
# Feature detection: how it works, when to use it, and a sample implementation

Marco Jacobs

October 2nd, 2013

videantis
passion for video

# Computer vision: the challenge

- 50% of the brain is used for vision
- Body uses 100W
- Brain consumes 20W
- → about 10W for vision analysis

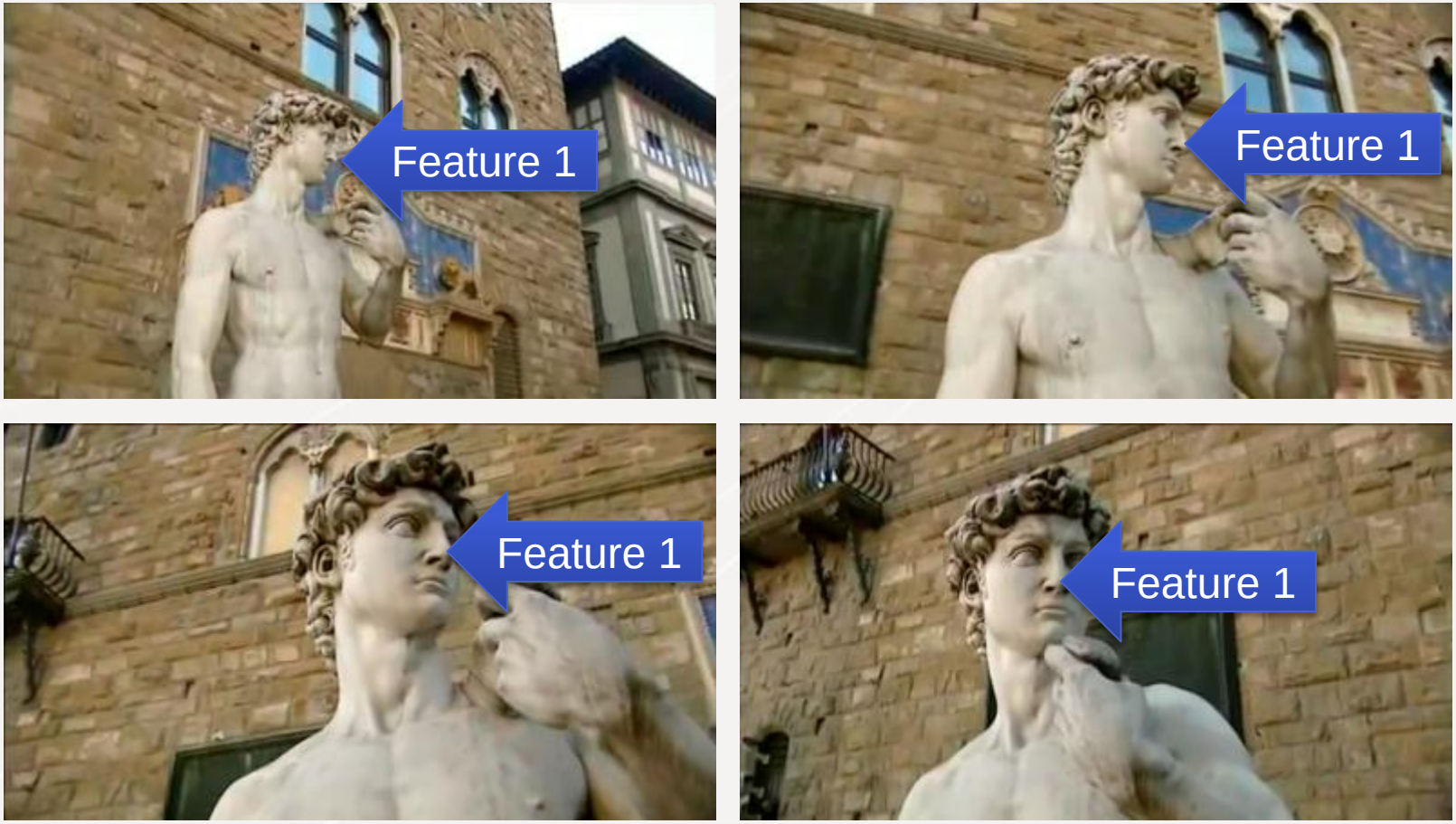
- Challenge: beat the human
- → Build machines that are faster, safer, cheaper, last longer, more accurate, etc.

