Vision Functions for the Guidance of Smart Vehicle

Swee Meng Wong
School of Mechanical and Production Engineering
Nanyang Technological University, Singapore
p7316503b@ntu.edu.sg

Ming Xie
School of Mechanical and Production Engineering
Nanyang Technological University, Singapore
mmxie@ntu.edu.sg

ABSTRACT

Numerous accidents occurred because the human driver failed to perceive danger due to human error or simply drove too rashly. If a robust electronic driver is developed that includes sensors, active vision system, GPS system, etc that can assist the human driver, these accidents may be avoided. This motivates our research in developing an electronic driver. In this paper, we discuss the perception aspects of the electronic driver and provide results with real experiments.

1. Introduction

An electric vehicle CyCab is used as our research platform. Our research is to develop an electronic driver and add perception capability to this CyCab vehicle. This electronic driver will eventually co-exist with the human driver in order to provide driving assistance or automated driving behaviours like parking and car following, etc. In this paper, we discuss some problems faced by on-vehicle vision system and the proposed solution to the problems. The discussions are centred on the use of active vision for vehicle guidance.

2. Lane Finding

2.1. Problem Statement I

One of the roles of an electronic driver is to identify the lane in which it will travel. We could identify a lane by its left and right boundary markings. These markings could be elevated curbs or color strips along the lane. A proposed solution to detect a lane is discussed in this section.

Related research works on lane finding include the use of Hough Transforms, color segmentation method, grey-level thresholding method, etc. Our proposed method uses edge linking that is based on the definition of the Causal Neighborhood Window (CNW) [7] and the Horizontal Edge Element (HEE) [7]. We find that this method is simple to implement and could be effectively used to find lanes.

2.2. Propose solution to Problem Statement I

The inputs are images captured by the on-vehicle active vision system. Our proposed method will process the input images so as to extract the lane information from them. A simple block diagram that describes this process is shown in Figure 1.

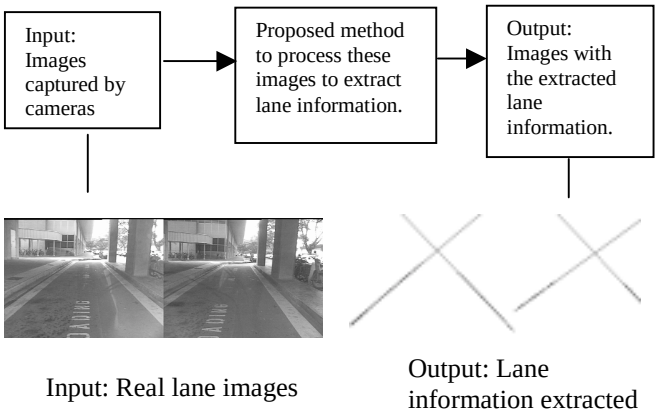

Figure 1. Lane finding block diagrams.

Two cameras are mounted at the front of the vehicle. They will capture the front view seen from the car. Breakdowns of the steps on the proposed solution are listed below.

Step 1. Capture front view images.
Step 2. Perform edge detection on images to obtain edge maps.
Step 3. Perform edge linking using Causal Neighborhood Window (CNW).
Step 4. Identify: (a) the longest chain with a slope angle close to 45 degree (that is the most probable candidate for the left boundary of a lane) and (b) the longest chain with a slope

angle close to 135 degree (that is the most probable candidate for the right boundary of a lane).

It is assumed that the elevated curbs or the color strips that mark the lane are distinct from the road itself. With this assumption, we can perform edge detection on the input images to obtain the corresponding edge maps as described in step 2.

An edge detection algorithm developed by R. Deriche [6] is used here to get the edge maps of the input images.

The next step is to interpret the edge maps. To do this, an edge linking process is performed on the edge maps [7]. This edge linking process is designed to detect lines that particularly fit the left and right boundaries of the lane.

2.3. Edge Linking

In this section, the edge linking strategy is discussed. As the edge linking strategy is based on [7], many references will be made to it. However, only the relevant details that could sufficiently explain the mechanism to the reader will be provided here

2.3.1. Definition of Contour Chains

In this section, we will discuss the definition of a contour chain [7] that is used in the edge linking process. Please refer to Figure 2.

