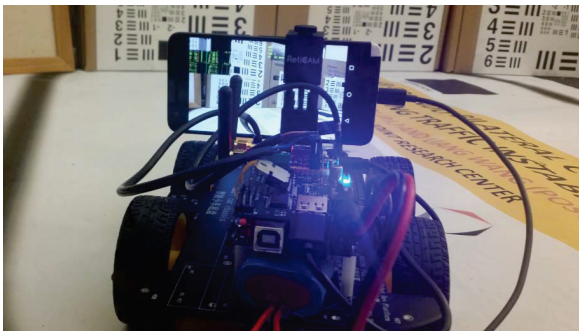


Fig. 4. The experimental environment. A smart phone is mounted on a controllable robot. The result on the screen is recorded on a laptop computer.

recorded results. Fig. 5(b) (2 sec. after Fig. 5(a)) is a frame when the robot was approaching the boxes. The upright bar on the right is colored red to indicate a dangerous condition (small positive TTC). Fig. 5(c) (4 sec. after Fig. 5(b)) is a frame when the robot was moving away from the boxes. The downward bar on the right is colored green to indicate a safe situation (negative TTC).

VI. Conclusion

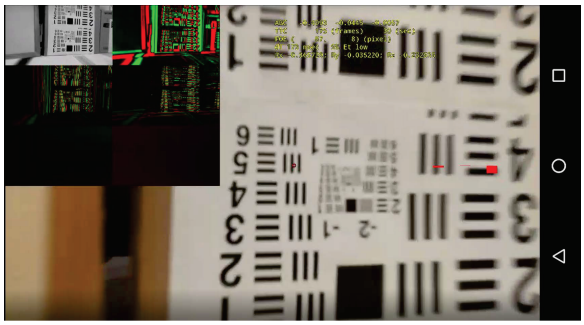
For safety and reliability of self-driving cars, relative velocity control is *important*. We could estimate the relative velocity by differentiating the distance. However, any noise in the distance measurement will be amplified by taking time derivative. Thus, relative velocity estimates by other approaches are need to generate more reliable and accurate measurements. In this paper, we provide a machine vision based approach to estimating the relative velocity. We first estimate the time to contact (TTC), and then calculate the relative velocity by multiplying the distance measurement by the inverse of the TTC. The whole system is implemented to run on Android smart phones in real time (See Fig. 4).

The TTC estimated from a time-varyiung image is not very accurate unless considerable filtering is used [5]. However, it provides valuable information to fuse with other relative velocity measurements. Moreover, TTC provides an important criterion for the safety of cars (and robots). If the TTC is small and positive, then a potentially dangerous situation may be developing and the system may need to enter an alarm state.

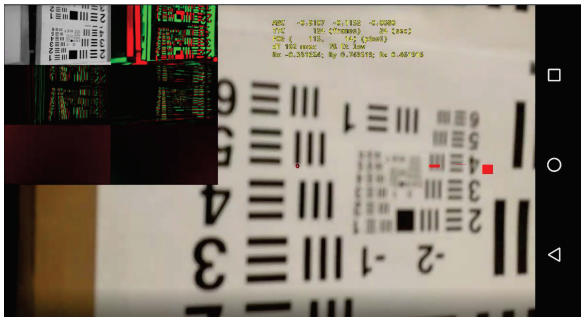
Vibration of the camera mounted in the car may cause the images to be smeared, which will adversely affect the quality of the estimated TTC. The gyroscope sensor in the smartphone provides a measure of the camera's rotational speed, which can be used to compensate for the image motion in the TTC calculation. This will be our future work.

Acknowledgment

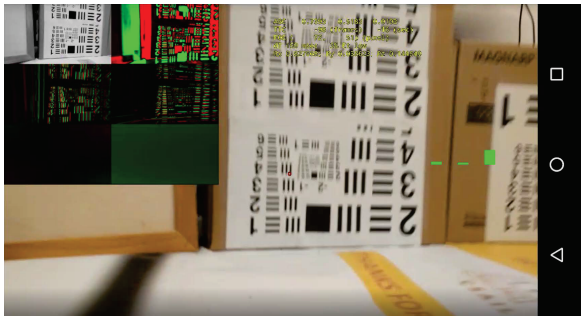
The authors would like to acknowledge the support of Toyota Motor Corporation (under grant LP-C000765-SR). We also thank Colin Bonatti, Byung Gu Cho, Daniel Lopez Martinez and Janille Maragh for the Android system implementation.



(a) One frame in the recorded result



(b) The car is approaching the boxes



(c) The car is leaving the boxes

Fig. 5. The result on Nexus 6P. (a) is one "reference" frame. (b) is a frame while approaching the boxes. (c) is another frame while away from the boxes.

References

[1] B. K. P. Horn, "Suppressing traffic flow instabilities," Intelligent Transportation Systems-(ITSC), 2013 International IEEE Conference on.

[2] T. Baran and B. K. P. Horn, "A Robust Signal-Flow Architecture For Cooperative Vehicle Density Control," *ICASSP* (2013).

[3] L. Wang, B. K. P. Horn and G. Strang, "Eigenvalue and Eigenvector Analysis of Stability for a Line of Traffic," Studies in Applied Mathematics, vol.138, January 2017.

[4] B. K. P. Horn, *Robot Vision*. MIT Press, Massachusetts, 1986.

[5] B. K. P. Horn, Y. Fang, and I. Masaki, "Time to contact relative to a planar surface," IEEE intelligent vehicles symposium. 2007.

[6] B. K.P. Horn and Brian G. Schunck, "Determining optical flow," Artificial intelligence 17.1-3 (1981): 185-203.

[7] A. R. Bruss and B. K. P. Horn, "Passive navigation," Computer Vision, Graphics, and Image Processing 21.1 (1983): 3-20.

[8] B. K. P. Horn, and E. J. Weldon, "Direct methods for recovering motion," International Journal of Computer Vision 2.1 (1988): 51-76.

[9] N. Shahriar and B. K. P. Horn, "Direct passive navigation." IEEE Trans. on Pattern Analysis and Machine Intelligence (1987): 168-176.

[10] I.S. McQuirk, B.K.P. Horn, H.S. Lee and J.L. Wyatt, "Estimating the Focus of Expansion in Analog VLSI," International Journal of Computer Vision, 28.3, (1998): pp. 261-?77.