

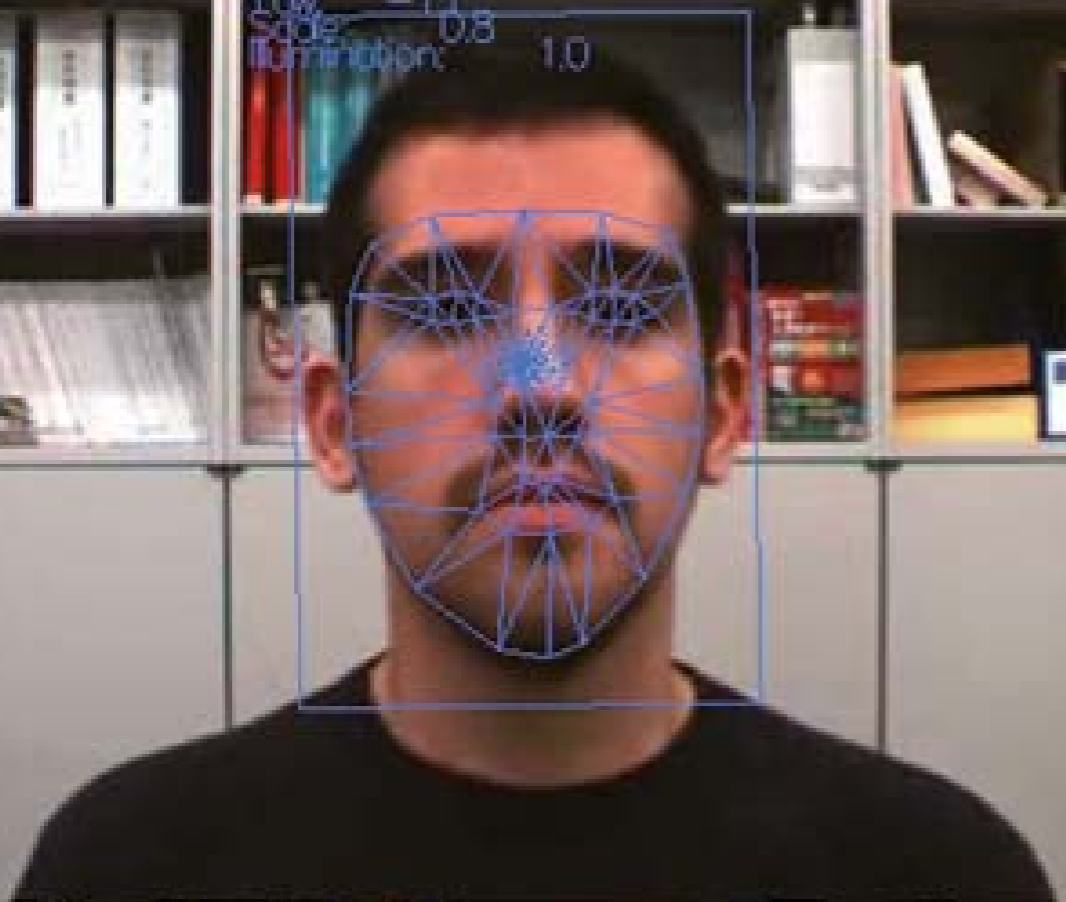
# REAL-TIME VISUAL TRACKER BY STREAM PROCESSING