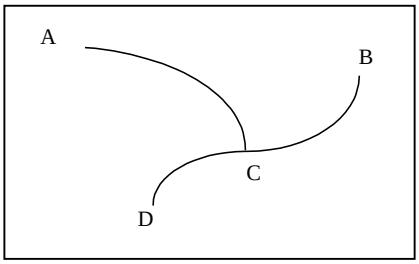

Figure 2. Contour Chains definition.

From the figure 2, it could be shown that there could more than 1 definition of "contour chains" depending on the edge linking strategy. We can define 3 contour chains {AC, CD and BC} which is separated by a junction point C. Instead it could be said that there are 2 contour chains {AC and BCD}. In the application of our edge linking process, the first interpretation of contour chain is used instead.

The definitions of junction [7] and contour chain [7] are as follows:

Junction: "A *junction* consists of an edge element at which two or more contour chains intersect".

Contour chains: "A *contour chain* is a set of linked edge elements, in which only its two end edge elements may constitute junctions".

The task of edge linking in our application is now reduced to finding the longest contour chain that best fit the description of the left and right boundary lines.

2.3.2. Horizontal Edge Element

To further understand the edge linking strategy, let us introduce the definition of horizontal edge element (HEE). By definition in [7], "a horizontal edge element is a set of adjacent edge pixels aligned in the same image line". A HEE is defined by its left endpoint (x_{left0}, y_0) and right endpoint (x_{right0}, y_0). Figure 3 illustrates the definition of a HEE.

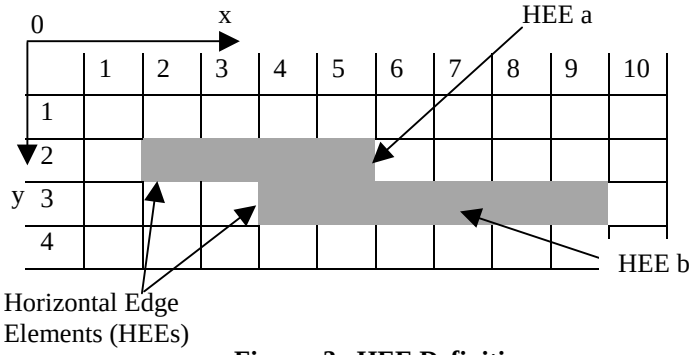

Figure 3. HEE Definition.

There are two HEEs in figure 3. HEE *a* is defined as (x_2, x_5, y_2) and HEE *b* is (x_4, x_9, y_3). Interpreting the coordinates, it means HEE *a* lies between units 2 to 5 in the x axis and at unit 2 in y axis.

This mapping of edges into HEEs enables us to create causal neighborhood window (CNW) based on them.

2.3.3 Causal neighborhood window (CNW)

We can create a CNW once the set of HEEs has been identified from the edge map. By definition in [7], a CNW positioned at a horizontal edge element is defined in Figure 4, where x_{gap} is the maximum gap value in X direction and y_{gap} the maximum gap value in Y direction.

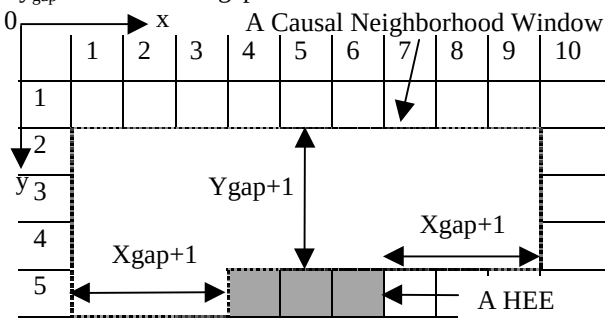

Figure 4. Definition of CNW.

Therefore, the corresponding window region in OXY is defined by:

$$\{x_{left\ 0} - x_{gap} - 1 \le x \le x_{right\ 0}\ ;\ y = y_0\},$$
$$\{x_{left\ 0} - x_{gap} - 1 \le x \le x_{right\ 0} + x_{gap} + 1;\ y_0 - y_{gap} - 1 \le y \le y_0 - 1\} \quad (1)$$

2.3.4. Edge Linking – Contour Chain creation

Now that the HEE and CNW are defined, we can proceed to discuss the creation of contour chains using CNW as in step 3. For an image captured by the camera, it is scanned line by line, (from top to bottom, left to right) to label any HEE found. When this is done, a CNW is created for every HEE that is labeled and the linking operations will be carried out as follows.

