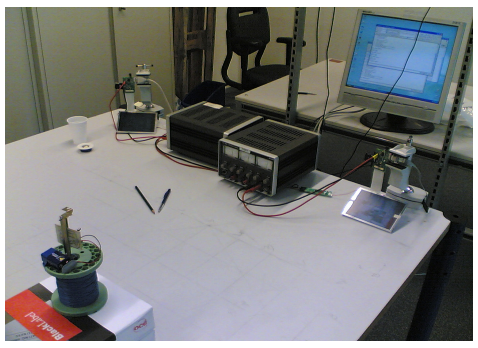
# Multiple Camera View Calibration for Localization

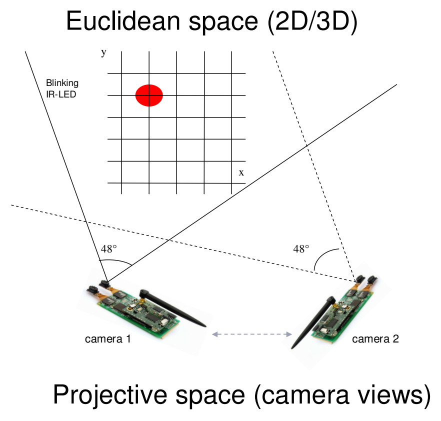
Peter Meijer[1], Christian Leistner[2] and Anthony Martinière[1]

1. NXP Semiconductors, Eindhoven, The Netherlands
2. Graz University of Technology, Graz, Austria

**ABSTRACT -** The recent development of distributed smart camera networks allows for automated multiple view processing. Quick and easy calibration of uncalibrated multiple camera setups is important for practical uses of such systems by non-experts and in temporary setups. In this paper we discuss options for calibration, illustrated with a basic two-camera setup where each camera is a smart camera mote with a highly parallel SIMD processor and an 8051 microcontroller. In order to accommodate arbitrary (lens) distortion, perspective mapping and transforms for which no analytic inverse is known, we propose the use of neural networks to map projective grid space back to Euclidean space for use in 3D localization and 3D view interpretation.

## Neural Networks applied to map the projective grid space in multiple camera views back to Euclidean "physical" space





measure $\rightarrow$

### 2D case

$$\begin{pmatrix} i_{1m} \\ i_{2m} \end{pmatrix} = \boldsymbol{F} \begin{pmatrix} x_m \\ y_m \end{pmatrix}$$

neural networks for inverse modeling $\downarrow$ ?

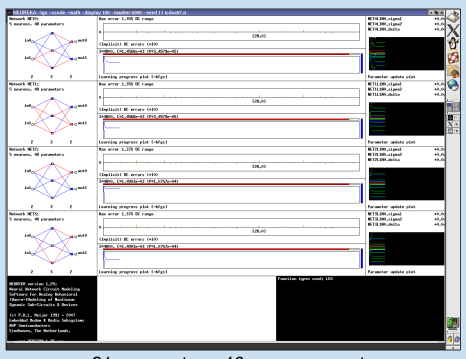
$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \boldsymbol{F}^{-1} \begin{pmatrix} i_{1m} \\ i_{2m} \end{pmatrix}$$

$$\begin{aligned} s_{ik} &\triangleq \boldsymbol{w}_{ik} \cdot \boldsymbol{y}_{k-1} - \theta_{ik} + \boldsymbol{v}_{ik} \cdot \frac{\mathrm{d}\boldsymbol{y}_{k-1}}{\mathrm{d}t} \\ &= \sum_{j=1}^{N_{k-1}} w_{ijk} y_{j,k-1} - \theta_{ik} + \sum_{j=1}^{N_{k-1}} v_{ijk} \frac{\mathrm{d}y_{j,k-1}}{\mathrm{d}t} \end{aligned}$$

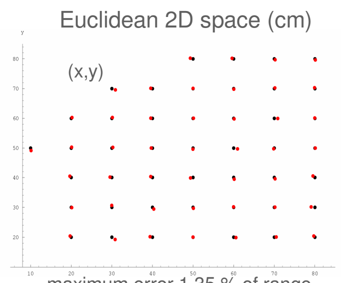
$$\tau_{2,ik} \frac{\mathrm{d}^2 y_{ik}}{\mathrm{d}t^2} + \tau_{1,ik} \frac{\mathrm{d}y_{ik}}{\mathrm{d}t} + y_{ik} = \mathcal{F}^{(ik)}(s_{ik}, \delta_{ik})$$

