Embedded Lucas-Kanade Tracking: How it Works, How to Implement It, and How to Use It

Goksel Dedeoglu

October 2nd, 2013

Contributors for Algorithms, Optimization, and Prototypes

Goksel Dedeoglu

Andrew Miller (UMD)

Understanding Motion: Sensors, Algorithms, Applications

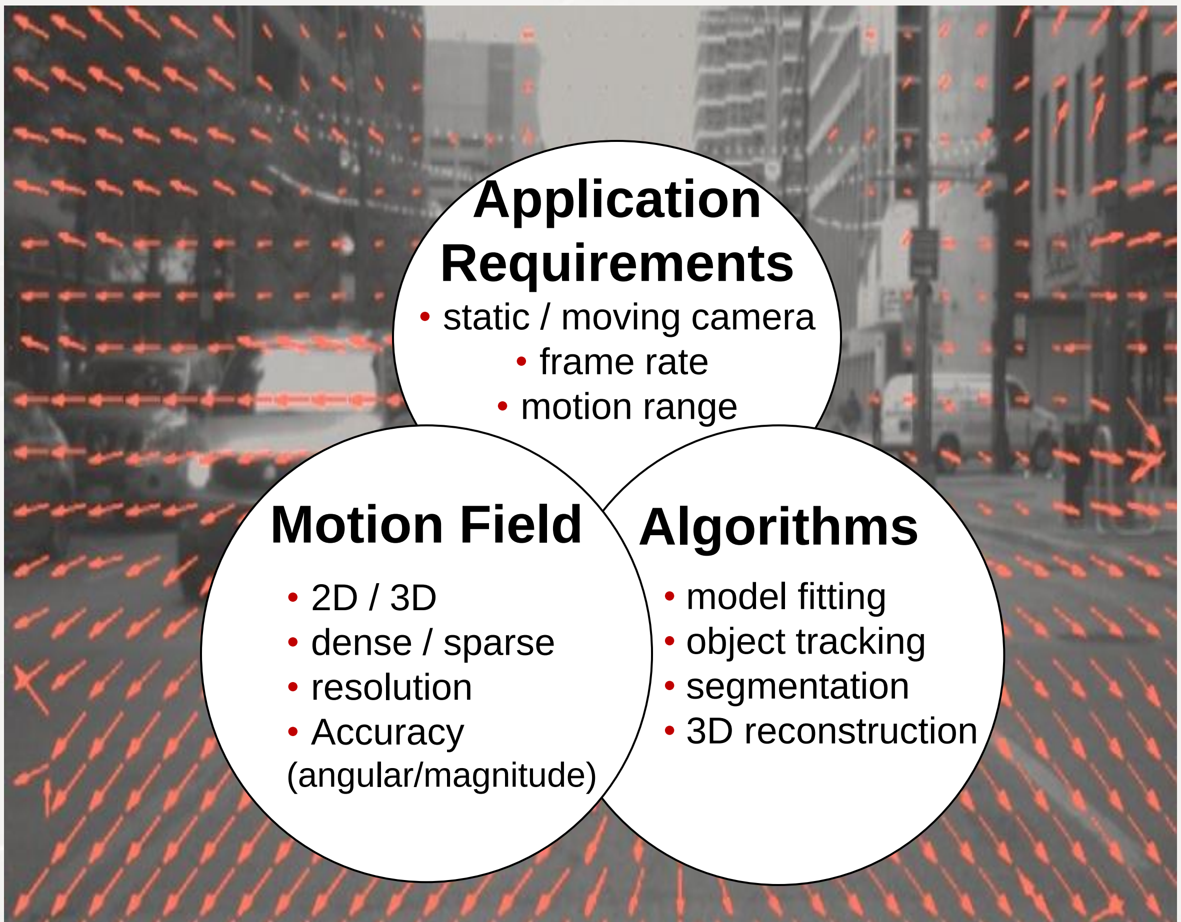
embedded VISION SUMMIT

Camera (visual)