SIMULTANEOUS AND FAST 3D TRACKING OF MULTIPLE FACES IN  
VIDEO SEQUENCES BY USING A PARTICLE FILTER

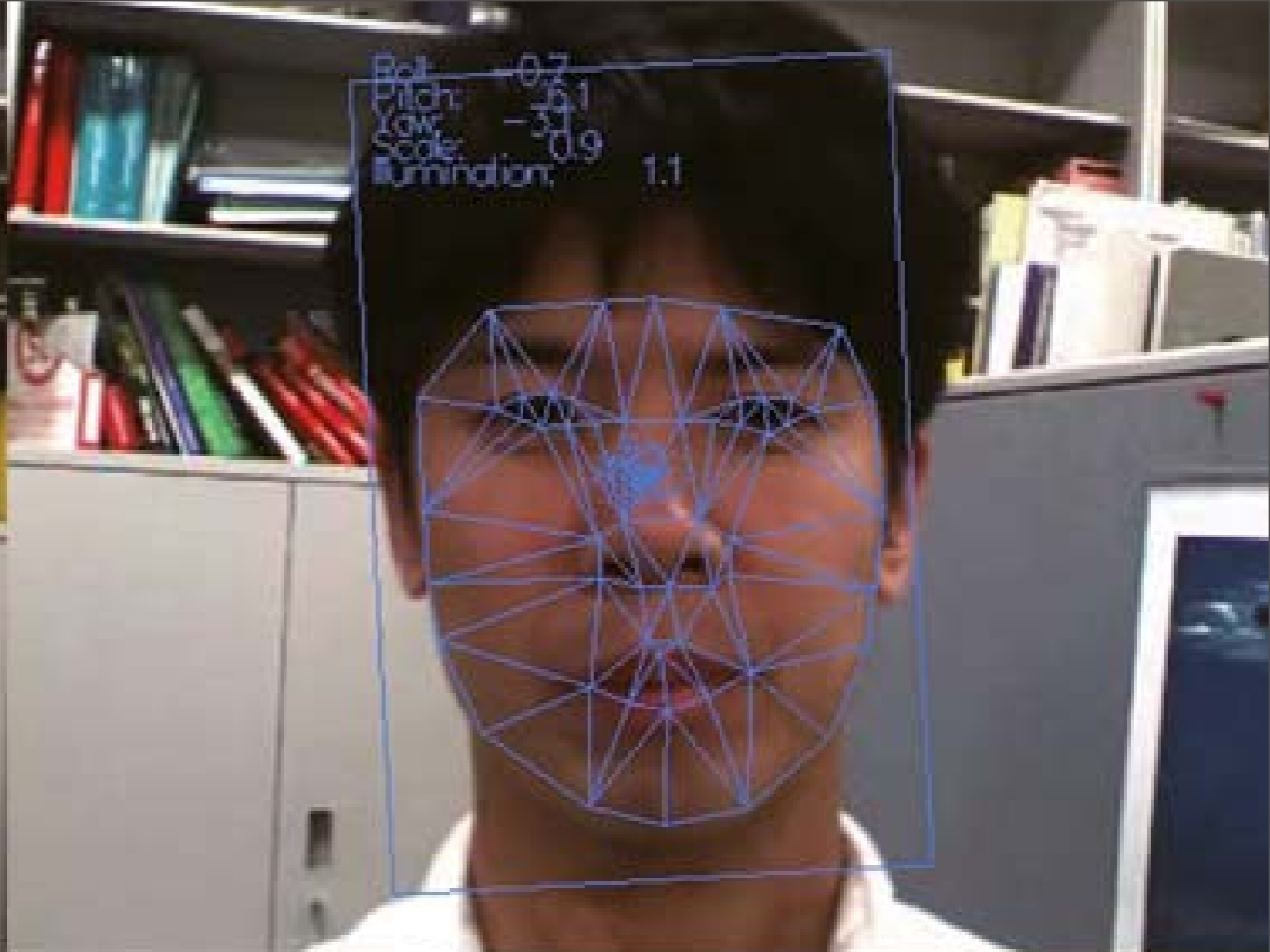
OSCAR MATEO LOZANO & KUZAHIRO OTSUKA  
PRESENTED BY PIOTR RUDOL

Particles per face: 1000 FPS: 30.7

Roll: 0.0  
Pitch: 0.0  
Yaw: -1.1  
Scale: 0.8  
Illumination: 1.0



Roll: -0.7  
Pitch: 3.1  
Yaw: -3.1  
Scale: 0.9  
Illumination: 1.1



Roll: 10.9  
Pitch: 0.3  
Yaw: 0.1  
Scale: 0.6  
Illumination: 0.9



Roll: -2.6  
Pitch: 3.3  
Yaw: 4.1  
Scale: 1.0  
Illumination: 1.1



# OUTLINE

- ✻ Particle Filter Basics
- ✻ Visual Tracking of Faces - Problem Definition
- ✻ CUDA Implementation
- ✻ Results
- ✻ Conclusion

# PARTICLE FILTERING

**Particle filtering** is a **state estimation** technique based on Monte Carlo simulations.

**Particles** are **random values of a state-space** variable which are **checked against** the current **output measure** to generate a weight value i.e. likelihood of a particle being the one that best describes the current state of the system.

**The collection of all particles and their weights** in each instant is a **numerical approximation** of the **probability density function of the system**.

The **probabilistic** approach of this method provides **significant robustness**, as several possible states of the system are tracked at any given moment.

