Achieving Seamlessness in Multi-Projector based Tiled Display using Camera Feedback

Pranav Kant Gaur

Graphics and Visualization section

November 13, 2014

Agenda

1. Objective
2. Motivation
3. Challenges
4. Why not a commercial solution?
5. Approaches tried
6. Followed approach
7. System setup
8. Algorithm
9. Contributions
10. Results
11. Issues

Objective

We want to create a large image on the projection screen by combining images projected by multiple projectors.

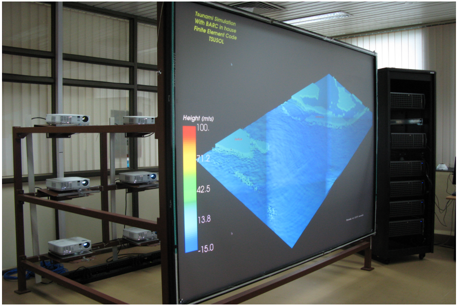


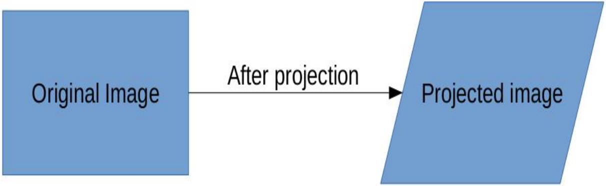
Figure: Multiprojector tiled display

Motivation

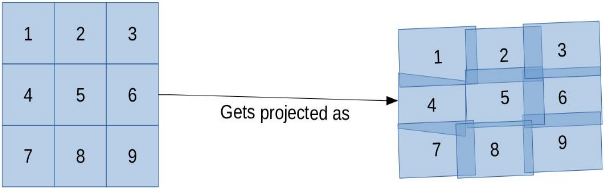
- Why *Tiled*?
  An image with spatial resolution higher than that *perceivable* by human eye cannot be visualized without reducing its resolution. We can achieve this by spatially *stretching* the content.
- Why *Multiprojector*?
  Seams of monitors used in our earlier Tiled display system were *distracting*. Projectors do not pose such limitation.

Challenges

1. Perspective distortion removal



2. Geometric continuity of projected content



3. Handling intensity in overlap region

Why not a commercial solution?

Our requirements:

- Configurable
- Portable
- Control over trouble-shooting

Limitations of current commercial systems:

- Software packages:
    - Limited support for cameras used for calibration.
    - Limited support for Graphics cards.
    - Not open source, some vendors provide API's only.
    - Do not work with Linux.

Commercial systems(contd.)

- Hardware + Software products:
  - Separate image processing box for generating seamless image.
  - Factory configured, support for configured is very limited.

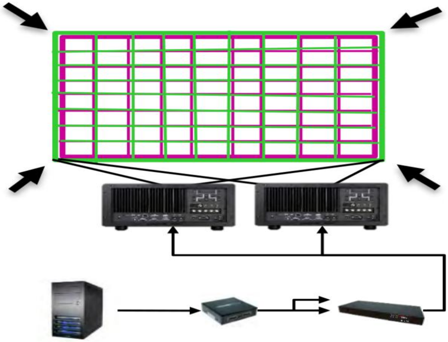


Figure: Commercial System, Image source: Optoma USA

Approaches tried

- Approaches
  - Fixed-projection-region
    - Iterative
      - Overlap
      - Hard-edge
    - Automatic
      - Hard-edge
      - Overlap
  - Arbitrary-projection-region
    - Automatic
      - Overlap

Followed approach

1. Geometric alignment
    1. For each projector, determine distortion introduced by it in the projected image on screen.
    2. Apply inverse of this distortion to the projector image to recover geometrically continuous projection across neighboring projectors.
2. Edge blending
    1. Determine overlapping region between neighbouring projectors
    2. Attenuate intensity of pixels in that region for all overlapping projectors so that their overlap does not create *bright* junction between them

System setup

Developed system has:

- 3X3 grid of projectors
- Rear projection screen
- 1 digital camera
- Workstations arranged in master-slave configuration

(a) System setup

(b) Projector-array behind the projection screen

Algorithm

- Improved the algorithm described in "PAPER NAME".
- Algorithm overview:
  1. Relation between Camera and screen coordinate system is determined.
  2. Each projector projects a checkerboard, camera captures the view and checkerboard corners are detected.
  3. For each projector, corresponding detected corners are addressed in a local bounding box.
  4. Global bounding box containing all local bounding boxes are computed.
  5. Overlap between adjacent projectors is determined.

Algorithm(contd.)

Compute screen to camera relation

- We want *Geometrically aligned* and *seamless* content on projection screen.
- Camera is just a *feedback device*.
- All later calculations are performed in screen-coordinate system.

Figure: View from camera

Algorithm(contd.): Remove perspective distortion

Project and detect features for each projector

Detected features are mapped to screen coordinate system.

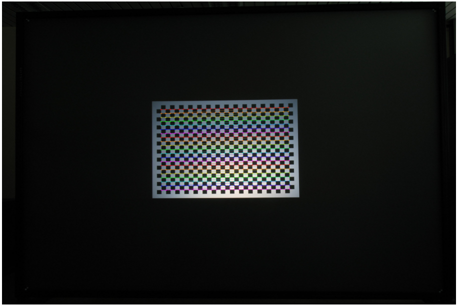


Figure: Low exposure image of detected features for the central projector

Algorithm(contd.): Remove perspective distortion

Compute *local* bounding boxes

Normalized pair of ($coordinate_{original\ image}$, $coordinate_{detected}$) for each checkerboard corner gives the *distortion* information.