Inertial Sensor

Range Sensor

GPS



Automotive

- cross-traffic alert
- collision avoidance
- parking assist

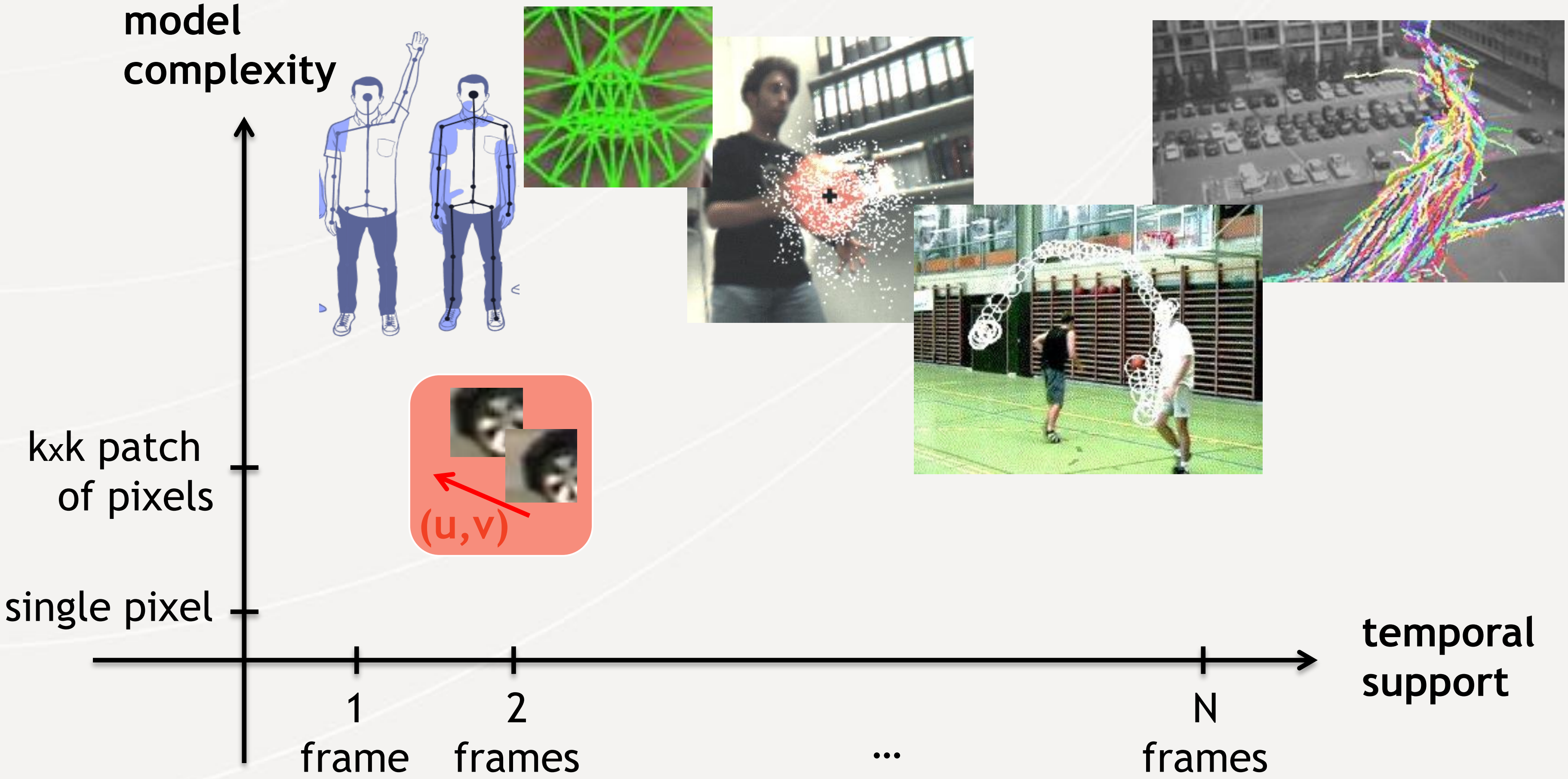
Video Security

- crowd analysis
- action recognition
- traffic analysis

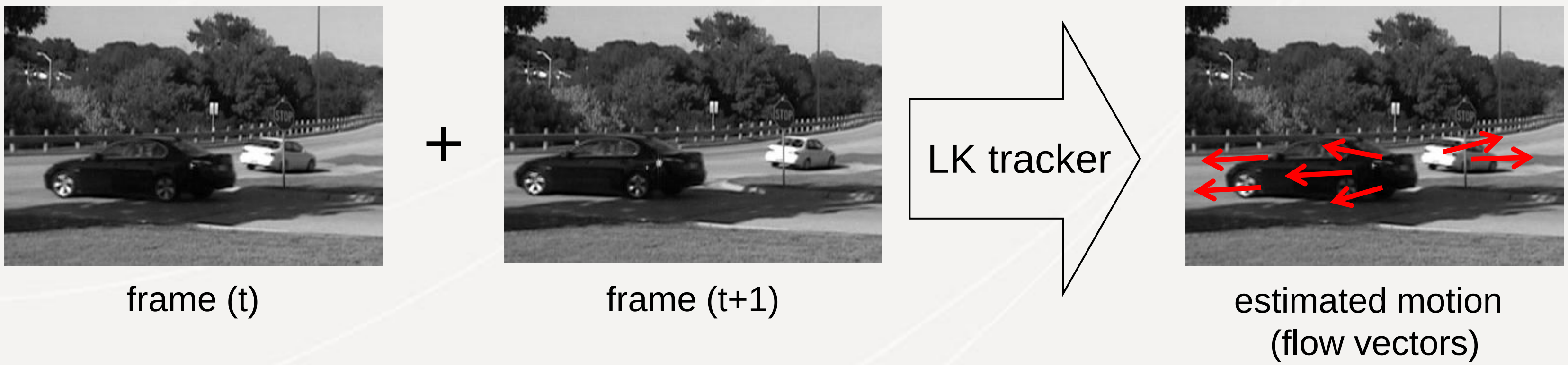
Human-Device Interaction

- gesture recognition
- sign recognition
- facial expression analysis

Putting the Lucas-Kanade Tracker on the map



Lucas-Kanade Estimates Motion Between Consecutive Frames



- *"An Iterative Image Registration Technique with an Application to Stereo Vision"*, Bruce Lucas and Takeo Kanade, published in 1981.
- Tested and proven over 30+ years in practical applications
  - There exist generalizations to more complex object & motion models
  - There exist much-simplified versions that work for very small displacements
- Good understanding of how & when it works well
  - "Good Features to Track", Jianbo Shi and Carlo Tomasi, published in 1994

Understanding Lucas-Kanade

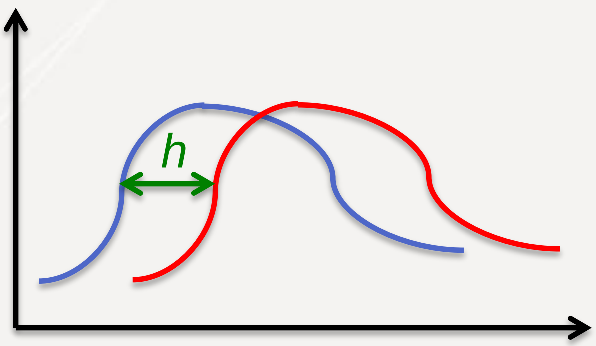
- Assumption: brightness constancy

$$F(x+h) = G(x)$$

frame (t)

displacement
to be estimated

frame (t+1)



- Underdetermined system of equations; additional constraints needed.

