Implementing Human-Acceptable Navigational Behavior and a Fuzzy Controller for an Autonomous Robot*

Vicente Matellán Olivera
Grupo de Sistemas y Comunicaciones
Universidad Rey Juan Carlos
e-mail: vmo@gsyc.escet.urjc.es

Reid Simmons
Robotics Institute,
Carnegie Mellon University
e-mail: reids@cs.cmu.edu

Abstract

Robots are just starting to appear in peopled environments but in order to be accepted by humans, they should obey basic people's social rules. In particular, they have to be able to move around without disturbing people. This means that they have to obey the social rules that manage the movement of people, for example following virtual lanes when moving through corridors, not crossing in front of moving people, etc. In this paper some of these aspects are explained, as well as the implementation of preliminaries works to implement the proposed solutions are described. So, a slight modification to the Lane-Curvature Method is presented to improve the behavior of a mobile robot when crossing people in a corridor. Other works needed to test this modifications in the robot Amelia of the Reliable Autonomous Systems Lab, as the implementation of a fuzzy controller, are also described in this paper

1 Introduction

A lot of interest has been shown in recent times in having service robots, museum tour guide implemented by robots as Minerva [6], Eldi [11], etc. are good examples of working systems. This kind of robots are going to be moving around spaces that people inhabit and interacting with a lot of people who don't know anything about robots. However, most of the work made in mobile robots navigation has only been concerned with theoretical methods for obstacle avoidance [1, 8] without considering human interaction aspects.

People instinctively will expect that robots behave as humans in the trivial issues as the movement. However, very few research has been done in adapting robots to the social rules of humans. One of the places where this issues are being studied is the RASL Reliable Autonomous Systems Lab of the Carnegie Mellon University (Pittsburgh, USA) where social behavior of robots are being developed. For instance, the behavior of waiting in a line has been implemented [13] in this lab. This behavior employs the notion of "personal space" for modeling a line of people and uses a stereo vision system to recognize people lines.

Vikia [12] is the name of the robot developed in the RASL in the idea of making robots more acceptable to people. This robot has been given a realistic personality, including her own personal history as the first robotic student at Carnegie Mellon University. Vikia's face has been built on a flat-screen monitor which is used to display an animated computer graphics model of a female face. This face lip-syncs any dialogue that the robot says using text-to-speech software and can be programmed to exhibit a variety of facial expressions. This flat monitor, which present Vikia face, is mounted on a robotic platform based in a commercial B21 robot named Amelia (see figure 1), the successor of the famous Xavier robot[10].

Amelia robot has a top speed of 32 inches per second, while improved integral dead-reckoning insures extremely accurate drive and position controls. The battery life is six hours, and are hot-swappable, so Amelia does not have to be powered down to change them. 2 Pentium-100s are the main CPU's on board with special shock-mounted hard drives. A 75Mhz 486 laptop acts as on-board console. All these are inter-connected by an internal 10M Ethernet, and to the world via a 2Mbs Wavelan wireless system. Sonar and infrared sensor arrays ring the robot, mounted on Smart Panels for quick and easy access to internal components. These Smart Panels also contain bump sensors. A Sony color camera is mounted on a Directed Perception pan/tilt head for visual sensing.

*The work described in this article has been partially supported by the Spanish Ministry of Science and Technology under the PROFIT program (FIT-070000-2001-118).

Figure 1: Amelia robot crossing the author of this paper in a corridor

This paper briefly summarizes the work I did while a short visit to the RASL during summer 2001. Basically, it comprised a slight modification to the Lane Curvature Method [3] to include the preferred turn direction in order to implement a social acceptable way of crossing people in corridors. Besides, I implemented a basic mechanism to write fuzzy reactive controllers for this robot, in order to easily test reactive behaviors. Next section describes the general software architecture that control the robot. The next one focuses on the LCM method for local obstacle avoidance and the modification made to behave in a more social way. Following section describes a Fuzzy controller added to the basic control of the Amelia software control in order to test the modifications made. Last section present some preliminary conclusions of these works.