# Feature detection & tracking: a key computer vision technique

- **Feature detection required for:**
  - Image alignment (e.g. panorama stitching)
  - Object recognition
  - 3D reconstruction (e.g. stereo cameras)
  - Motion tracking
- **Applications that depend on it:**
  - Automotive driver assist systems
  - Gesture interfaces
  - Augmented reality
  - Indexing and content-based retrieval
  - Robot navigation
  - Et cetera


Textbook: Computer Vision: Algorithms and Applications, Richard Szeliski

# What is feature detection / tracking?




*Recognizing the same feature in different images allows us to extract meaningful information*

# Feature detection, matching, tracking

**Detection** (e.g. Harris algorithm)
- Identify the interest points

1

Once every n frames

*find again*

**Description** (e.g. Harris algorithm)
- Extract a feature vector descriptor for each interest point (location, gradient, strength, color, etc.)

$$X_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

2

**Matching** **OR**
- Match correspondence between feature descriptors in 2 images (e.g. for stitching)

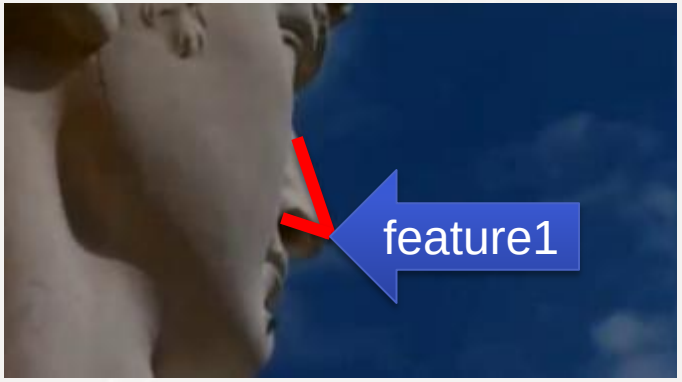
**Tracking** (e.g. Lucas-Kanade algo)
- Find the displacement of the feature (local search) – used for video


every frame

# Features: edges—why are they interesting?

- Sharp changes in image intensity are a key indicator of image content

- Extracting meaningful edges from images amounts to a dramatic reduction in the amount of data

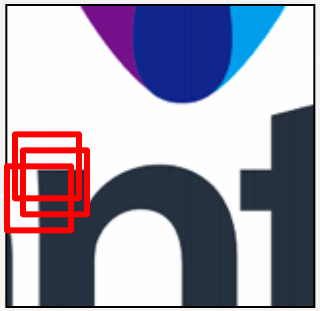


# Feature detection: Moravec

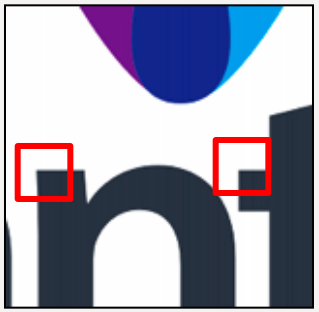
- H. Moravec, Carnegie-Mellon (1980)
- Start from a point, then move a window around
  - Calculate sum of squared difference between two regions
  - Little change: no good feature (flat region, or single edge)
  - Lots of change: single point that is a “corner”

**flat region**
No change in any direction

**edge region**
change in the edge direction



**corner**
change in all directions



**aperture problem**
considering only a small part of the image may produce ambiguous results

# Harris corner detection: Sobel

Harris and Stephens, Plessey Research, 1988

1. Edges in X direction (Sobel)

2. Edges in Y direction

Only use luma channel



1. Sobel in x
2. Sobel in y
3. Derivative calc
4. Box Filter
5. Harris calc
6. Max location
7. Threshold
8. Dilate
9. Select

Remaining steps:

Edges in two directions → we found a corner

# Harris

Calculate the **eigenvalues** of the (2x2) **gradient matrix M** from **image derivatives**