- Assuming the flow is constant in a small neighborhood of pixels, estimate the displacement (optical flow) vector h by minimizing

$$\mathrm{E} = \sum_{\substack{x \\ k\mathrm{x}k}} \left[F(x+h) - G(x)\right]^2$$

The Lucas-Kanade Tracking Algorithm

1. The objective function is $$\mathrm{E} = \sum_{\substack{x \\ \text{kxk}}} \left[F(x+h) - G(x)\right]^2$$

2. Linearize $$\mathrm{E} = \sum_{\substack{x \\ \text{kxk}}} \left[F(x) + h\frac{\partial F}{\partial x} - G(x)\right]^2$$

3. Compute the least-squares solution for the displacement *h*

$$\begin{bmatrix} h_x \\ h_y \end{bmatrix} = \begin{bmatrix} \sum F_x^2 & \sum F_x F_y \\ \sum F_x F_y & \sum F_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum F_x (G - F) \\ \sum F_y (G - F) \end{bmatrix}$$

4. Iterate until convergence!

When does Lucas-Kanade work best?

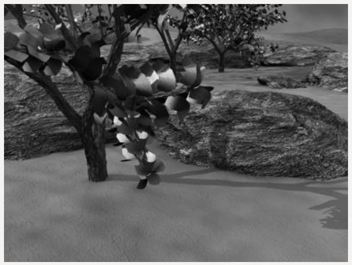
- When image regions are "textured"

$$\begin{bmatrix} h_x \\ h_y \end{bmatrix} = \begin{bmatrix} \sum F_x^2 & \sum F_x F_y \\ \sum F_x F_y & \sum F_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum F_x (G - F) \\ \sum F_y (G - F) \end{bmatrix}$$