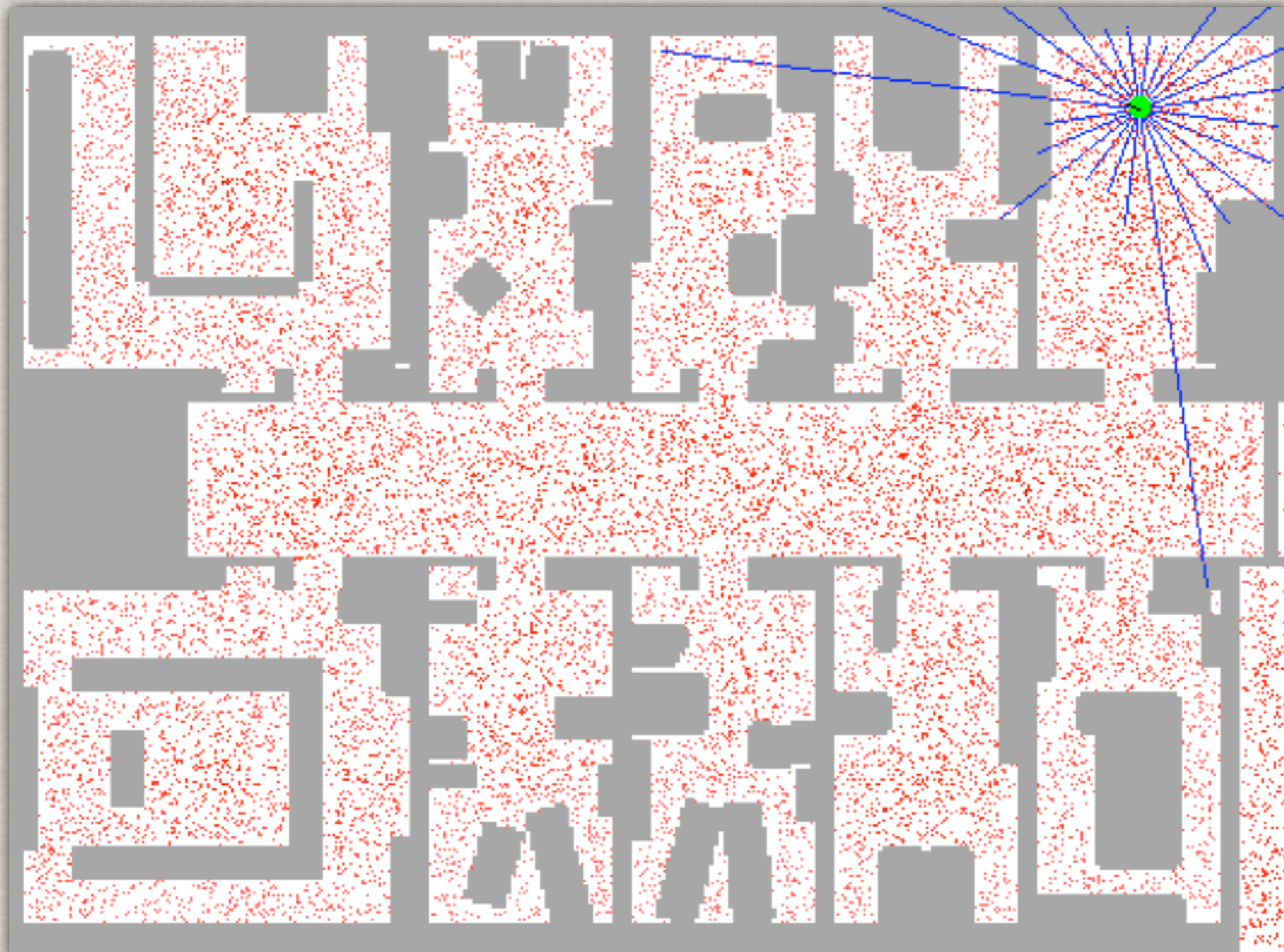
# PARTICLE FILTERING

First, a population of  $M$  samples is created by sampling from the prior distribution at time 0,  $\mathbf{P}(\mathbf{X}_0)$ . Then the update cycle is repeated for each time step:

- Each sample is propagated forward by sampling the next state value  $\mathbf{x}_{t+1}$  given the current value  $\mathbf{x}_t$  for the sample, using the transition model  $\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t)$ .
- Each sample is weighted by the likelihood it assigns to the new evidence  $P(\mathbf{e}_{t+1} | \mathbf{x}_{t+1})$ .
- The population is resampled to generate a new population of  $M$  samples. Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight.

This procedure puts in focus samples in high-probability regions of the state space.

# PARTICLE FILTER-BASED LOCALIZATION



# PROBLEM DEFINITION

We consider the state sequence of a target:

$$X_t = (T_x, T_y, S, R_x, R_y, R_z, \alpha)$$

$T_x, T_y$  - translational coordinate

$S$  - scale

$R_x, R_y, R_z$  - rotations along all axis

$\alpha$  - global illumination variable



# PROBLEM DEFINITION

The state-space vector is given in each discrete time step by:

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \quad - \quad \text{state transition model}$$

$f_t$  is possibly non-linear function of state  $\mathbf{x}_{t-1}$ ,  $\{\mathbf{v}_{t-1}\}$  is an independent and identically-distributed process noise sequence.

The objective of tracking is to recursively estimate  $\mathbf{x}_t$  from measurements:

$$\mathbf{z}_t = h_t(\mathbf{x}_t, \mathbf{n}_t) \quad - \quad \text{observation model}$$

$h_t$  is possibly non-linear function,  $\{\mathbf{n}_{t-1}\}$  is an independent and identically-distributed measurement noise sequence.

We seek filtered estimates of  $\mathbf{x}_t$  based on the set of all available measurements  $\mathbf{z}_{1:t}$  up to time  $t$ , together with some degree of belief in them. Namely, we want to know the p.d.f.  $p(\mathbf{x}_t | \mathbf{z}_{1:t})$ , assuming that the initial p.d.f.  $p(\mathbf{x}_0)$  is known.

# PARTICLE FILTER SOLUTION

We consider discrete particles forming the set:

$$\left(\tilde{\mathbf{X}}_{t-1}, \Pi_{t-1}\right) = \left\{ \left(\tilde{\mathbf{x}}_{t-1}^0, \pi_{t-1}^0\right), \dots, \left(\tilde{\mathbf{x}}_{t-1}^M, \pi_{t-1}^M\right) \right\}$$

every particle contains information about one possible state of the system and its importance weight.

