Interpolation Methods for Global Vision Systems

Jacky Baltes and John Anderson

Autonomous Agents Laboratory
University of Manitoba
Winnipeg, Manitoba, Canada R3T2N2

Introduction

- New field sizes in the Small-Sized league pose challenges for global vision systems
- Rather than changing mounting height or using multiple cameras, we advocate using a more model-based approach with a side camera view
- Many interesting problems with oblique views, including compensation for occlusion
- Our vision server, *Doraemon*, uses a side view and employs a standard camcorder with no wide-angle lens
- Larger field size requires better vision processing and interpolation to track smaller features. This work compares the accuracy and efficacy of several interpolation methods for a soccer domain, and illustrates a reconstructed overhead view of the playing field

Doraemon/Vision Kart

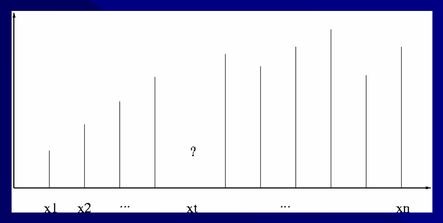
- Mobile vision Kart consists of a p4 2.4 GHz small-sized PC, tripod-mounted camera, infrared transmission mounted on the side of the kart, and a wireless router
- Laptops connect to vision server via wireless and control robots

Sample Camera View

- Camera view shows significant perspective distortion
- Three robot images selected from camera view show the difficulty of extracting color features (e.g. pink spots become almost white)
- Use of geometric features removes the colour calibration problem, but the same features useful to identify robots are themselves significantly distorted by the angle of the camera
- Doraemon uses Tsai camera calibration, selecting calibration points using a coloured carpet. User selects a coloured rectangle on the carpet and specifies the distance in the X and Y direction between the centres of rectangles of that colour. Even using a side view this results in errors of <1cm

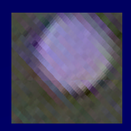
Interpolation Algorithms

- Interpolation: given a set of data points, find the value of the target function between these points
- For example, in the one-dimensional case given $x_1 \ldots x_n$ below, what is the value of the target function at x_t?



Square Pulse Interpolation

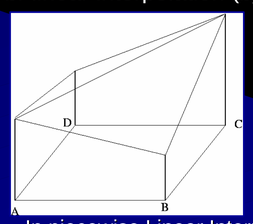
- Identical to nearest neighbour algorithm (1-nn) where each sub-scanned pixel is assigned the value of its closest neighbour
- computationally efficient but poor results – on our sample image, spot and robot top are extremely jagged, which also serves to enhance false colour info (yellow edges on robot, an artifact from errors in colour processing)



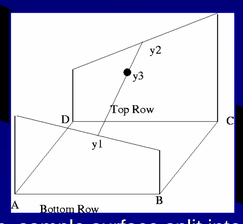
Bi-linear Interpolation

- Using triangle instead of square approximation is only slightly more complex
- Extending this to 2-d space (images) is problematic – four data points do not in general lie on an interpolation line
- This can be overcome by piecewise linear interpolation (a) or bi-linear interpolation (b)

A



B



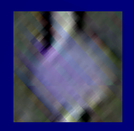
  - In piecewise Linear Interpolation, sample surface split into two planes (ACD and ABC). In the top left of the image, points are interpolated using ABC, while in the bottom right, ACD is used
  - in bi-linear interpolation: points are linearly interpolated along rows/cols & resulting points are then used to linearly interpolate the value of the target point
  - computationally efficient and better results in practice
- Bi-linear is computationally efficient and better results in practice – interpolation of robots from the sample image results in much rounder edges on spots and a less jagged robot shape:

Cubic B-Spline Interpolation

- Fits cubic b-splines to the data. This method results in smoother interpolation of the data, but uses a larger neighbourhood around the pixel to calculate the parameters of a cubic function – computational considerations normally restrict to a 4x4 pixel region
- Results of cubic b-spline interpolation are only slightly better than bi-linear, and computational considerations led us to restrict ourselves to 2 x 2 interpolation functions

Average Gradient Method

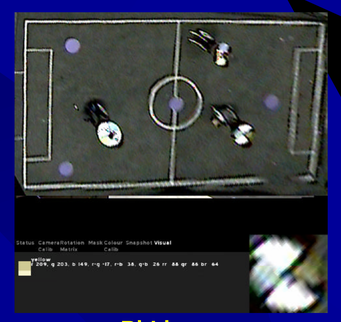
- Noise in an image often results in having to blur pixels – commonly, replacing each pixel with the average or median of pixels in its 4/9/16 neighborhood
- This loses contrast on the edges, which is an important recognition cue, making such simple methods unsuitable
- We developed a gradient blurring interpolation by computing averages along rows and columns, finding the closest point to the target point and then applying the average gradient at the starting point
- Results are similar to bi-linear and better than square /pulse interpolation:



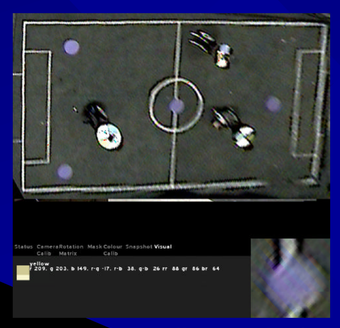
Evaluation

- Our method of evaluating the scene recognition performance of a computer vision system is to reconstruct the scene from an overhead view using these interpolation methods on sample images captured at an angle similar to the sample view on the left
- We tested all the interpolation methods above on 50 test images, and subjectively determined the accuracy of edges, area, and angles in the resulting overhead reconstructions. The following are samples:

Square Pulse



Bi-Linear



Average Gradient Method

- In all cases, performance of bi-linear, cubic b-spline, and our own gradient pulse interpolation was significantly better than that of square pulse interpolation
- Bi-linear interpolation was the most efficient of these, and so we are employing that in Doraemon
- The errors in the right edge of the playing field are due to errors in camera calibration – in most cases these are not significant because the field is locally still consistent (i.e. relative position of a robot and the ball is still correct
- Interpolating over the entire field is inefficient during a match, it provides evidence of the accuracy that can be achieved with a solid camera calibration and good interpolation methods
- No additional model knowledge has been used and robots are assumed to be flat on the playing field. The quasi-overhead view reconstruction could be improved by correcting for the known height of the robots in the image
- Currently able to maintain 30 fps when controlling 11 objects, but better optimization will improve this