Supposed a causal neighborhood window (x_{gap}, y_{gap}) has been created at a HEE j: ($x_{left\ j}$, $y_{right\ j}$, y_j), if any other HEE i: (xleft i, y right i, y i) verfies the following relations:

$$y_j - y_{gap} - 1 \le y_i \le y_j - 1,$$
$$x_{left\ i} \le x_{right\ j} + x_{gap} + 1,$$
$$x_{right\ i} \ge x_{left\ j} - x_{gap} - 1,$$

Or (2)

$$y_i = y_j,$$
$$x_{right\ i} < x_{left\ j},$$
$$x_{right\ i} \ge x_{left\ j} - x_{gap} - 1$$

then the HEE i is called the neighboring HEE of HEE j.

Suppose again if HEE i is found to be a neighboring HEE of HEE j, a CNW will be created that is positioned on HEE i. A search for any neighboring HEE of HEE i will be carried out. This recursive process is continued until no neighboring HEE is found. However, the labeling of neighboring HEE stops when a junction is detected. This recursive process is similar to the labeling of the contour chains described in section 2.3.1.

2.3.5. Parameters in Contour Chain creation

This section describes the parameters that we can set when creating contour chains. The choice of the parameters is also discussed.

In our application, we want to extract the lane boundary information. Figure 5 is an image of a lane captured by a pair of cameras.

Figure 5. Real image of a lane.

After edge detection by using the algorithm in [6], the output edge maps are shown in figure 6.

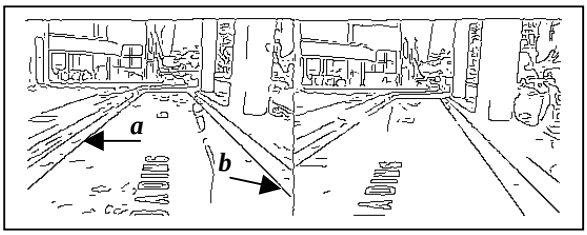

Figure 6. Edge maps.

The key parameters that can be set are (x_{gap}, y_{gap}) that define a CNW. Our task is to use edge linking method to extract the boundary lines automatically from Figure 6.

We can make a few observations from Figure 6.

1. The left lane boundary line (a) makes an angle close to 45°.
2. The right lane boundary line (b) makes an angle close to 135°.
3. The lane boundary lines are connected straight edges inclined close to either 45 or 135 degrees.

The first task is to label all the HEEs from the edge map. After this process, we would customize a Causal Neighborhood Window (CNW) that would describe the characteristic of the lane boundary lines.

To find the left lane boundary, a modified CNW is defined in Figure 7.

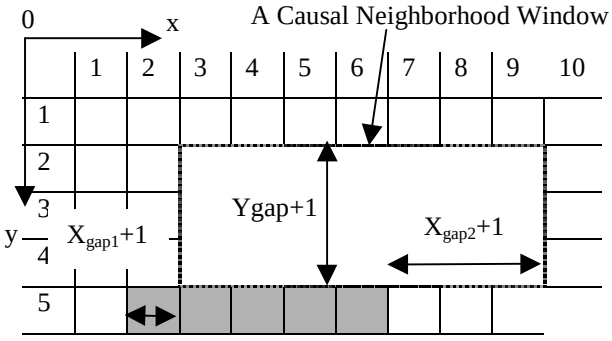

Figure 7. CNW to find left lane boundary.

Where x_{gap1}= -2 ,x_{gap2}= 2 and y_{gap}= 2.

Though the modified CNW now has two x_{gap} parameters instead of one, the definition of CNW is still the same. The values of x_{gap1}, x_{gap2} and y_{gap} are chosen to detect contour chains that resemble the left lane boundary edges. The contour chains formed by using this CNW tend to lean on the right, and they resemble the left boundary edges that made an angle close to 45°. This CNW is thus designed to detect the left boundary edges.

Similarly, the CNW used to detect the right lane boundary line is shown in Figure 8.