This sample is propagated in the **Condensation** algorithm as follows:

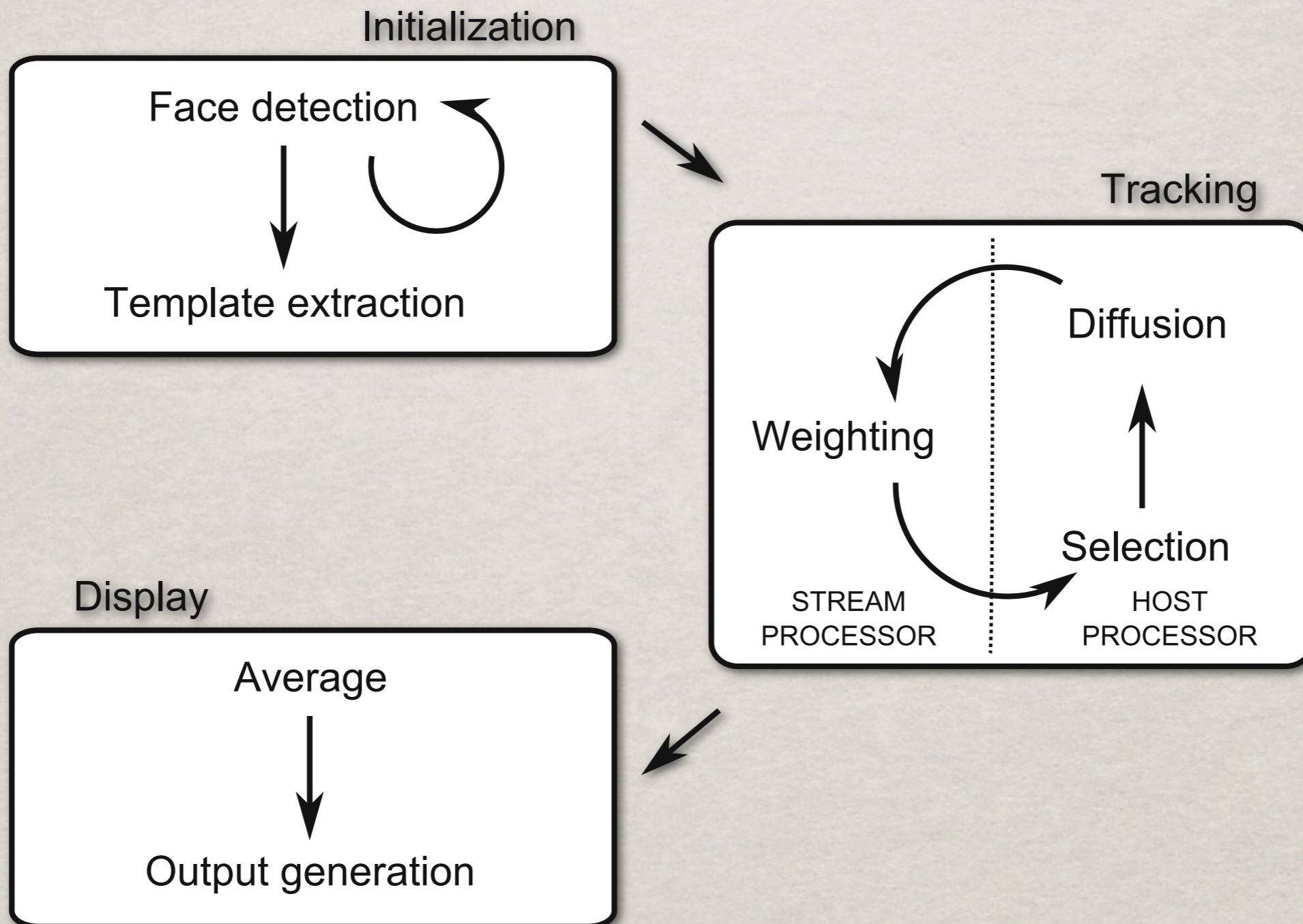
*Select* - A set of **M** new particles is generated by random selection from the previous particles by updating the p.d.f. according their weight.

*Diffuse* - A Gaussian noise is added to each parameter of each particle

*Measure weight* - The likelihood of each particle being the true state is calculated based on template matching using the input image at each step.



