Crosswalk Detection via Computer Vision

JASON BANICH

ADVISOR: DR. JOHN SENG

Goals

- Repurpose existing lane following algorithms for crosswalks
- Algorithms optimized for mobile devices
- Recognize multiple different kinds of crosswalks
  - Zebra
  - Standard
- Work under different lightings
- Don’t modify the environment
- Android, requiring no extra money spent

Related Work

- Walk Light detection
- Locating crosswalks with camera phone
- Camera phone system for orienting pedestrians

Walk light detection

- Utilize phone's accelerometer in order to determine horizon line (less of image to search)
- Use other factors to narrow down search (green traffic light, etc)

Fig. 2. Zoomed-up images of Walk lights, photographed from typical viewing distances, rendered at the lower resolution available to the camera running in video mode. Note that the Walk icon is blurry, hard to resolve clearly, and may be subject to color distortions.

Walk light detection cont

- Require users to hold phone sideways
  - (Future work suggested using accelerometer to auto-rotate image for algorithm)
- Main issue was aligning with crosswalk (hard to detect walk sign when its not in the image)

Zebra Crosswalk Detection

- Algorithm
  - Extract straight line segments from the image
  - Applies segmentation algorithm to group segments
  - Score the detected segments
  - Use a threshold to determine if score is high enough

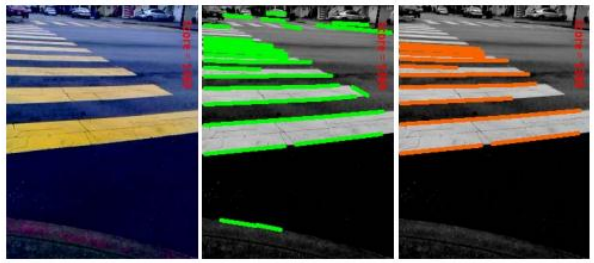

Fig. 2. (a) Typical intersection image photographed by camera phone. (b) Straight-line segments (green) extracted from image. (c) Segments (orange) chosen by algorithm as belonging to crosswalk. The algorithm assigns a score of 1369 to the image, well above the minimum threshold (500) for detecting a crosswalk.

Zebra Crosswalk Results

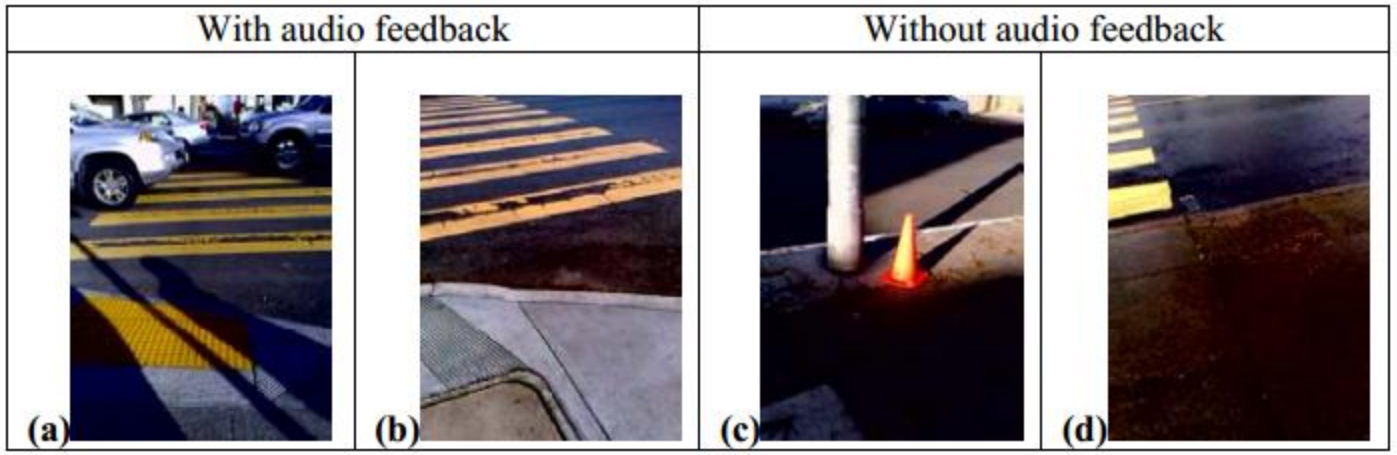

Fig. 4. Pictures taken by blind subject from Experiment 2. (a), (b) show images with audio feedback from the cell phone; (c), (d) show images without audio feedback.

Two-Stripe Crosswalks

- Algorithm
  - Read accelerometer to determine orientation
  - Discard data above the horizon line
  - Use brightness to estimate if points are in crosswalk stripes
  - Draw lines

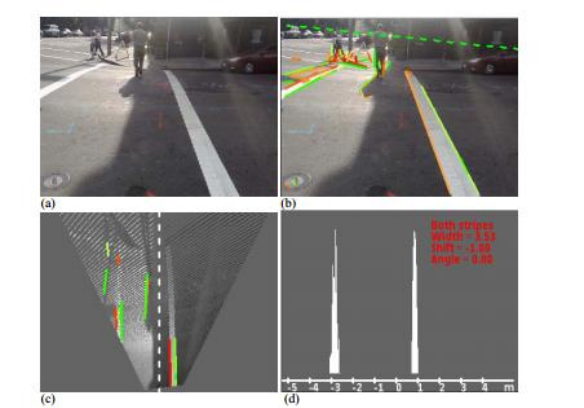

Fig. 2. Main steps of algorithm. (a) Input image. (b) Colored lines indicate straight segments; dashed green line is horizon. (c) Ground plane geometry shown in aerial view; dashed white line shows dominant direction. (d) Two peaks along x-axis (units in meters) indicate two stripes.

Simple Lane Following

- Algorithm
  - Convert to grayscale
  - Crop to center portion
  - Identify edges and draw onto new image
  - Apply Hough line detection to find shapes
  - Delete extraneous lines
  - Reapply lines to main image
  - Draw lines onto image

Validation

- Two main overall goals:
  - Successfully recognize crosswalks
  - Successfully guide users across crosswalks

Validation Cont

- Metrics to measure
  - CPU usage/frames per second
  - Jitter (in suggested direction)
  - Robustness (lighting)
  - Frame by frame analysis (leading user out of crosswalk a lot?)

Difficulties

- Video Jitter
- Methods of keeping user in crosswalk
- Recognizing end of crosswalk (when to stop routing)
- Adaptive thresholding
- Calculate vector that user is walking

References

- See related work/validation

Questions?