A Generic Framework For Mobile Video Analysis

Matthieu Garrigues <matthieu.garrigues@gmail.com>
ENSTA-ParisTech

EUREKA-ITEA2 SPY Project [1], funded by the French Ministry of Economy.

May 10, 2012

Introduction: $2D$ vs real time $2D + t$

$2D$ image processing

- Less constraints on processing time per pixel.
- All computation are done on only one image.
- Get the most of each pixel.

Real time $2D + t$ image processing

- Input throughput: $211 Mb/s$ for a 30fps $640 \times 480 \times$ 24bits video stream.
- Information spreads spatially **and temporally**.
- Knowledge must accumulate over the time.
- **Problem: How to merge computations done at time t and $t + 1$ in case of moving objects and moving camera? How and where do we store them?**

Outline

Tracking?

- The classical 2D grid of pixels is not accurate enough for video analysis.
- We need a low level intermediate structure to build algorithms that execute incrementally over the frames.

Semi-Dense?

Advantages of a high density point tracking

- Neighbor points are close in memory.
- Fast spatial incoherences filtering.
- Allow robust voting algorithms (example: Stabilization)

Particles

Particle Definition

- A point with coordinates in the image plane.
- Move with the object O projected on the same coordinates.
- Live as long as O is visible.

Requirements for a quasi-dense particle field

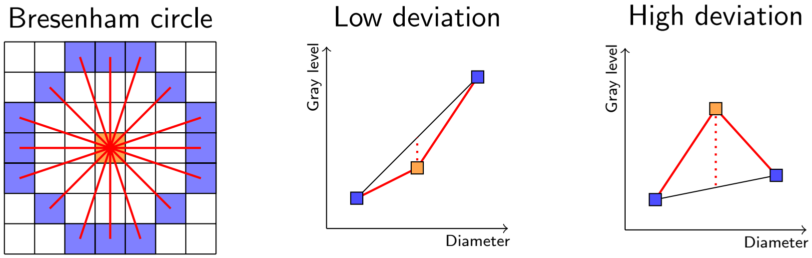
- Very low memory footprint per particle.
- Very fast particle matching.

Keypoint Selection: Goals

- Select a maximal number of points that contain **just enough** information for tracking.
- Discard homogeneous areas and straight contours.
- Contrast invariant.

Keypoint Selection: Saliency

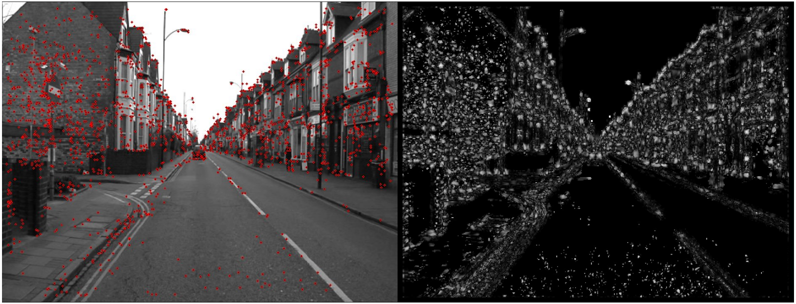
We define **the saliency of pixel** p as the minimal deviation from linearity along each diameter of the Bresenham circle of radius 3 centered on p. Normalization by local variation provides contrast invariance.



Keypoint Selection: Particle Creation Criteria

We add new particles where:

- Local contrast is above a given threshold.
- No particle already lives in the 3x3 neighborhood.
- No pixel has higher saliency in the 3x3 neighborhood.

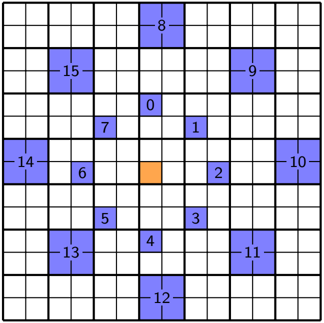


Good (enough) Feature to Track

We sample 16 gray level values (128bits vector) on the Bresenham circle of radius 3, at scale s and $s - 1$.

Why a very basic and non robust descriptor is enough?

- High frame rate $\Rightarrow$ small appearance changes between frames.
- Prediction reduces the number of matching candidates.



2-scale particle descriptor

Particle Position Prediction

Position prediction advantages

- Reduces the number of matching candidates.
- Less candidates to discriminate $\Rightarrow$ Smaller matching descriptor.
- Reduces false matching probability (if the prediction is good).

Particle Position Prediction

Goal: Match particles with pixels of the incoming frame. Need to predict particle shift between frame t and $t+1$.

Particle speed

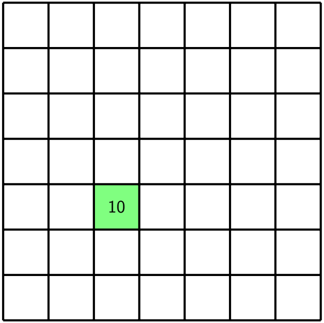
Sum of 2 components:

- **Projected object speed** (*p.speed*): Highly predictable. Since we process video at high frame rate, we consider particle speed constant between t and $t+1$.
- **Movement due to camera pan and tilt** (*cmov*): can be large and unpredictable.

$$p.pos_{t-1} = p.pos_t + p.speed \times \Delta t + cmov$$

Matching Algorithm

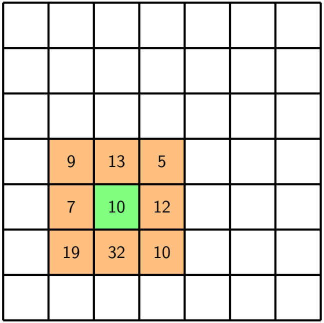
- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.