# THE ALGORITHM

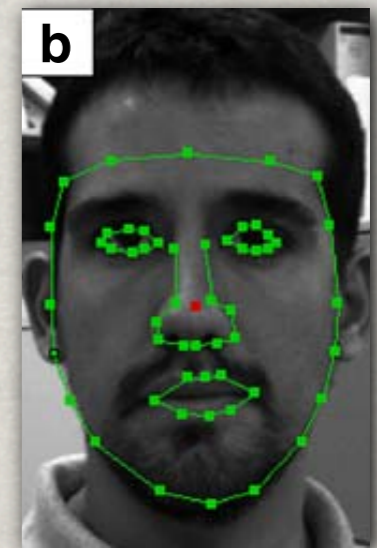


# INITIALIZATION

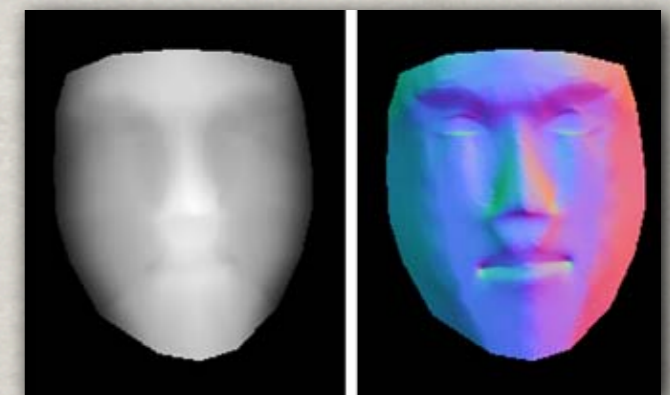
Every n-th frame is analyzed by a boosting algorithm to detect new faces giving a rectangular sub-image.



A fitting of a statistical deformable model is performed resulting in a series of 53 landmark points that correspond to known features of a human face.



The landmarks include a pair of texture coordinates, a set of numbers that refer to a unique part of a texture resource stored in memory. This way two image buffers (“height-” and “normalmap”) are created by applying a 3D face model.



# INITIALIZATION

The feature points (position and gray level) that form the sparse template are selected using image processing techniques, and their depth and normal to the surface values are extracted from prior maps.

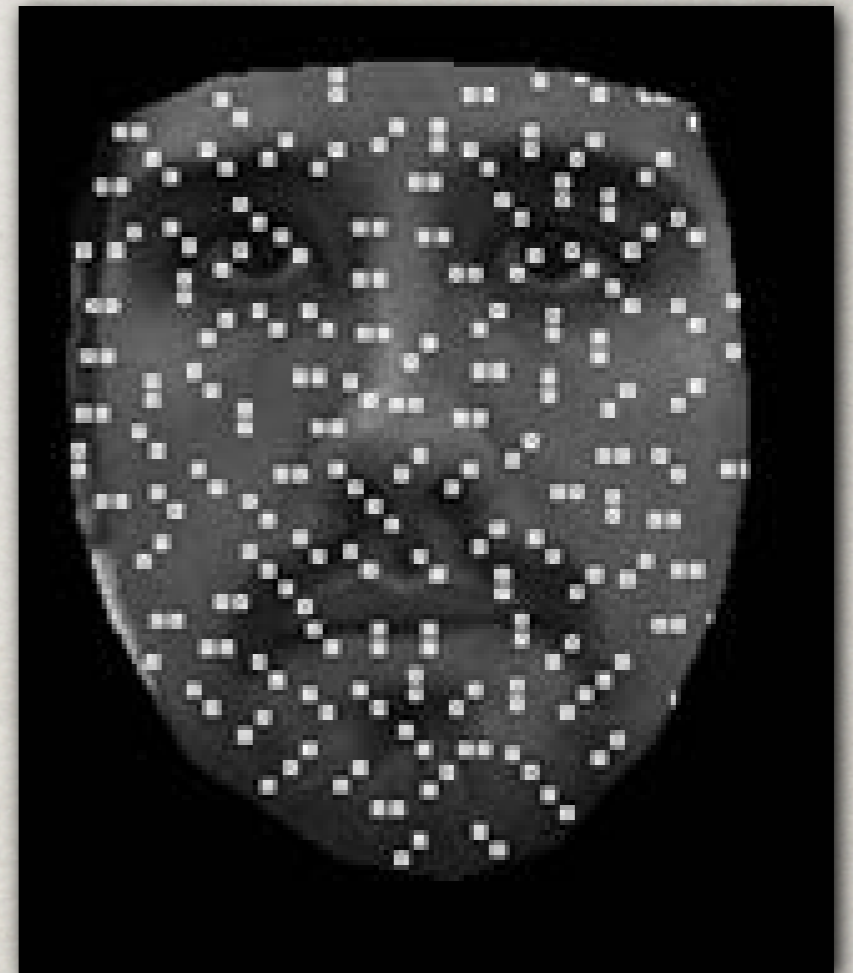
Finally the template is formed by a stream of  $\mathbf{N}$  feature points formed as follows:

$$\rho = (P_x, P_y, P_z, N_x, N_y, N_z, J)$$

$P_x, P_y, P_z$  - coordinates of feature points

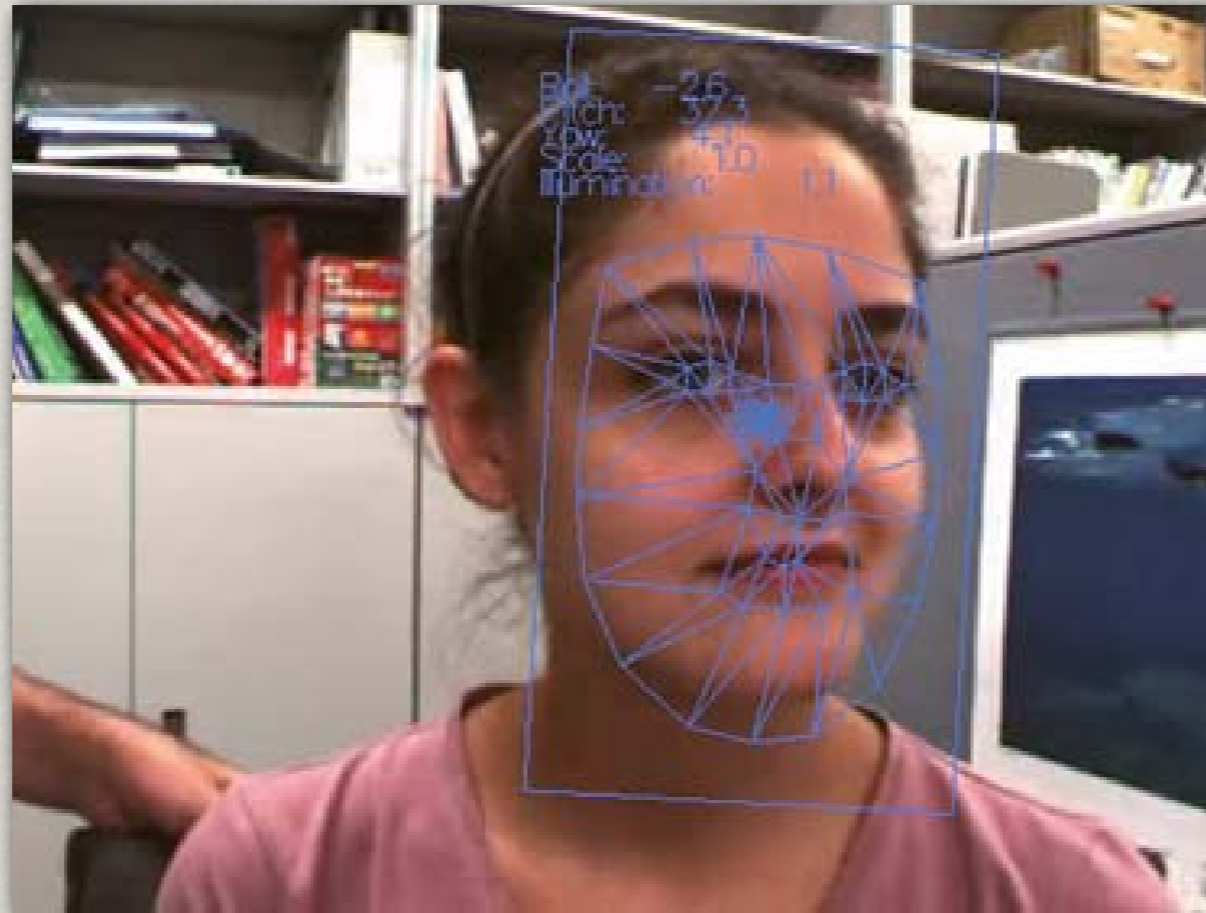
$N_x, N_y, N_z$  - form the vector normal to the face surface in the feature point

$J$  - gray level of the feature point



# INITIALIZATION

Why do feature points include normal vectors?



If a normal vector points away from the camera it is likely occluded by another feature point. It's weight can be disregarded.

# TRACKING

Tracking is performed by means of a Particle Filter.

Out of *selection*, *diffusion*, and *weighting* the latter is the most computationally intensive.

It requires performing the 3D transformation of each feature point as estimated by each of the particles, and then a comparison is made of the feature point gray level against the resulting point in the full image. The sum of all those comparisons for each feature point results in the weight of one particle.

For example with  $M=1000$  particles and  $N=230$  feature points it's **230000** weight computation operations!

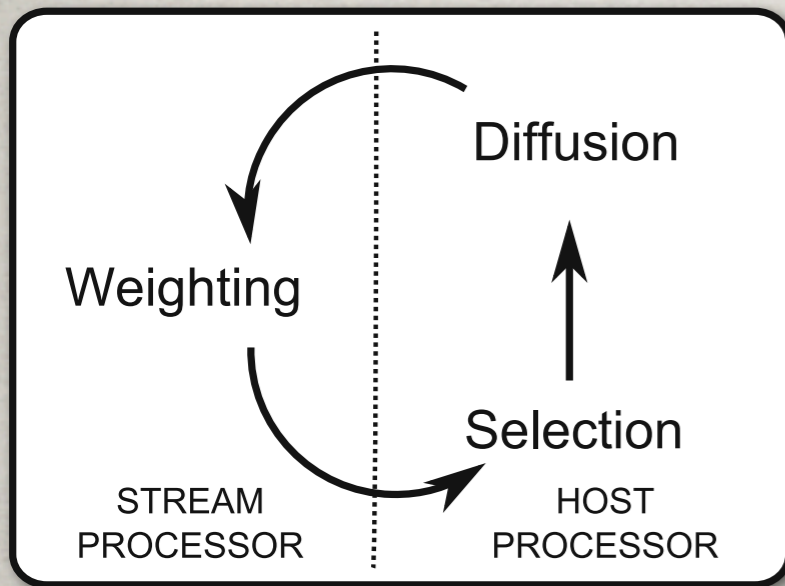
Weight calculation of each particle is an independent process!