2 Modifying the LCM

The modification made to implement the corridor crossing behavior has consisted in modifications to the Lane-Curvature Method (LCM), the local avoidance method for indoor mobile robots used in the navigation module. This method combines Curvature-Velocity Method (CVM) with a directional method named Lane Method. The lane method divides the environment into lanes, and then chooses the best lane to follow in order to optimize the along the desired heading. A local heading is then calculated for entering and following the best lane, and CVM uses this heading to determine the optimal translational and rotational velocities considering the current heading direction, physical limitations of the robot, and the environmental constrains.

The modification made to the original LCM proposed in [3] basically consists in considering the socially preferred turned direction when choosing the best lane to follow. In order to understand these modifications both CVM and LCM methods are going to be explained. Then, the modifications made will be detailed.

2.1 The Curvature-Velocity Method

The Curvature-Velocity Method (CVM) presumes that the robot can control both translational and rotational velocity, but cannot turn instantaneously, so the robot travels along arcs of circles. The basic idea is to add constrains to the velocity space and to choose the point in that space that satisfies the physical constraints of the robot and the obstacles and maximizes an objective function which prefers higher speeds, trajectories that travel longer before hitting obstacles, and should try to orient the robot to head in the desired goal direction.

To define this objective function the main variables are tv, the translational velocity, and rv, the rotational velocity. The objective function that has to be maximized is defined as:

$$f(tv, rv) = \alpha_1 dist(tv, rv) + \alpha_2 head(rv) + \alpha_3 speed(tv)$$

where

- $dist(tv, rv) = D_{limit}(tv, rv, OBS)/L$ is the arc distance that the robot can go with curvature $c = rv/tv$ before hitting a set of obstacles OBS. This value is normalized by a limiting distance L (3 meters in the current version of CVM installed in Amelia).

- $head(rv) = 1 - |\theta_c - rvT_c|\pi$ is the normalized difference between the commanded heading θ_c and the heading the robot will take if it turns at rv for some time T_c.
- $speed(tv) = tv/tv_{max}$ is the normalized translational speed.

This function has to be maximized taking into account the physical constrains of the robot that limit its translational velocity, rotational velocity, translational acceleration, and rotational acceleration within a predefined maximum values.

As a summary, CVM finds a point in the translational-rotational velocity space satisfying the constrains and maximizing the function, that is achieving the highest speed movement close to the commanded heading direction, while traveling longer before hitting the obstacles. α values are the weights to be given to each normalized term of the function. Typical experimental values of these weights are $0.6, 0.1, and 0.3$. More details of the CVM can be found in [8].

2.2 The Lane-Curvature Method

To find a heading direction for collision free movement, the Lane-Curvature Method [3] (LCM) divides the environment into lanes oriented in the direction of the desired goal heading. Then it selects the best lane for collision free and efficient motion. Finally, it calculates a heading direction to enter the selected lane.

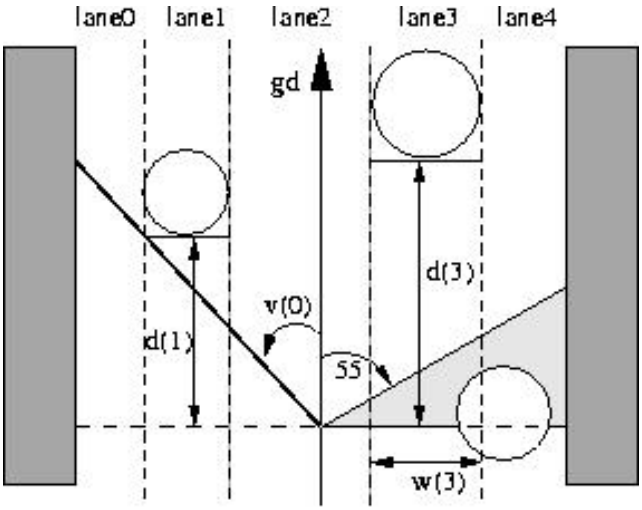

Figure 2: Lanes and their definition

Figure 2 show how lanes are defined. First any obstacle detected by the sensors is augmented to a circle. Then lanes are built in the direction of the robot goal heading (gd). For each lane k, $w(k)$ indicates the width of the lane, $d(k)$ indicates the distance that can be traveled before hitting obstacle k, and $v(k)$ is the angle to enter lane k from gd.