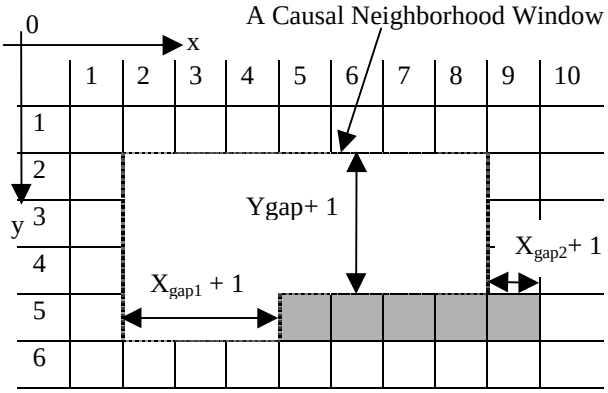

Figure 8. CNW to find right lane boundary.

Where x_{gap1}= 2 ,x_{gap2}= -2 and y_{gap}= 2.

The setting of these parameters is designed to search for contour chains that lean toward the left, ie. contour chains that make an angle close to 135°. This setting is designed to look for the right boundary edges.

Experiments can be performed using different combinations of x_{gap1}, x_{gap2} and y_{gap} in order to find a better settings. Our result is shown in figure 1.

This preliminary results showed that this method would allow us to find the lane boundaries. The Causal Neighborhood Window (CNW) can be adjusted adaptively to give robust results.

3. Obstacle Detection

3.1. Problem Statement II

Another task of an electronic driver is to be able to detect obstacle automatically. This is important for avoiding collision. In this section, a method based on 2D vision will be discussed for the detection of obstacle on road surface.

3.2. 2D vision principle

Our approach uses the principle of 2D vision. The ground plane that the vehicle moves on is assumed to be a 2D space. In this case, the image coordinates are related to the object coordinates by a 3x3 transformaton matrix. This matrix can be obtained by a 2D calibration process. Figure 9 shows a 2D plane on which 2D calibration is performed.

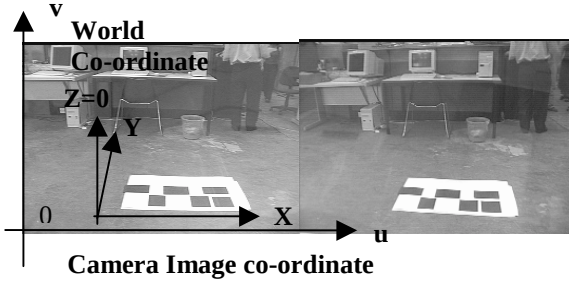

Figure 9. The 2D plane defined in 2D calibration.

The Z coordinate is set to zero in 2D calibration. The X and Y-axes therefore represent the 2D ground plane. When the calibration is finished, we would be able to obtain a set of 3x3 transformation matrices that gives the relationship between the camera image coordinates and the world coordinates. Two calibrations are performed on the left and right cameras.

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}_{left} \times \begin{pmatrix} u_{left} \\ v_{left} \\ 1 \end{pmatrix} = \begin{pmatrix} k_1 X \\ k_1 Y \\ k_1 \end{pmatrix} \qquad (3)$$

3x3 transformation matrix $({}^{world}M_{left})$ — Screen coordinate — World coordinate

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}_{right} \times \begin{pmatrix} u_{right} \\ v_{right} \\ 1 \end{pmatrix} = \begin{pmatrix} k_2 X \\ k_2 Y \\ k_2 \end{pmatrix} \qquad (4)$$

${}^{world}M_{right}$

Equations 3 and 4 are obtained after the calibrations.

Equation 3 describes the mathematical relationship between the left image coordinates and 2D object coordinates, while equation 4 gives the right image with the 2D object. We want to derive a matrix that describes the mathematical relationship between the left and right image coordinates. This can be done by simple mathematical manipulations from the basic equations 3 and 4 as follows:

From (3)

$$\frac{1}{k_1} \times {}^{world}M_{left} \times \begin{pmatrix} u_{left} \\ v_{left} \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \qquad (5)$$

From (4)

$$\frac{1}{k_2} \times {}^{world}M_{right} \times \begin{pmatrix} u_{right} \\ v_{right} \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \qquad (6)$$