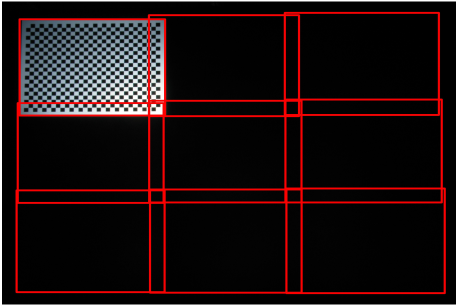
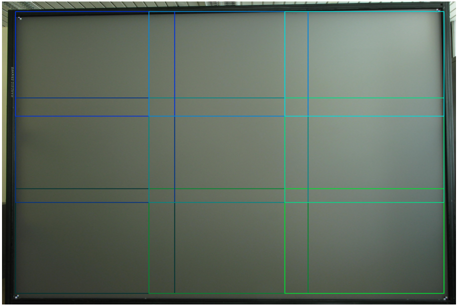


Figure: Boxes bounding the projection region of each projector

Algorithm(contd.): Geometric continuity

Compute *global* bounding box

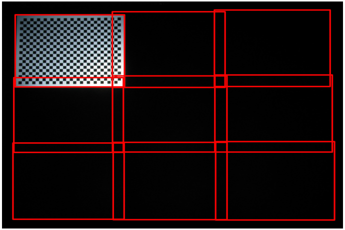
- Addressing all local bounding boxes wrt. a common coordinate system.
- Global bounding box represents the original image to projected on the screen.
- Helps in computing share of each projector in the original image.



Algorithm(contd.): Seamlessness

Compute alpha map

1. Compute region of overlap between adjacent projectors.
2. Attenuate image intensity of each overlapping projector.



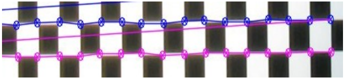
(a) Projection region

(b) Corresponding attenuation map

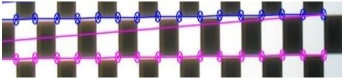
Contributions

Corrected detected corners using line fitting:

- Utilized the fact that collinearity under perspective projection is preserved.
- Fitted line on detected corners and corrected the detected coordinates using intersection of fitted lines.
- It resulted in more uniform texture mapping.



(c) Without line fitting

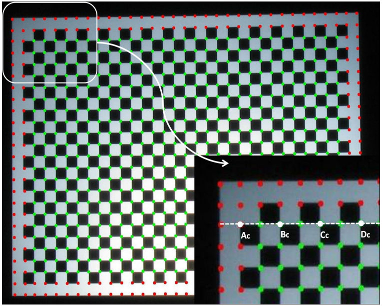


(d) With line fitting

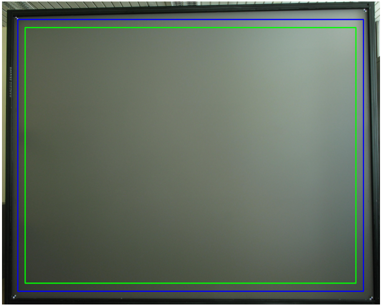
Contributions(contd.)

Used Cross-ratio invariant to recover full projection region:

- It is a perspective projection invariant.
- Relates 4 collinear points: $CR_p = \frac{|AC|_c * |BD|_c}{|BC|_c * |AD|_c}$



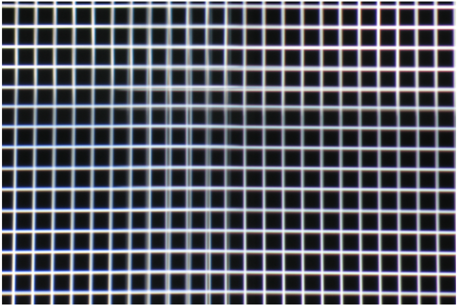
(e) Applying cross ratio invariant



(f) green: without cross ratio, blue: with cross ratio invariant

Results*

- Alignment procedure completes in 3-4 minutes as opposed to $\sim$ 30 mins. consumed in our earlier alignment approaches.
- Average misalignment between the grid lines projected at junction between neighbouring projectors was around $\sim 1.0mm$ on a grid size of $\sim$ 14mm.



Results(contd.)

- Recovered $\sim 10\%$ more projection region using cross-ratio invariant.
- Junctions between projectors are more imperceptible using cross ratio invariant.

(g) Without cross ratio: seams more visible

(h) With cross ratio: seams more imperceptible

- Resolution of the display is ~ 7 Megapixels.

Issues

- Blending function which can provide *completely* imperceptible seams in the overlapping regions is unknown.
- View dependant bright spot formation is still an *open* problem.

Figure: Open issues

Some snapshots

ADD FIGURES HERE!!!

Thank you!!

- Software:
  - Written in C
  - Dependent on OpenCV(v2.4.1) and libgphoto2(v2.5.2)
  - Works on Ubuntu(12.04 LTS) and Scientific Linux(6.1)
- Hardware:
  - 3X3 grid of NEC 200X DLP projectors
  - 2.4mX1.8m acrylic glass based rear projection screen(from ScreenTech,Germany)
  - Canon Powershot G7 digital camera
  - 4 Workstations(1 master+ 3 slave) each with Intel Xeon Six Core Processor X5670, NVIDIA Quadro FX 4600 Graphics card and 48 GB ECC RAM.