The space is divided into lanes according to the obstacles detected by the sensors only in front of the robot, obstacles behind the robot are ignored, and behind is defined as 55 degrees if there is an obstacle in the line defined by the robot actual situation (shadowed are at the robot left), and 90 if there are no obstacles in that line (robot right).

The LCM algorithm selects the lane with the longer collision-free distance (d) and the wider lane width (w). This selection is made using a linear function:

$$f_s(k) = \beta_1 d(k) + \beta_2 w(k) - \beta_3 ad_{v,cd}(k) - \beta_4 ad_{v,gd}(k)$$

where:

- $ad_{v,cd}(k)$ indicates the preference for smaller change in the current heading direction.
- $ad_{v,gd}(k)$ indicates the preference for a heading command closer to the current robot orientation.

Each term is normalized by the corresponding maximum values. The β values are the weights to be given to each term of the function. Experimental tuning of these parameters have shown (see [3]), that a relation $\beta_1 : \beta_2 : \beta_3 : \beta_4 = 6 : 1 : 6 : 1$ selects a wide, collision-free, and motion-efficient lane.

Once the best lane has been chosen CVM is used to enter that line. The values of α_k used in the LCM approach differ from those used when CVM is the only avoidance mechanism. In the LCM version they are set to be $\alpha_1 = 0.1$, $\alpha_2 = 0.6$, and $\alpha_3 = 0.3$, while they are set to be $\alpha_1 = 0.6$, $\alpha_2 = 0.1$, and $\alpha_3 = 0.3$ if CVM alone is used for obstacle avoidance. The main reason is that α_1 represents the importance of long, collision-free arcs, and it is lowered in LCM because obstacle avoidance is considered in the LCM. However, α_2 which dictates the importance of staying close to the goal heading, is set higher in LCM to force the robot keep closely to the heading command.

2.3 The Modified LCM

LCM is the method used in Amelia robot. It is a reliable method as a local navigator. However, its behavior was considered not "socially acceptable". In few words, the robots just evaluates the *best* path, not taking into account how this path interferes with human paths in its environment.

Figure 3 shows this desired "social behavior". Let's suppose that the robot goal direction (gd in the LCM previously explained) is 90^o, that is, the robot wants to take the right corridor. We will also suppose that the local navigation method is LCM and that the obstacle represented corresponds to a human moving down.

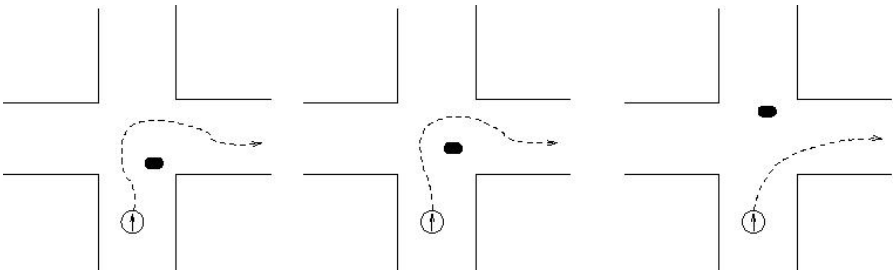

Figure 3: Examples of social behavior

In the first situation, numbering from left to right, the robot is behaving in a "human" way, letting the human follow her way without disturbing her. The second and the third situations represent the conflict situation consider in this work. We can think that the robot has to take the shortest path when this selection don't disturb the human crossing it.

It is important to notice that the sensor map has no information about the movement or direction of the moving obstacles. This kind of information in the TCA architecture is supposed to be in a higher level, so there is no way to take into account this kind of stimuli, only static information can be considered.

Let us analyze these two situations in more detail. Figure 4 represents the more relevant data, the goal direction, the widths, and the distances without obstacles in each of the lanes.