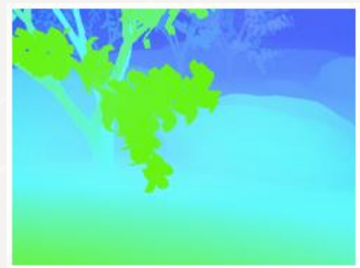
- When displacements are small
  - The higher frame-rate the sensor, the better!
  - At the expense of increased computation, more robust expected with faster numerical convergence
  - Common remedy: multi-resolution pyramids

How to chose typical parameters

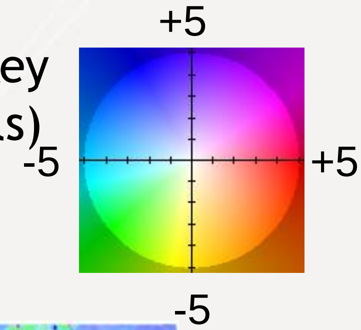
Scene

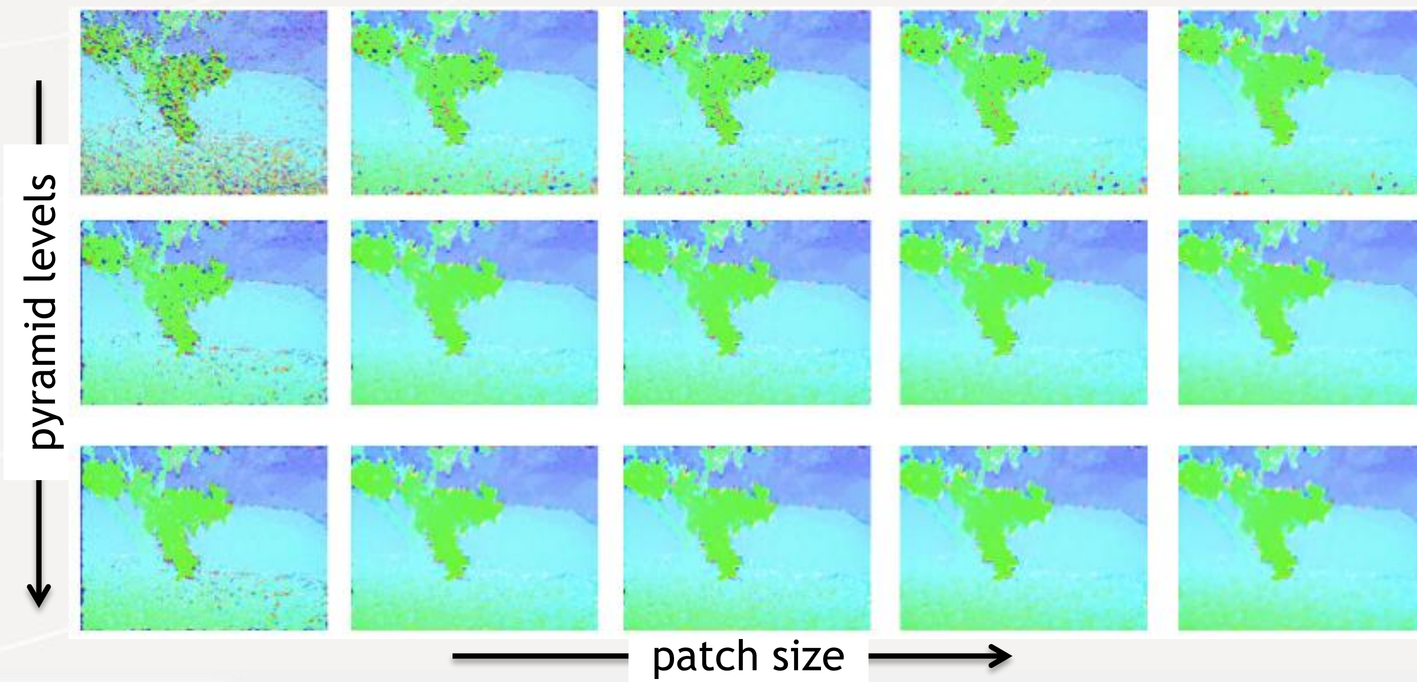


Ground Truth Motion

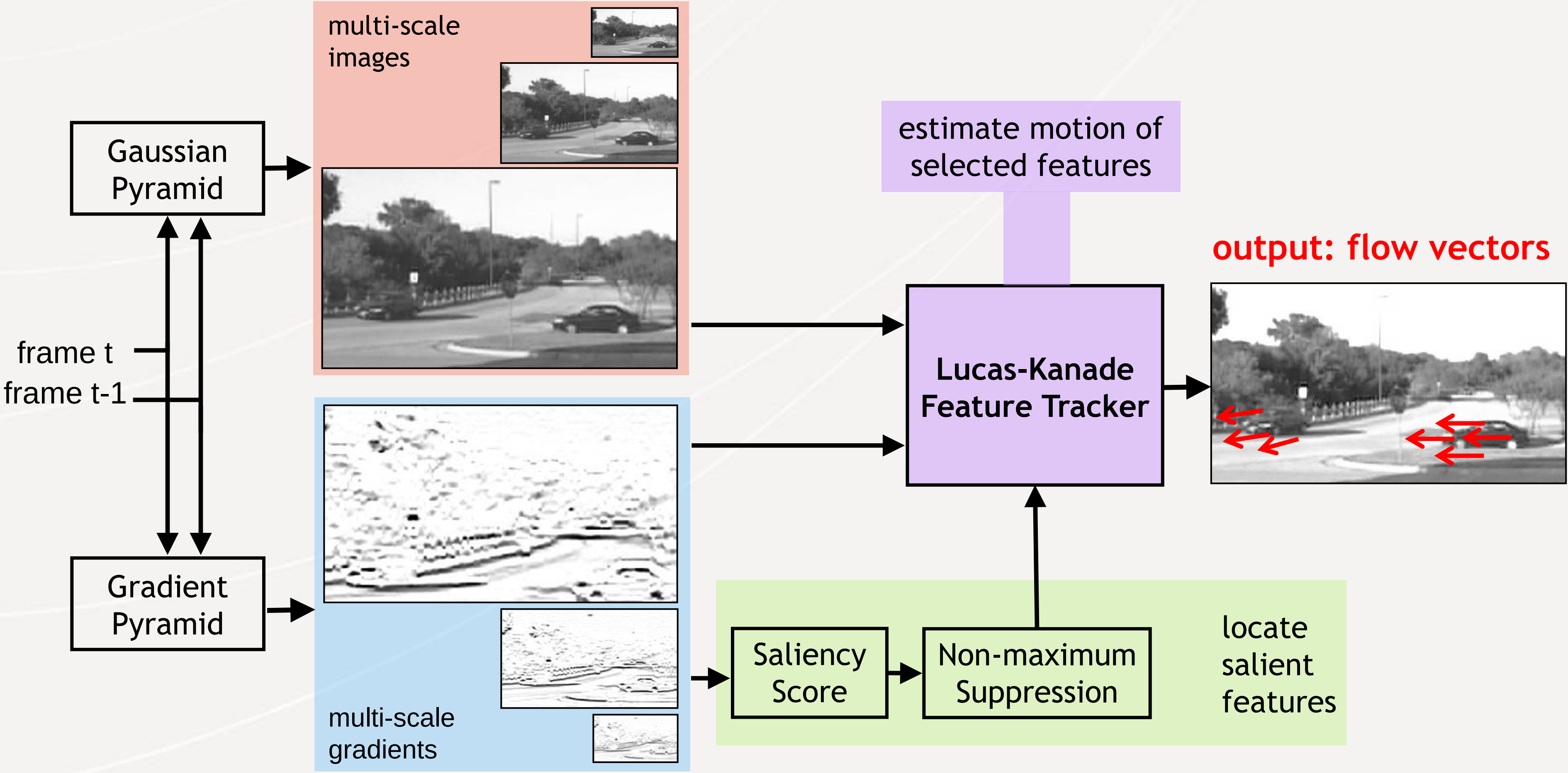


Motion Key (in pixels)

