Virtual Postcards

Project Checklist

6.111 Final Project - Fall 2007
Jessica Barber, Andrew Meyer

1. Essential Components

1.1 Real-Time Video Capture
- Buffer video input via ZBT
- Convert to YCrCb and then to HSV
- Interpolate or simple scale from 640x480 to 1024x768 Resolution

1.2 Color separation and identification
- Determine optimal YCrCb value to isolate fixed color index card
- Determine optimal HSV value to isolate fixed color index card
- Generate pixel mask containing only desired color
    - Filter mask to reduce/eliminate outliers

1.3 Corner Detection
- Approximate location of corner bounds of single index card
- Filter consecutive corner estimates to reduce noise and smooth result
- Output 4 2-tuples to image transform block (4 vertex locations)

1.4 Image Storage
- Utilize on-chip ROM for low resolution storage of one or more images
- Display Image from ROM at arbitrary translations/rotations on screen

1.5 Implement floating/fixed point mathematical ALU
- Must be capable of high decimal precision
- Square root, Divide, Multiply, Add/Subtract
- High Latency for Divide/Square root is OK since we only perform these operations once per vsync

1.6 Perform Vector Algebra and Normalization operations
- Takes 4 2d vertexes
- Generates 4 normalized vectors whose direction is normal to the edges of the detected index card (requires 4 square roots, 8 squares, 8 divides)

1.7 Visualize debugging information
- Display bounding lines of index card on 1024x768 screen at full 60Hz
- Separate the pixel space into Outside-car and Inside-card regions (using color to identify)

1.8 Overlay stored image onto the arbitrary detected quadrilaterial
- 10 multiplies, two divides, 10 summations (must be performed once per pixel!!!)

2. Integration

2.1 Combine overlay pixels with current (or partially buffered) video frame (from camera)
2.2 Display to screen via VGA interface

3. First Tier Improvements (Pending Extra Time)

3.1 Filter the overlay image to reduce aliasing along edges

3.2 Utilize FLASH memory or second labkit to allow for multiple image overlays

3.3 Further improve corner detection

3.4 Optimize overlay operations against video input (version 1 simply applies the current overlay to the NEXT frame, but we could buffer part of the current frame and then apply the current overlay)

4. Second Tier Improvements (Pending Extra Time)

4.1 Perform Alpha blending between video image and overlay image to allow smooth transparencies on either side

4.2 Attempt to handle multiple cards and occlusion (very ambitious)