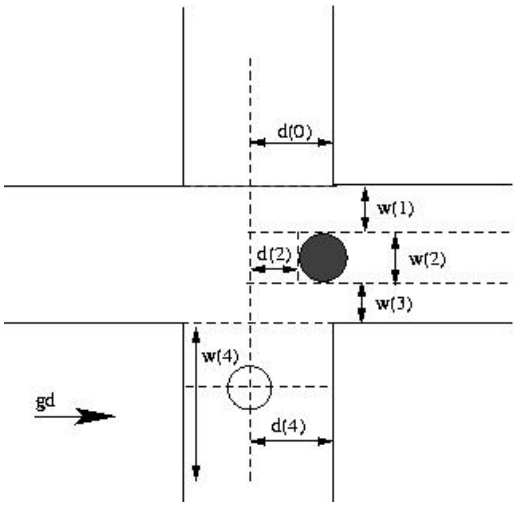

Figure 4: Lines of LCM in the second situation

We will consider that the distances $d(1)$ and $d(3)$ are equal (the normalized maximum), also the difference

in orientation of these lanes $v(1)$ and $v(3)$ to the gd are the same. Then, the only difference between these cases is the width of the lanes $w(1)$, $w(3)$. This means that the LCM method will decide which lane in a typical crossing attending just to the relative width of the virtual lanes.

$$f_s(k) = \beta_1 d(k) + \beta_2 w(k) - \beta_3 ad_{v,cd}(k) - \beta_4 ad_{v,gd}(k) + \beta_5 d_{prefd}(k)$$

where d_{prefd} is the distance to the preferred direction. This distance is calculated as the distance from the lante k to the closest line to the preferred direction. In the current implementation of the LCM method for RASL robots lane 0 is the lane most to the left, and $numLanes-1$ is the one most to the right. $prefd$ has been implemented in the input controller as 0 or 1, where 0 means "right", 1 means "left", and 2 means that there is no preferred direction and that traditional LCM will be used. This has been implemented in human given way because preferred direction is a "cultural" variable. For instance, in Japan people tend to go through the left side of busy corridors, while in Spain people usually go through the right one. This same behavior can be observed in less crowed corridors [5]. Obviously, the $d_{prefd}(k)$ factor is normalized acording to the maximum value of the current situation.

In summary, this factor will make the robot prefer lanes closer to that orientation. This means for instance that if the preferred direction is "right" the robot should try to go on its right, that is closer to the right wall in a corridor.

This modification was tested using the Amelia robot in an indoor environment, as shown in figure 1. Same crossing situations, similar to the ones of figure 3 were tested using both the classical LCM method and the modified version described, showing that this last one chooses lines in a more "natural" way, that is, humans can behave normally when crossing the moving robot.

3 Implementing of a Fuzzy Controller for RASL robots

When implementing the modifications described in the previous section, one of the main problems I faced was that CVM, and LCM are not easily "adaptable". The architecture that controls the robots TCA [7] includes an implementation of some basic behaviors: *follow-wall*, *wander*, *follow-direction*, etc. using the CVM and the LCM. However it is difficult to implement new ones.

In order to provide an easy way of designing low level behaviors for the RASL robots I made an implementation of a fuzzy controller driven by a set of rules that could be easily modified.

This section mainly tries to explain the fuzzy controller implemented for RASL robots. The goal was that its behavior could be governed by a set of fuzzy rules read from a text file. So, to change the behavior of the robot we just need to change the rules written in a file. The use of a fuzzy controller let modify the behavior of the robot without having to modify the source code of the controller, or at least modifying it slightly.

A trivial example of rules is provided in this paper. It just takes into account the distance and orientation to the nearest obstacle in front of the robot, and returns the translational and rotational velocity to wander around avoiding obstacles. So, this fuzzy controller has just got two inputs and two outputs. In the real `fuzzy_rules.dat` file 5 controllers are provided, implementing the existing four basic behaviors: wander, go to goal, follow direction, and follow wall. The fifth one is used when the new command *fuzzycontrol* is used.

3.1 Behaviors based in fuzzy-rules files

I have written four fuzzy-rules files to give the same functionality (wander, go to goal, follow direction, and follow wall) as the current controllers of the RASL control software provide. Each of this controllers has three major parts: inputs, outputs, and rules. Each part starts with a line beginning with mark ("*"). Anything written after a mark in the same line is considered as a comment. Marks are always at the beginning of lines.