$$\mathcal{F}(s_{ik}) \triangleq \frac{1}{1 + e^{-s_{ik}}}$$

blue parts extend the standard multilayer perceptron networks


21 parameters, 46 measurements

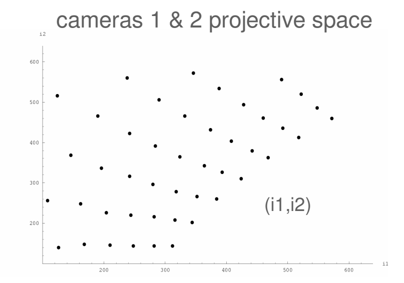
ANSI C $\rightarrow$

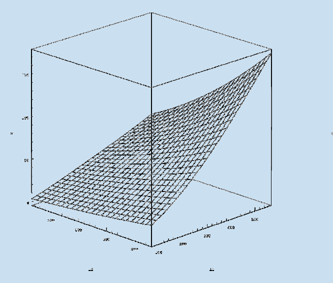


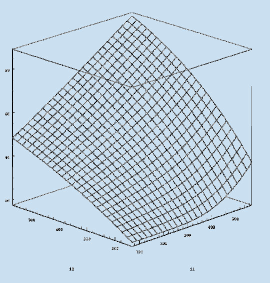
maximum error 1.35 % of range

measure $\rightarrow$

$\leftarrow$ generate NN model



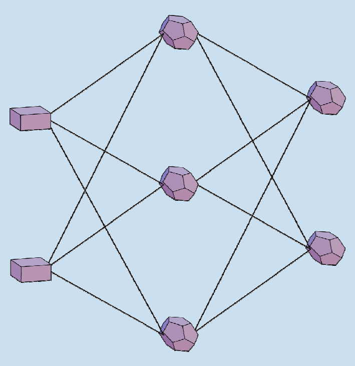




**Accounts for any projective distortion, lens distortion, etc.**

## Resulting neural network code runs on NXP smart camera mote with 8051 CPU

```c
#include <math.h>

double f1(double s) {
   return(1.0 / (1.0 + exp(-s)));
}

void net0(double in[], double out[]) {
   double net0l1n0, net0l1n1, net0l1n2,
   net0l2n0, net0l2n1;

   net0l1n0 =
   f1(-3.8127875846276100e-03 * in[0]
      -4.2295785912067137e-04 * in[1]
      +3.1665623494486885e+00);

   net0l1n1 =
   f1(-9.1588388556664759e-03 * in[0]
      +1.2290676953540203e-04 * in[1]
      +9.0614964706215173e-01);

   net0l1n2 =
   f1(-3.3711715680603675e-03 * in[0]
      +1.2160470931375164e-03 * in[1]
      +1.9435120601542750e+00);

   net0l2n0 =
   f1(+4.3930104113043384e+00 * net0l1n0
      -6.5139698438269900e-01 * net0l1n1
      -7.0793079196186302e+00 * net0l1n2
      -1.2824132753249337e-01);

   net0l2n1 =
   f1(-1.0007183969110965e+01 * net0l1n0
      -2.7076964131170778e-01 * net0l1n1
      +6.7926874876093679e+00 * net0l1n2
      +5.4240606874043866e+00);

   out[0] = -3.3278043362310491e+01
            +7.0023061720757732e+02 * net0l2n0;

   out[1] = -4.8864233148180318e+02
            +6.0016425300491528e+02 * net0l2n1;
}
```

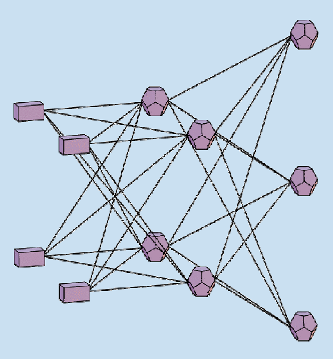


**The calibrated smart camera network determines any physical locations in 2D Euclidean space**

## Smart cameras now have brains!

**General approach for distributed camera setups: dynamic neural networks can also acommodate calibration for imperfect timing among cameras, e.g., due to systematic camera network latencies or intrinsic image sensor delays**

## Same calibration approach for 3D ➡ 3D physical locations from two camera views

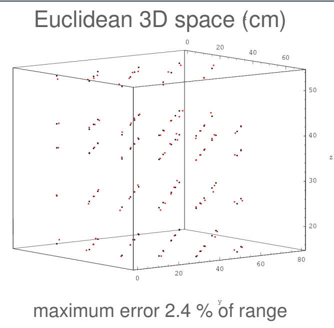


$$\begin{pmatrix} i_{1m} \\ j_{1m} \\ i_{2m} \\ j_{2m} \end{pmatrix} = \boldsymbol{F} \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix}$$

$\downarrow$ ?

$$\begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} = \boldsymbol{F}^{-1} \begin{pmatrix} i_{1m} \\ j_{1m} \\ i_{2m} \\ j_{2m} \end{pmatrix}$$

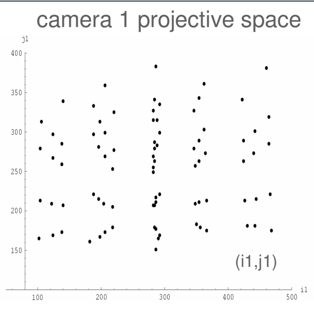
48 parameters, 92 measurements

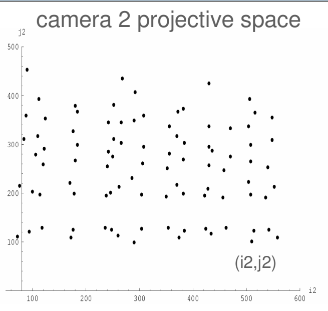


maximum error 2.4 % of range

measure $\rightarrow$

$\leftarrow$ generate NN model


camera 1 projective space
(i1,j1)


camera 2 projective space
(i2,j2)