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

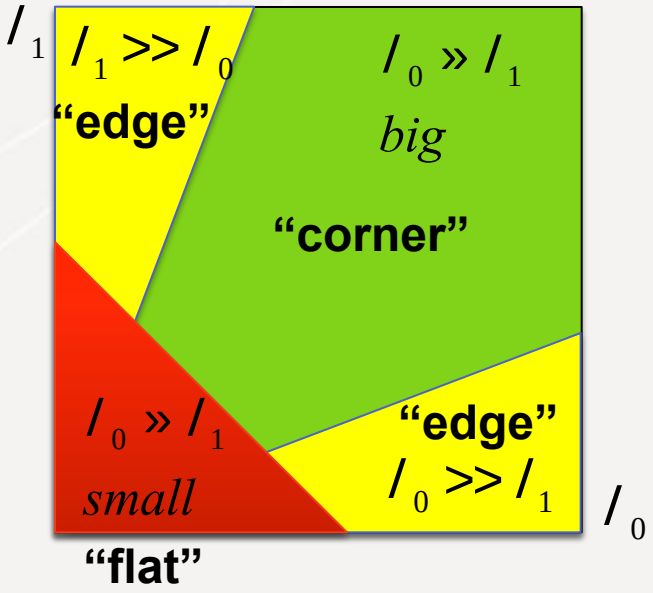
$$(I_0, I_1) = eigenvalues(M)$$

**“Corner strength” metric:**

- Harris and Stephens
  - $R = \lambda_0\lambda_1 - a(\lambda_0 + \lambda_1)^2$
- Shi-Tomasi (Kanade-Tomasi):
  - $R = min(\lambda_0,\lambda_1)$
- Triggs:
  - $R = \lambda_0 - a\,\lambda_1$
- Szeleski:
  - $R = (\lambda_0\,\lambda_1) / (\lambda_0 + \lambda_1)$

*Choose constant K empirically*

*R > K → found good feature*




1. Sobel in x
2. Sobel in y
3. Derivative calc
4. Box Filter
5. Harris calc
6. Max location
7. Threshold
8. Dilate
9. Select

# Selecting features

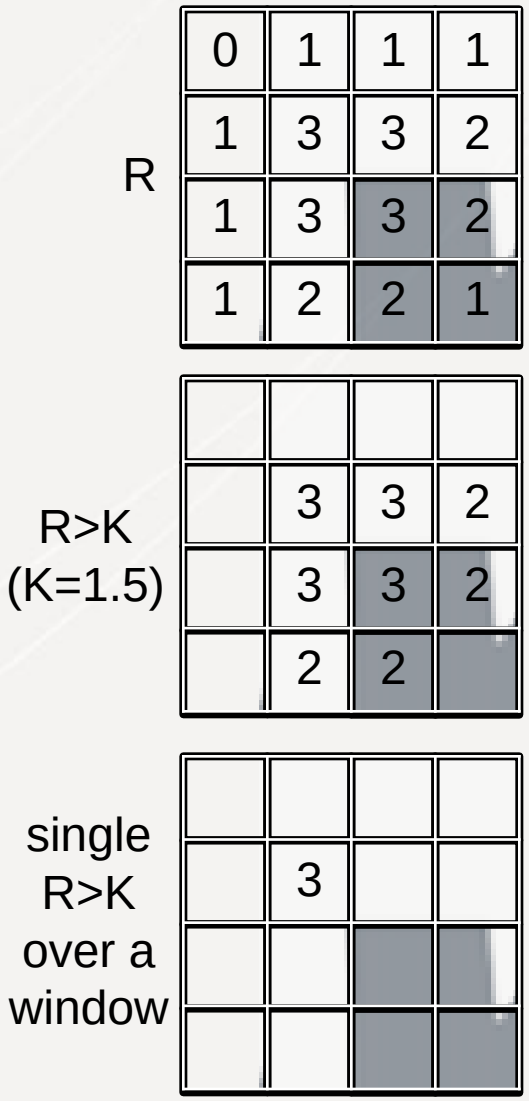
1. Max location
2. Threshold
3. Dilate
4. Select and store feature point

Select features with R > threshold K.