Both, inputs and outputs, are made up by an arbitrary number of variables, each of them beginning with the mark "{", followed by the name of the variable, and ended by the mark "}". Each of the variables is defined by an arbitrary number of labels. The following is an example of a variable definition:

```
{ Orientation to closest obstacle 270-0 (L=left, R=right)
0
VR -1 0 45 90
R 0 45 85 90
F 80 85 95 100
L 90 95 135 270
VL 90  135 270 275
}
```

Each of the labels is defined in a different line, and its definition is made up by the four abscissa coordinates of a trapezoid. For instance, the previous variable "O" is defined by five labels: VR, R, F, L, and VL (where R means "right", L means "left" and V stands for "Very"), which correspond to the trapezoids drawn in figure 5

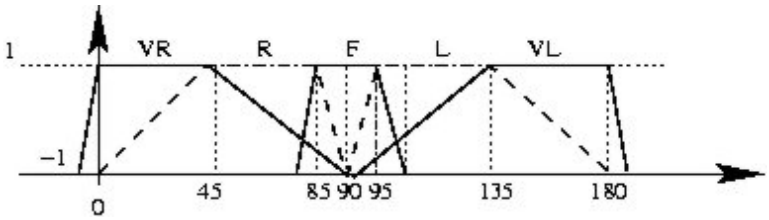

Figure 5: Definition of the labels of the variable Orientation (O)

Rules have two parts, antecedent (premise, if-part) and consequent (conclusion, then-part). An arrow, written as "=>", divides the two parts. A simple rule could be *O es R* => *RV is L*, which should be read as "If **O**rientation **is R**ight **THEN R**otational **V**elocity is **L**eft".

Both, antecedent and consequent may be composed of various conjunctions (ANDs) of these statements. So more sophisticated rules can be written:

```
D es VN & O es L => TV es S & RV es VR
```

which should be read as "If Distance is Very Near **AND** Orientation is Left THEN Translational Velocity is Slow **AND** Rotational Velocity is Very Right".

The *OR* operator has not been implemented, but it could be used by splitting the rules. For instance, the rule:

```
D es VN  || O es L => TV es S & RV es VR
```

Could be write as:

```
D es VN => TV es S & RV es VR
O es L => TV es S & RV es VR
```

3.2 A trivial example of a fuzzy controller

The following file defines the simplest controller: the one for wandering without bumping into obstacles. Two inputs are defined: distance (D) and orientation (O) to the closest obstacle. Two outputs are also defined: Translate velocity (TV) and rotational velocity (RV). Just eight rules define the behavior of the robot.

```
* Inputs (Label values of the trapezoid)
{ Distance to closest obstacle  (N=Near, F=Far)
D
VN  -100 0 35 40
N 15 35 55 75
F 35 55 80 150
VF 55 80 250 300
}
{ Orientation to closest obstacle [local] R=right (0-90) L (90-180) V= Very
O
VR -1 0 45 90
R 0 45 85 90
F 80 85 95 100
L 90 95 135 180
VL 90 135 180 181
}
* Outputs (Label 4 values of the trapezoid)
{ TV: traslate velocity (max 30). Stop, Low, Medium, High
TV
S -1 0 0 1
L -1 0 10 40
M 0 10 30 60
H 10 30 60 61
}
{ RV rotational velocity. VLeft, Left (negatives), Right, VRight (positives)
RV
VL -20 -10 -5 0
L  -10 -5 0 1
R  -1 0 5 10
VR 0 5 10 20
}
* Rules ()
{
D es VN & O es L => TV es S & RV es VR
D es VN & O es VL => TV es S & RV es R
D es VN & O es R => TV es S & RV es VL
D es VN & O es VR => TV es S & RV es L
D es N & O es L => TV es L & RV es R
D es N & O es R => TV es L & RV es L
D es F => TV es M
D es VF => TV es H
}
```

4 Conclusions

As result of the two works made (modification of the LCM and the addition of a fuzzy controller), the control interface of the control software of the RASL robots has been changed. The traditional console or remote control of the robot is made through the CTR [2] interface. This interface now has been modified to include the new functionality as shows the figure 6.