Substitute (5) into (6)

$$\frac{1}{k_1}\times {}^{world}M_{left}\times\begin{pmatrix}u_{left}\\ v_{left}\\ 1\end{pmatrix}=\frac{1}{k_2}\times {}^{world}M_{right}\times\begin{pmatrix}u_{right}\\ v_{right}\\ 1\end{pmatrix}$$

$$\frac{k_1}{k_2}\times({}^{world}M_{left})^{-1}\times {}^{world}M_{right}\times\begin{pmatrix}u_{right}\\ v_{right}\\ 1\end{pmatrix}=\begin{pmatrix}u_{left}\\ v_{left}\\ 1\end{pmatrix}$$

$$({}^{world}M_{left})^{-1}\times {}^{world}M_{right}\times\begin{pmatrix}u_{right}\\ v_{right}\\ 1\end{pmatrix}=\begin{pmatrix}k.u_{left}\\ k.v_{left}\\ k\end{pmatrix} \quad (7)$$

Finally, equation (7) gives us the direct relationship between the right image and the left image coordinates. Using equation (7), we could project the right image onto the left image. Based on a necessary condition, if any points on the right image belong to the 2D ground plane, when projected onto the left image plane, the points will superimpose with the corresponding points on the real left image. Figure 10 shows the processes of our approach.

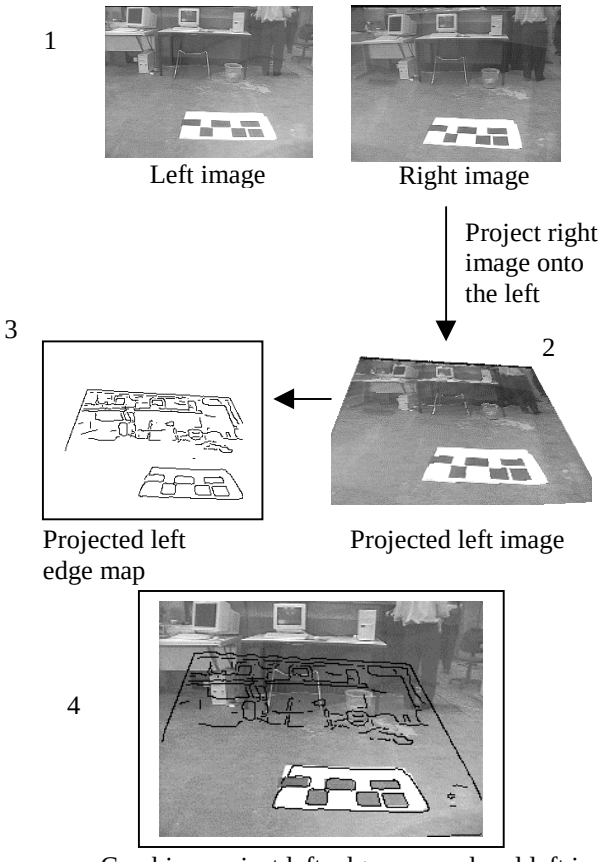

Figure 10. Obstacle detection processes 1-4.

The edge map of the projected left is superimposed onto the real left image as shown in figure 10. It can be observed that only the object on the ground plane fits with the edge map of the projected left image. Using this characteristic, we could separate out objects that are on the ground plane from elevated ones. Figure 11 shows the separated obstacle from the real image. The obstacle separation is implemented by setting a similarity threshold between projected left and real left images. This approach is simple to implement and enables us to quickly determine the presence of obstacles, whether static or moving.

Figure 11. Left picture is real image. Right picture is detected obstacle.

4. Distance Assessment between obstacle and the vehicle

4.1. Problem Statement III

When an obstacle is found to be in front of our vehicle, there is a need to find out the range of the obstacle. In our proposed method, we made use of the relationship between image disparity and range. Qualitative stereo vision principle is used.

4.2. Proposed Solution: Qualitative Stereo Vision

It is not imperative to obtain very accurate range information in our application. Therefore, using image disparity and range relationship will give us an approximate range information. Our proposed method is to obtain the image disparity value of the obstacle by using template matching technique. From the image disparity value, the range information can be estimated by using a pre-established look up table. This look-up-table qualitatively describes the relationships between image disparity and range.