**Problem:**

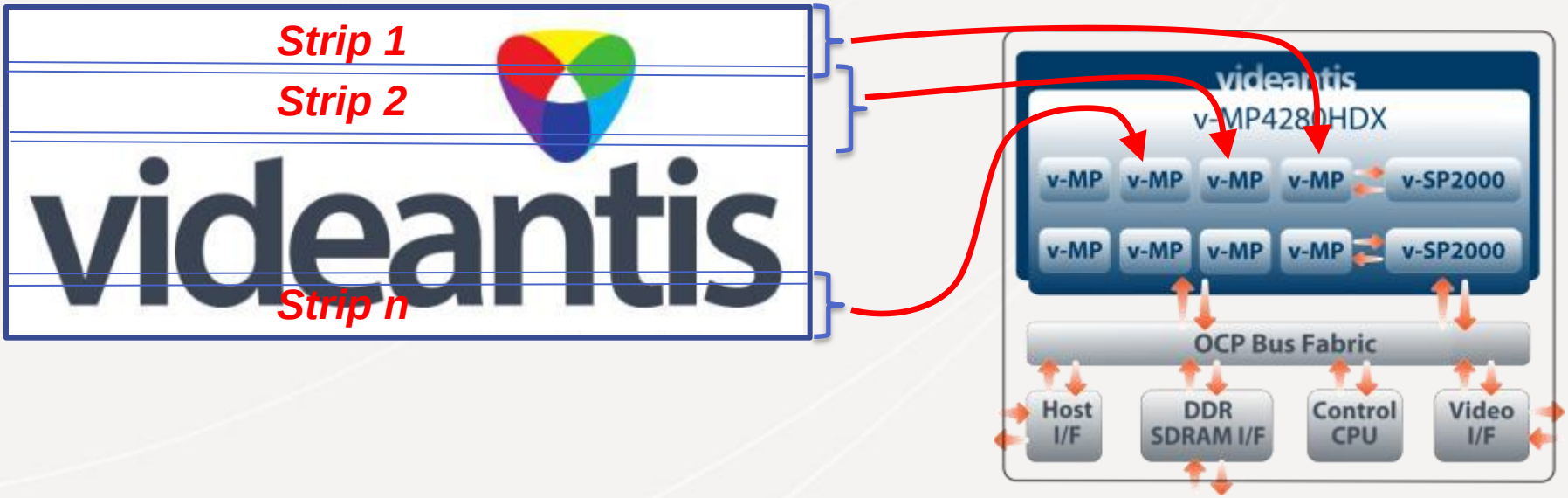
- For same corner, we find multiple feature points
- Can‘t distinguish features that are close to each other

Suppress these by selecting only the local maxima




1. Sobel in x
2. Sobel in y
3. Derivative calc
4. Box Filter
5. Harris calc
6. Max location
7. Threshold
8. Dilate
9. Select

# Parallelization on videantis processor




- Divide image into n strips (n is selectable at runtime)
- 2 pixels overlap between strips
- Each strip is processed on a different core
- E.g. for 1280x720 on 8 cores, 1280x94 strips per core

# Harris feature detect performance and power

- Clock speed 400 MHz @ 40nm (TSMC LP process)
- Feature detection in each image at 30 fps (each frame in video) or at 1 fps (for combination with tracking)

| Resolution | # of v-MP cores @30 fps | Core power @30 fps | Core power @1 fps |
|---|---|---|---|
| VGA | 1 at 400MHz | 5 mW | 0.16 mW |
| 720p | 3 at 400MHz | 15 mW | 0.5 mW |
| 1080p | 7 at 400MHz | 35 mW | 1.12 mW |

OpenCV original: all steps read in and write back to external DRAM memory

- 1280x720@30fps: 2.37GB/s
- The videantis-optimized version keeps all data in local memories
- 1280x720@30fps: 26.4MB/s (90x lower external bandwidth)

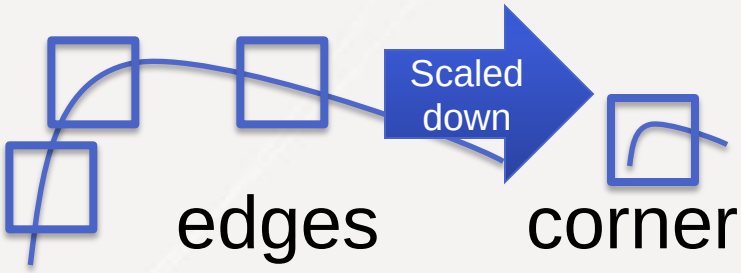
* all performance and power numbers are measured on real silicon