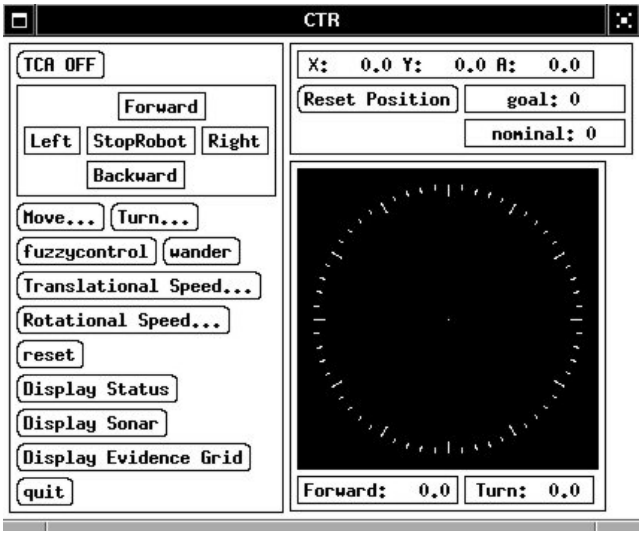

Figure 6: CTR controller

Besides the graphical interface, CTR also offers a text interface. From that interface now the **W** can be used to invoke the fuzzy controller. Another text command has also been implemented **S <direction>** to set the goal direction. There were another command (**d**) which also set the goal direction, but it called CVM, LVM, methods to command the robot toward that goal. **S** just set the goal direction without calling any reactive controller. The direction given as argument to **S** is given in DEGREES from the current heading direction of the robot.

The command for selecting the lane+curvature method has been modified and now a parameter can be given. This parameter tell the robot which is the **Preferred Turn Direction**. Possible values of this command are: 0 (meaning right), 1 (meaning left), and 2 (meaning that there is no preferred direction).

Finally, another *Local Navigation Method* has been added: **(fu)zzy/z**. So, now vector field, potential field, curvature method, lane+curvature method, and fuzzy controller can be selected, as shown in the figure 6.

The two works described in this paper are preliminary tasks only relevant for users of the RASL robots in particular, and of TCA software in general. Obviously, it has to be completed in order to present real results. This paper just tries to keep the work done documented, and to serve as a starting point for the discussion of further works.

References

[1] J. Borenstein. *Real-Time Obstacle Avoidance for Fast Mobile Robots*. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 5, pp. 1179-1187. Sept/Oct 1989.

[2] Joseph O'Sullivan, Karen Zita Haigh and G.D. Armstrong *Xavier, Manual v 0.4* Carnegie Mellon University, 1997.

[3] N. Y. Ko and R. Simmons. *The Lane-Curvature Method for Local Obstacle Avoidance*, Proceedings of Conference on Intelligent Robotics and Systems, Vancouver Canada, October 1998.

[4] M. Malmberg, *Human Territoriality: Survey of behavioural territories in man with preliminary analysis and discussion of meaning*, Mouton Publishers, 1980.

[5] R. Sack, *Human Territoriality*, Cambridge University Press, 1986.

[6] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. *Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva* International Journal of Robotics Research, Vol. 19, No. 11, November, 2000, pp. 972-999.

[7] R. Simmons, R. Goodwin, K. Zita Haigh, S. Koenig and J. O'Sullivan *A Layered Architecture for Office Delivery Robots*, First International Conference on Autonomous Agents, Marina del Rey, CA, February 1997.

[8] R. Simmons, *The Curvature-Velocity Method for Local Obstacle Avoidance*, International Conference on Robotics and Automation, Minneapolis MN, April 1996.

[9] R. Simmons, R. Goodwin, K. Zita Haigh, S. Koening and J. O'Sullivan, *A Layered Architecture for Office Delivery Robots*, In Proc. of Autonomous Agents, pp.245–252, 1997.

[10] R. Simmons, J. Fernández, R. Goodwin, S. Koenig, J. O'Sullivan, *Lessons Learned From Xavier*. IEEE Robotics and Automation Magazine, Vol 7, No 2, pp 33-39, June 2000.

[11] M. Castrillón et al. *Eldi's activities in a Museum*. In Actas of WAF'2001 (Workshop on Autonomous Agents), pp. 61-73, Móstoles (Madrid, Spain), 2001.

[12] A. Burce, I. Nourbakshsh, R. Simmons, *The Role of Expressiveness and Attention in Human-Robot Interaction*. AAAI Fall Symposium, oston MA, October 2001.

[13] Y. Nakauchi and R. Simmons *A Social Robot that Stands in Line*, . In Proceedings of the Conference on Intelligent Robots and Systems (IROS), Takamatsu Japan, October 2000.