An object of interest will appear on the left and right camera images.

Let Right(x) = x screen coordinate of object on right camera image.
Left(x) = x screen coordinate of object on left camera image.

Image disparity of the object is defined as *disparity* = Right(x) - Left(x). The corresponding points of the object on the left and right images can be known by using template

matching. These points are then used to calculate the image disparity of the object of interest.

The steps to obtain range information from our proposed method are as follows:

Step 1. Perform a simple image disparity calibration: obtain the range and disparity information from 2m to 6m in steps of 1m interval

Step 2. Build a look up table using range and image disparity data. (See figure 12).

Step 3. Use template matching method to locate the image disparity value of the object, then use the image disparity value to obtain range information.

In our application, least square method is used to model the relationship between distance and image disparity. Some results using this approach are shown in figures 13 to 14. These pictures were 2 sequence images estimating the range of the front vehicle. Our preliminary experiments show that this approach to obtain range information is simple and fast.

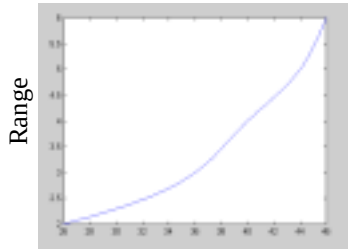

Figure 12. Relationship between Range and Image Disparity.

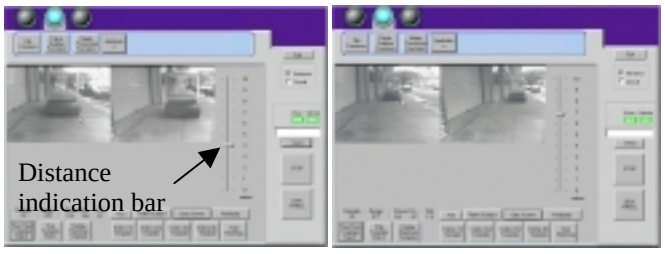

Figure 13. Distance detected at 3.93m shown on distance indication slider bar on right (Image sequence 1/2).

Figure 14. Distance detected at 6.71m. (Image Sequence 2/2).

5. Conclusion

This paper presented three solutions to solve three corresponding problems related to the visual guidance of smart vehicle. These visual functions are important for an on-vehicle electronic driver. Experiments with real images show the efficiency of the proposed methods. Our future works will be to integrate them together and to test their real-time performance.

6. References

[1] Ichiro Masaki. Vision-based Vehicle Guidance. Springer-Verlag New York, Inc. 1992.

[2] Yiannis Aloimonos. Visual Navigation. From Biological Systems to Unmanned Ground Vehicles. Lawrence Erlbaum Associates. 1997.

[3] Crisman, Jill D., and Thorpe, Charles E. (1990). "Color Vision for Road Following." Vision and Navigation: The Carnegie Mellon Navlab. Kluwer Academic publishers, Chapter 2.

[4] Kenue, Surender K. (1989). Lanelok: Detection of Lane Boundaries and Vehicle Tracking Using Image-Processing Techniques. Part I: Hough-Transform, Region-Tracing, and Correlation Algorithms. Proc. Mobile Robots IV, Society of Photo-optical Instrumentation Engineers, Bellingham, Washington, pp 221-223.

[5] Kenue, Surender K. (1989). Lanelok: Detection of Lane Bundzries and Vehicle Tracking using Image-Processing Techniques. Part II: Template Matching Algorithms. Proc. Mobile Robots IV, Society of Photo-optical Instrumentation Engineers, Bellingham, Washington, pp 234-245.

[6] Deriche, R. (1987). Using Canny's Criteria to derive a recursively implemented optimal edge detector. Int. J. Computer Vision.

[7] Ming Xie. Edge linking by using causal neighborhood window. Pattern Recognition Letters 13 (1992) pages 647-656, September 1992.

[8] Pomerleau, D. A. (1989). ALVINN: An Autonomous Land Vehicle in a Neural Network. CMU Technical Report, CMU-CS-89-107

[9] Dickmanns, E., and Zapp, A. (1986). A Curvature Based Scheme for Improving Road Vehicle Guidance in Computer Vision. Proceedings of SPIE Conference on Mobile Robots, Cambridge, Massachusetts.