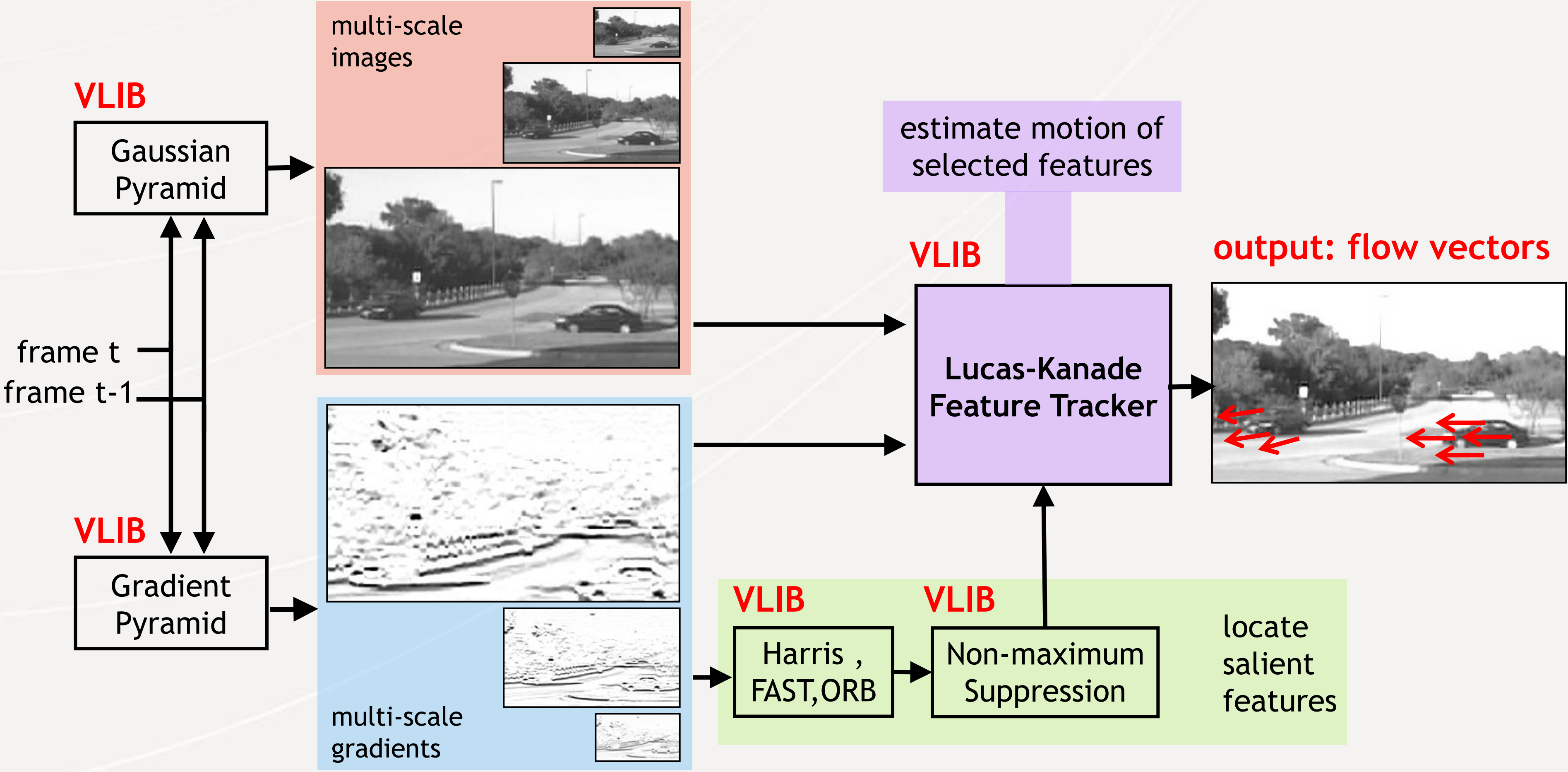




Typical Lucas-Kanade Tracking Pipeline



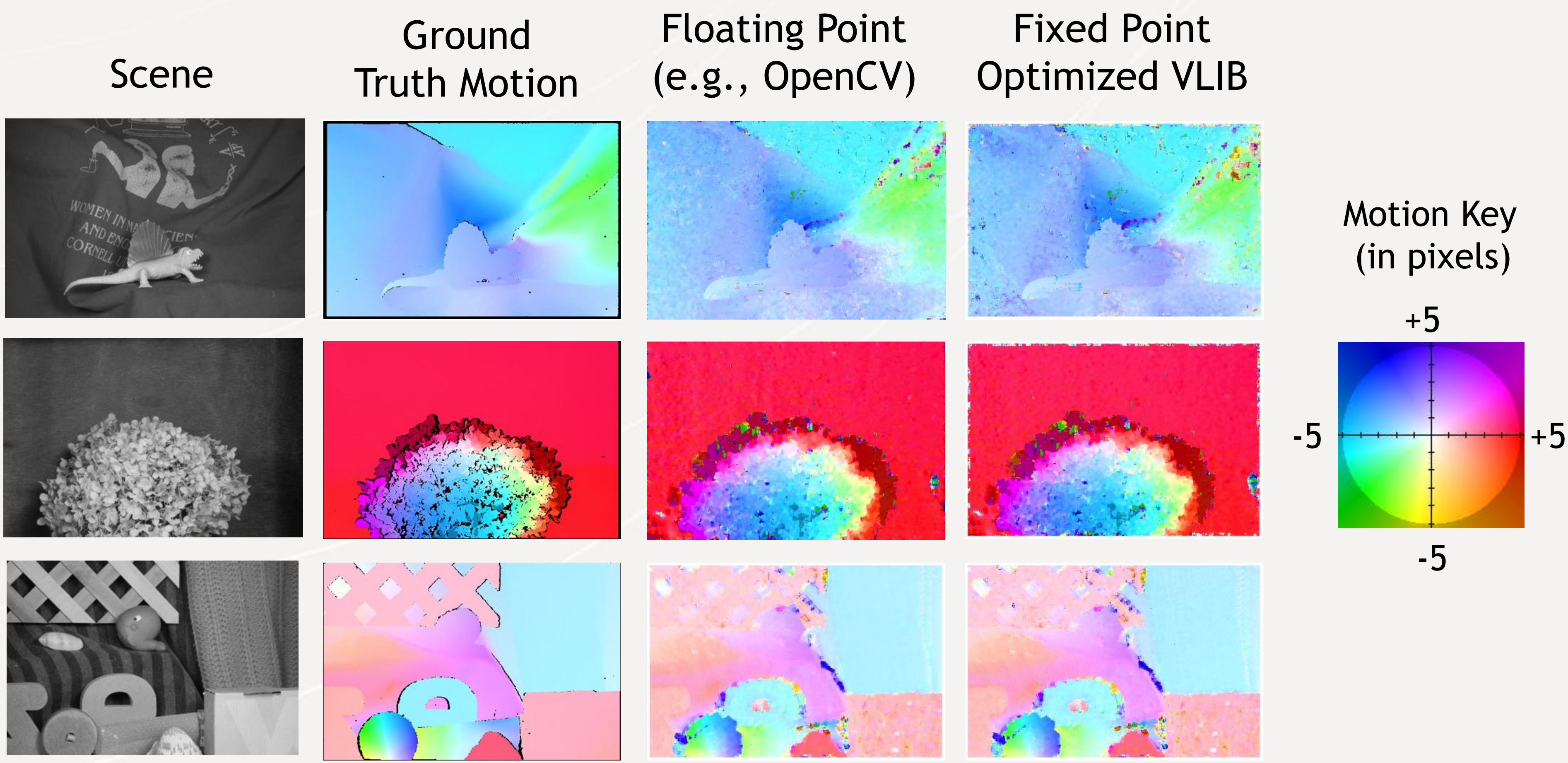
Lucas-Kanade Tracking with TI's Vision Library VLIB



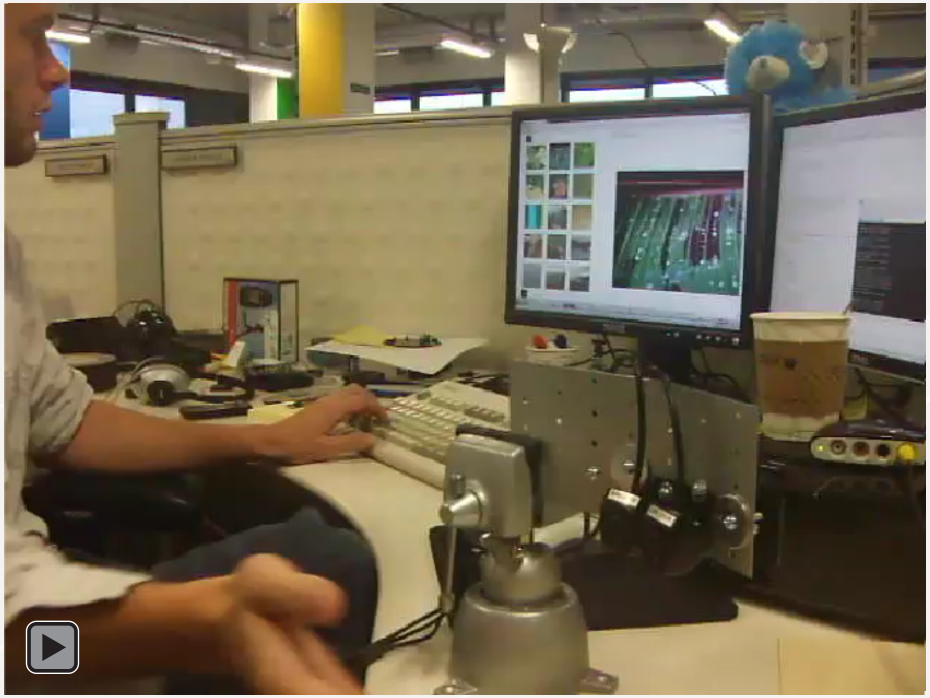
The API for the Lucas-Kanade Tracker in VLIB

```
S32 VLIB_trackFeaturesLucasKanade_7x7_track_error(
        const U08 * restrict im1,
        const U08 * restrict im2,
        const S16 * restrict gradX,
        const S16 * restrict gradY,
        S32 width,
        S32 height,
        S32 nfeatures,
        const S16 x[],      // SQ11.4
        const S16 y[],      // SQ11.4
        S16 outx[],         // SQ11.4
        S16 outy[],         // SQ11.4
        S32 max_iters,
        const U08 * restrict scratch_klt,
        U32 track_error[],
        U08 patch)
```