# DISPLAY



After every tracking step, the best particles are averaged to get an approximation of the system state. The resulting state-space values can be displayed on the screen as rotated, scaled and translated 3D-mesh models.

# CUDA IMPLEMENTATION



After initialization, **feature point stream** is copied to **device memory**. It doesn't change during tracking.

At every step the **current video frame** and a **buffer with particles** is copied to **device memory**.

The kernel is invoked and executed in M blocks, each with N threads i.e. **one block per particle and one thread per feature point per block**.

**Each thread** computes the **matching error of a particle-feature pair** i.e. transforms the feature point position and normal vector, accesses the video frame pixel in that position (if normal vector does not mean occlusion) and compares the gray level.

The result is placed by every thread in different position in the block's **shared memory**.

When all threads in a block finish (sync), first thread of every block sums up matching errors and the result is placed in **global device memory**.

The host retrieves **stream of weights** of every particle.

# HARDWARE

Intel Core II 2.66GHz host system with 2 GB RAM

NVIDIA GeForce 8800GTX GPU:

- 128 processing units,
- 16 multiprocessors,
- 350GFLOPS under CUDA.

For comparison purposes, a software-only version of the weighting algorithm was also developed (single-threaded, no SIMD extensions used) for execution on the host CPU alone.

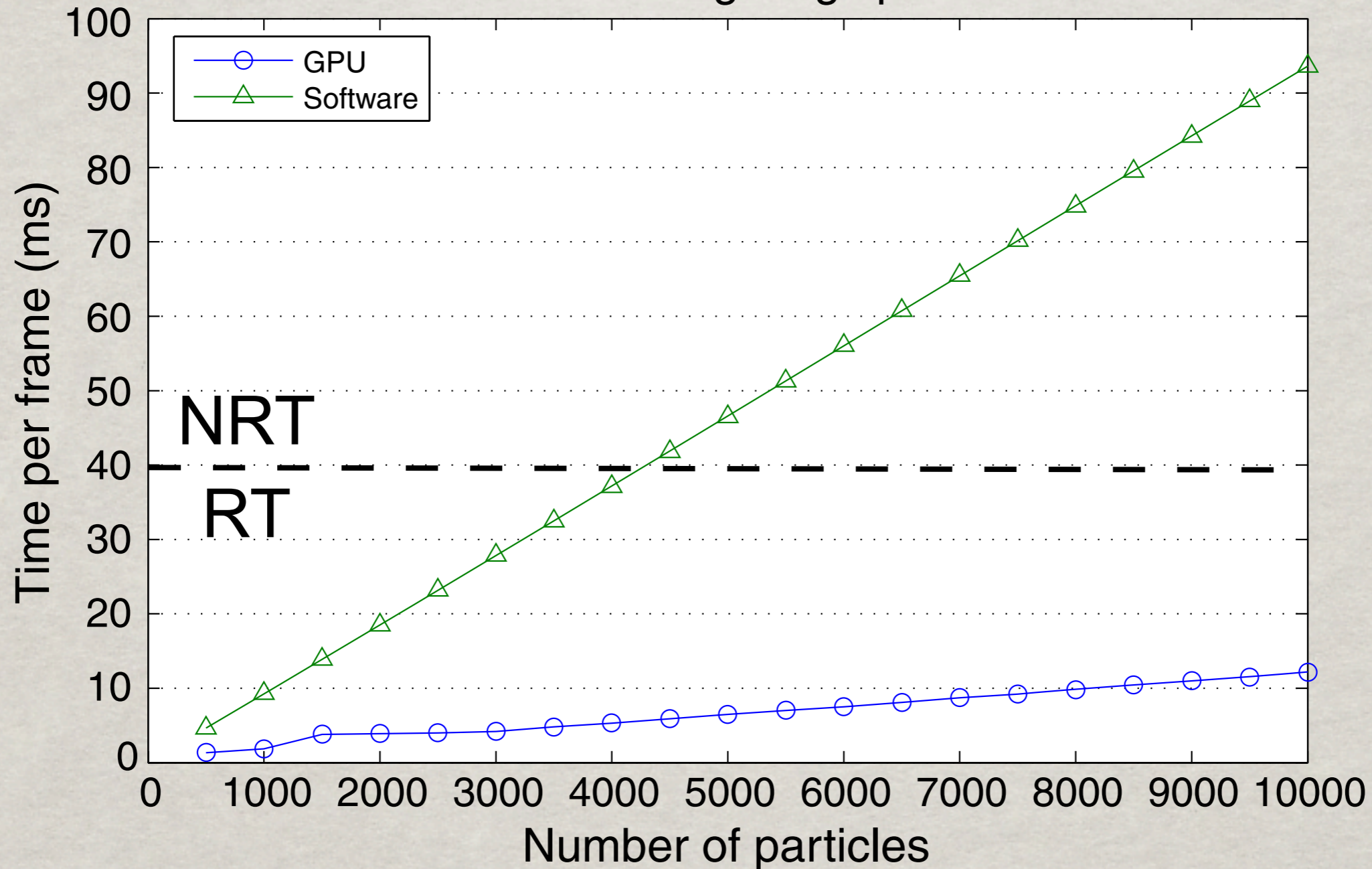
# RESULTS



Results of the simultaneous tracking of four people. The frame sequence is taken from a synthetic  $1024 \times 768$  video, the sparse templates are composed of approximately 230 feature points, and each template is tracked using 1000 particles.