# Robustness

Harris corner detection can handle:

- Image rotations
- Small scale differences

Doesn't work at bigger scale differences




Two well-known algorithms try to overcome this by using a pyramid of scaled images:

- SIFT (Scale-invariant feature transform)
- SURF (Speeded Up Robust Features)


Other important algorithms:

- HOG (Histogram of oriented gradients)
- ORB (Pyramidal Harris with binary intensity tests)

# Feature detection, matching, tracking

**Detection** (e.g Harris algorithm)
- Identify the interest points

1


Once every n frames

*find again*

**Description** (e.g. Harris algorithm)
- Extract a feature vector descriptor for each interest point (location, gradient, strength, color, etc.)

$$X_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

2

**Matching** **OR**
- Match correspondence between feature descriptors in 2 images (e.g. for stitching)

**Tracking** (e.g. Lucas-Kanade algo)
- Find the displacement of the feature (local search) – used for video


*match*

*track*

every frame

# Pyramidal Lucas-Kanade algorithm (aka Kanade-Lucas-Tomasi)

- Purpose:
  - Find the movement of features in successive images of a scene.
  - We call this movement the optical flow (i.e. an [x,y] vector for each feature)
- Algorithm:
  - Build pyramid levels of image n, n+1
  - Foreach feature to be tracked
    - Foreach pyramid level
      - Calculate gradient matrix
      - For 1..K (or error<threshold)
        - Calculate difference error
        - Use gradient matrix to calculate next location
        - Found flow vector estimate
      - Initial guess for next level
    - Found final flow vector for this feature


Image n


Image n+1


Optical flow: Find v

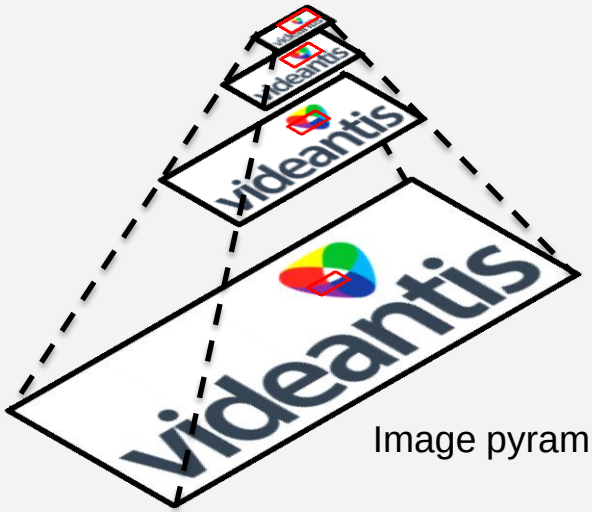
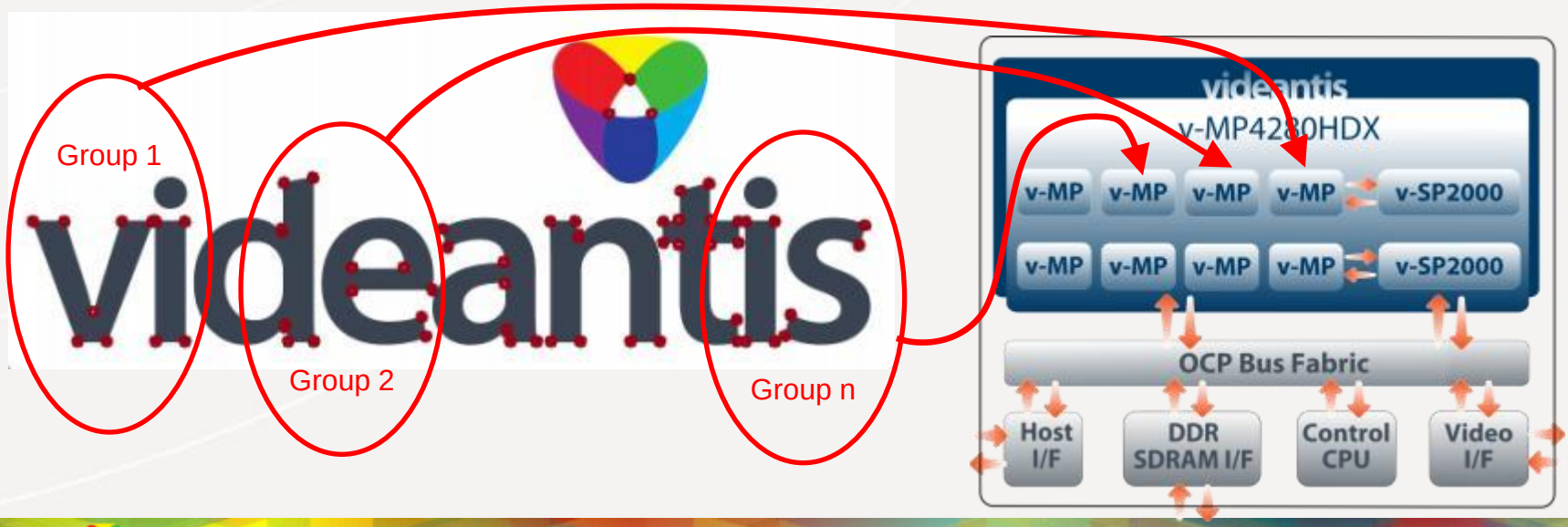