Studying the Fixed-Point Approximation



Prototype: Gesture Recognition

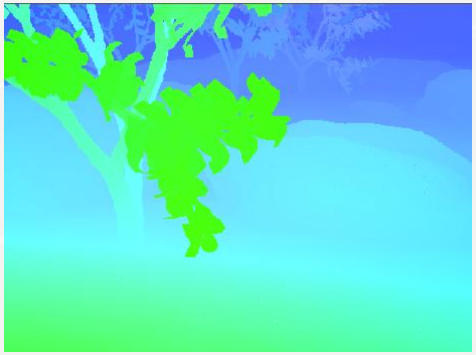


navigate to select image

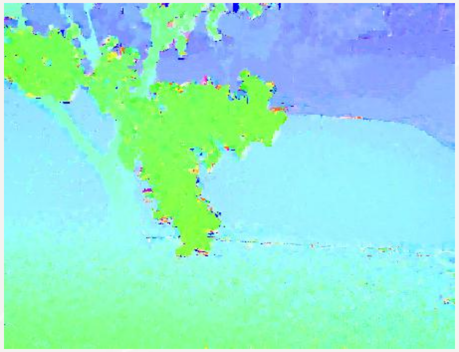
rotate image left / right

Quality Control for “Lucas-Kanade”

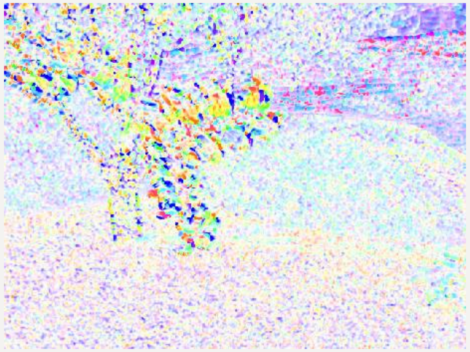
- Computer Vision is evolving, with Open Source libraries expanding fast, but with limited scrutiny & quality control. There exist two “Lucas-Kanade” optical flow functions in OpenCV:



ground truth
optical flow



calcOpticalFlowPyrLK
pyramid-based
good results
“LK-brand”



cvCalcOpticalFlowLK
single-step algorithm
can be **100x** faster
handles **very** small motion
now **obsolete**

- Embedded developers & architects beware!

Conclusions

- Lucas-Kanade is a well-understood & widely deployed method for tracking feature points
- We have implemented an embedded Lucas-Kanade tracker on the DaVinci DM6437 SoC; APIs are available in TI's Vision Library VLIB
- Three key messages:
  - Advantages: tested & proven over 30+ years, works reliably in textured image regions and small motion vectors
  - Challenge: computationally demanding, algorithmic extensions continue
  - With the right programmable processor and careful design trade-offs, Lucas-Kanade can be implemented with cost & power consumption suitable for embedded systems.

Resources for Further Investigation

- *"An Iterative Image Registration Technique with an Application to Stereo Vision"*, Bruce Lucas and Takeo Kanade, Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), April, 1981, pp. 674-679. (URL)
- *"Lucas-Kanade 20 Years On"*, project lead by Simon Baker & Iain Matthews at the Robotics Institute of Carnegie Mellon University (URL)
- *"Determining Optical Flow"*, Berthold Horn and Brian Schunck, Artificial Intelligence, vol. 17, pp. 185-203, 1981.
- For related algorithms, benchmarks and datasets, see
  - Middlebury dataset: http://vision.middlebury.edu/flow
  - KITTI dataset: http://www.cvlibs.net/datasets/kitti