On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Matching Algorithm

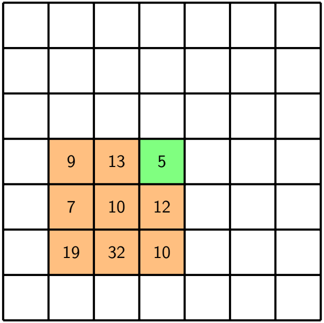
- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.



On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Matching Algorithm

- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.



On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Matching Algorithm

- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.

		15	5	7		
	9	13	5	4		
	7	10	12	8		
	19	32	10			

On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Matching Algorithm

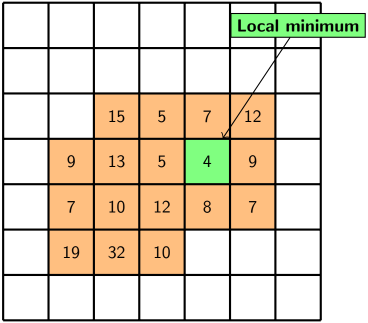
- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.

		15	5	7		
	9	13	5	4		
	7	10	12	8		
	19	32	10			

On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Matching Algorithm

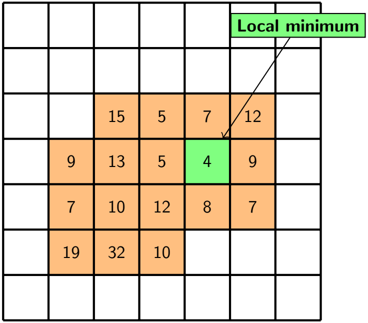
- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.



On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Matching Algorithm

- **Goal**: Correct prediction errors.
- **Algorithm**: Iterate until there is no 3x3 neighbor closer in the feature space.



On average, 2.5 iterations before convergence to a local minimum (Camvid dataset).

Coarse to Fine Camera Pan-Tilt Estimation

- Camera pan-tilt estimation ($cmov$) at scale s helps prediction at scale $s + 1$.
- We start with $cmov \leftarrow [0, 0]$ at the coarsest scale.
- Each particle votes for a specific $cmov$.

Performance on GPU and CPU

Test video

- Camera embedded in a car.
- 480x360 pixels.
- 4500 alive particles in average.
- More results: `http://ensta.fr/~garrigues/r/particletracking`

CPU implementation

- One thread per CPU core.
- Intel 4cores I5 3.3GHz.
- 150 frames per second.

GPU CUDA implementation

- Thousands of threads.
- Geforce 280 GTX.
- 250 frames per second.

Outline

A C++ Building Block for Mobile Video Analysis

Goals

- Make the tracking algorithm a generic building block for video analysis.
- Allow algorithms to store results directly inside particle memory space.
- For CPU and CPU+GPU architectures.

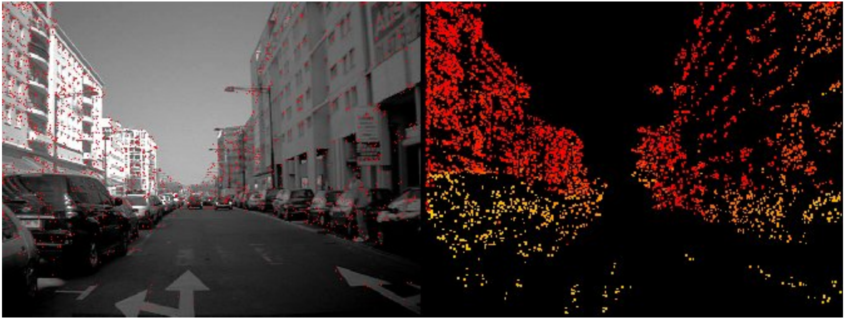
Framework API

```
// User defined particle attributes.
struct my_attributes
{
  unsigned label;
  float depth;
};

// Tracker instantiation.
semi_dense_tracker<[GPU|CPU], my_attributes> tr;

videostream vid(argv[1]);
host_image2d<uchar3> img(vid.domain());

while (not vid.finished())
{
  vid >> img;

  // Tracking.
  tr.update(img);

  // Access and update particles attributes at finest scale.
  for (const auto& p : tr.particles(0))
  {
    float new_depth_estimate = user_depth_estimation(p);
    p.usr_attr.depth = new_depth_estimate;
  }
}
```

Outline

Depth Estimation

- Estimate particle relative depth when the camera moves along the optical axis.
- Depends on focus of expansion estimation.
- Relative to camera speed and focal distance.
- Results: `http://ensta.fr/~garrigues/r/relative_depth`



Stabilization

- Pan / tilt correction only.
- Each particle votes for camera translation in a 2d histogram.
- We filter large camera accelerations.
- Results: `http://ensta.fr/~garrigues/r/stabilization`

Outline

Future works

- Robust and fast extraction of focus of expansion.
- Improve stabilization.
- Improve depth estimation.
- Real time approximative 3D reconstruction.
- Detection and tracking of moving objects.
- Combine all of those to help semantic scene parsing.

Conclusion

- We propose a framework for mobile video analysis.
- For CPUs (with OpenMP) and Nvidia GPUs (with CUDA).
- Target real time video improvement and understanding.

Thank you for your attention.
Any questions?

References I

[1] EUREKA-ITEA2.
SPY project.
http://www.itea2-spy.org.

[2] M. Garrigues and A. Manzanera.
Real time semi-dense point tracking.
In *International Conference on Image Analysis and Recognition (ICIAR'12)*, pages 245–251, Aveiro, Portugal, June 2012.