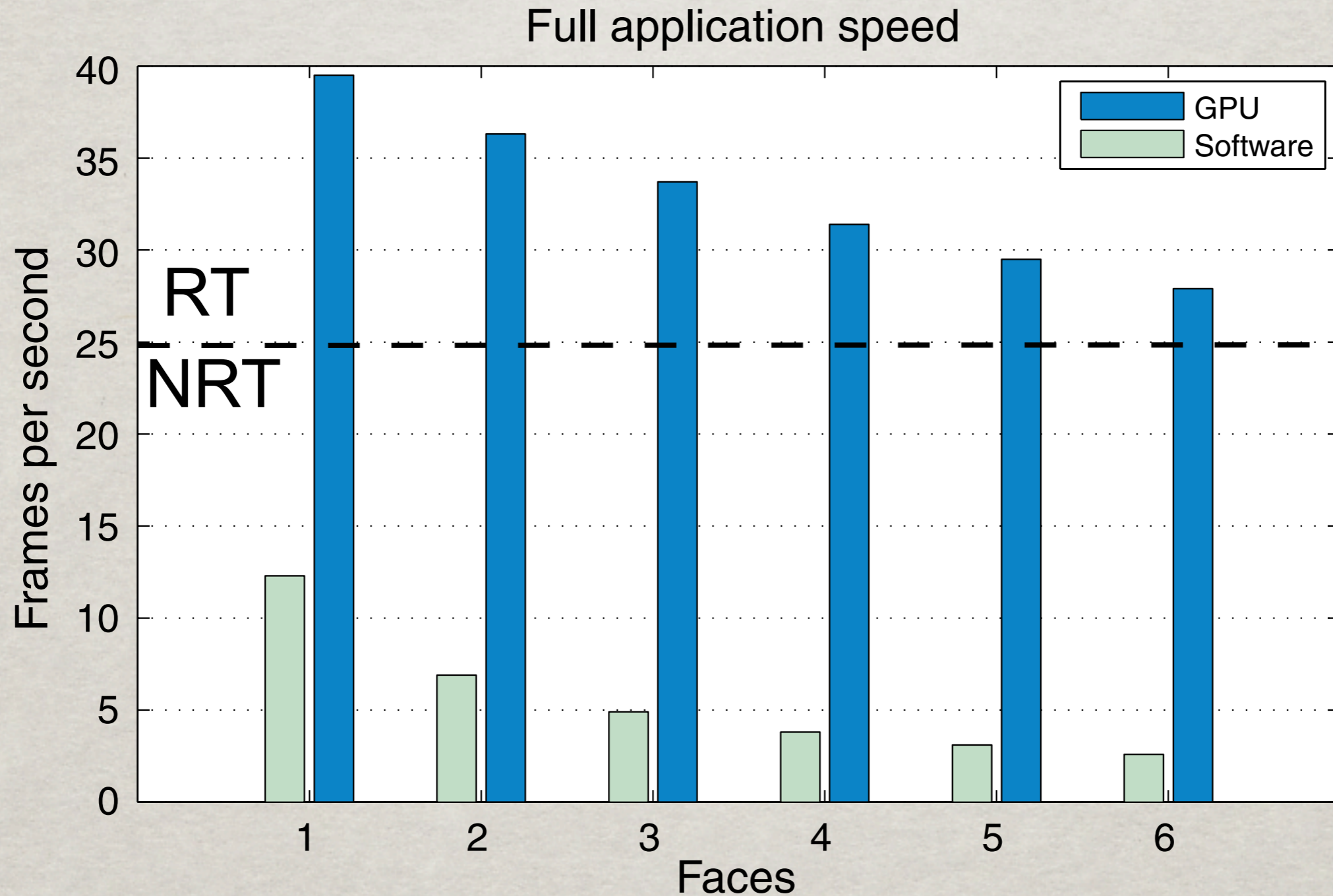
# RESULTS

## Particle weighting speed



Speed of the particle weighting stage, comparing Stream processing in the GPU version to a serial CPU-only version. Video 1024×768 and 217 feature points

# RESULTS



Speed of the full application, comparing stream processing in the GPU version to a serial CPU-only version. Video 1024x768, 230 feature points per face, and 1000 particles per face.

# SUMMARY

- ✿ Particle Filter on GPU-based visual tracking of human faces was presented.
- ✿ Thanks to easy parallelization of weight computation operations a significant performance increase was achieved.
- ✿ GPU based filter runs easily in real-time with 10k particles comparing to approx. 4.5k for CPU only version.
- ✿ GPU based filter can track up to 6 faces in a video stream in real time comparing to 0 for CPU only version.
- ✿ The GPU-based particle filter can easily be used in different domains.