Image pyramid

# Lucas-Kanade Parallelization

1. Build image pyramids – each processor works on strips
2. Track features:
   - Divide feature list into n groups (where n is number of cores)
   - Each group of features is processed on a separate core.
   - Algorithm doesn‘t map well to wide SIMD processors, but does map well to multicore architectures (with close to linear speedups)



# Lucas-Kanade performance

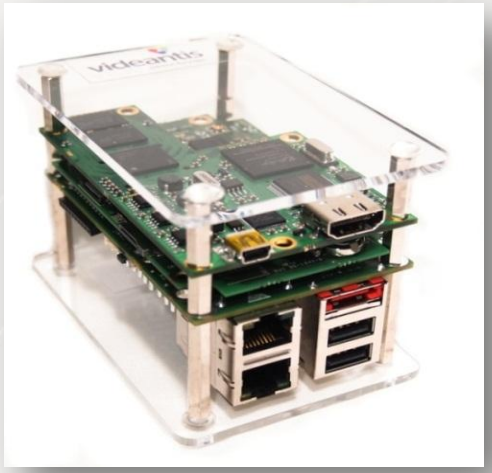
- Processing steps
  - Pyramidal downscale (3 levels)
  - SobelX/Y on all levels
  - Search window 8x8 for LK
- Performance and power figures
  - Clock speed 400 MHz @ 40nm TSMC LP
  - Frame resolution/rate: 1280x720 @ 30fps

| # of v-MP cores vision subsystem | Core power | Trackable Harris features |
|---|---|---|
| 2 | 10 mW | >350 |
| 4 | 20 mW | >1000 |
| 8 | 40 mW | >2350 |

* all performance and power numbers are measured on real silicon

# Conclusions

- Harris corner detection is a key feature detector
  - Many newer algorithms use parts of the Harris algorithm
- The Lucas-Kanade algorithm can track the detected features
- The algorithms can be implemented efficiently at high resolution while consuming low power on the videantis parallel signal processor



*Please drop by our booth for a real-time demonstration*

# Resources

- Building Machines That See: Finding Edges in Images, Eric Gregori, BDTI
  - http://www.embedded-vision.com/platinum-members/bdti/embedded-vision-training/documents/pages/building-machines-see-finding-edges-i
- "Embedded Lucas-Kanade Tracking: How It Works, How to Implement It, and How to Use It" by Goksel Dedeoglu, Texas Instruments
  - http://www.embedded-vision.com/summit/oct2013/presentations
- UCF Computer Vision Video Lectures 2012, Dr. Mubarak Shah
  - http://www.youtube.com/playlist?list=PLd3hlSJsX_Imk_BPmB_H3AQjFKZS9XgZm

# Questions?