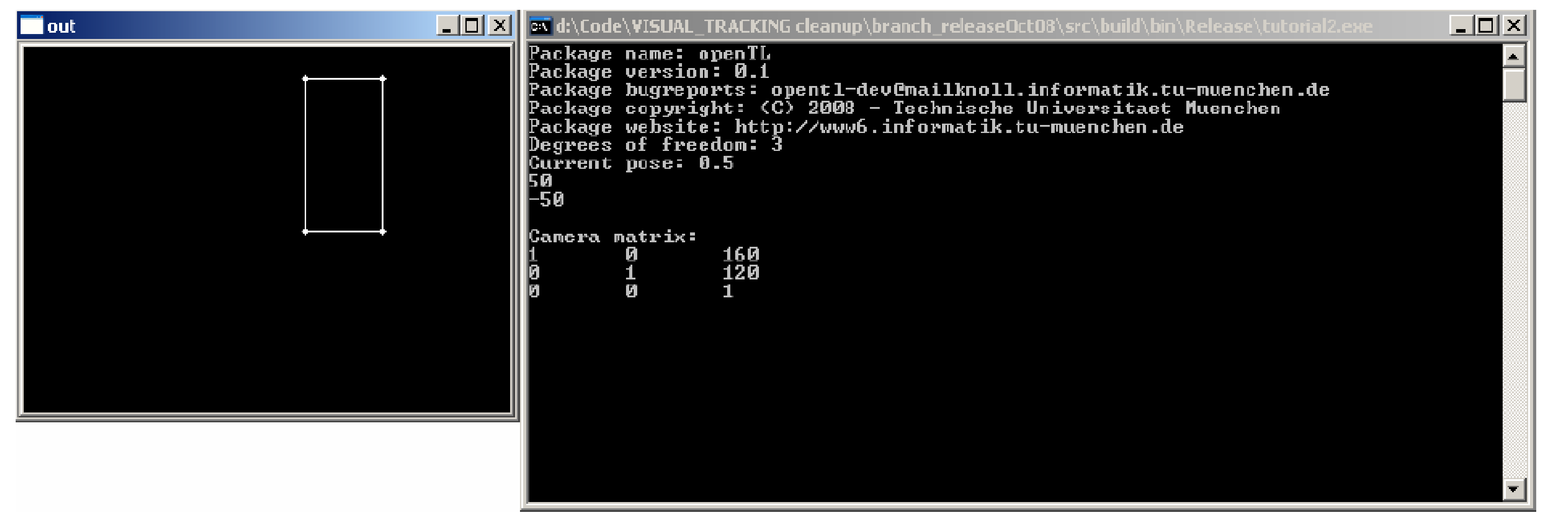
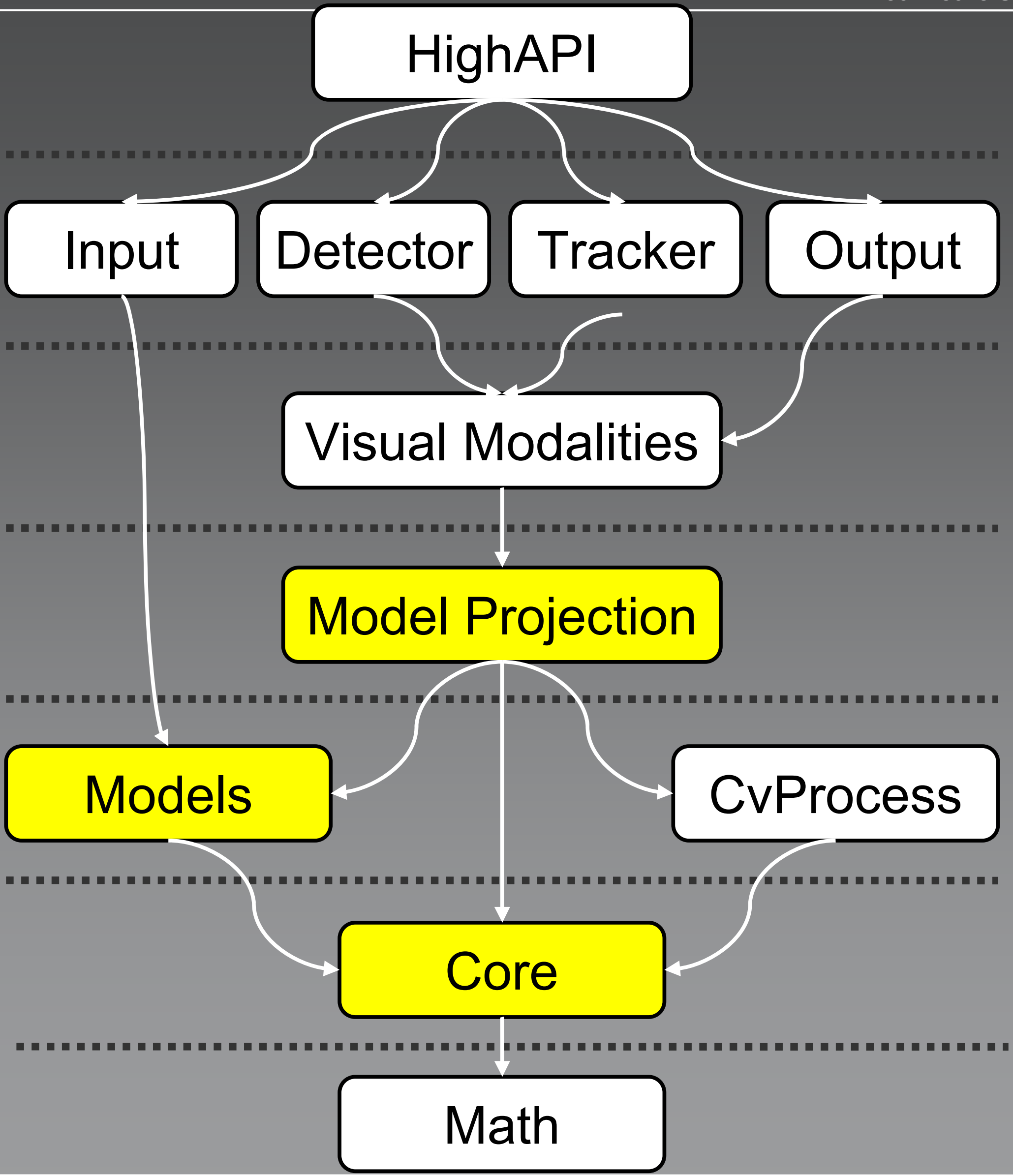
OpenTL – Tutorial 2

- Projecting points to screen using a pose-representation


out
d:\Code\VISUAL_TRACKING cleanup\branch_releaseOct08\src\build\bin\Release\tutorial2.exe

```
Package name: openTL
Package version: 0.1
Package bugreports: opentl-dev@mailknoll.informatik.tu-muenchen.de
Package copyright: (C) 2008 - Technische Universitaet Muenchen
Package website: http://www6.informatik.tu-muenchen.de
Degrees of freedom: 3
Current pose: 0.5
50
-50

Camera matrix:
1       0       160
0       1       120
0       0       1
```

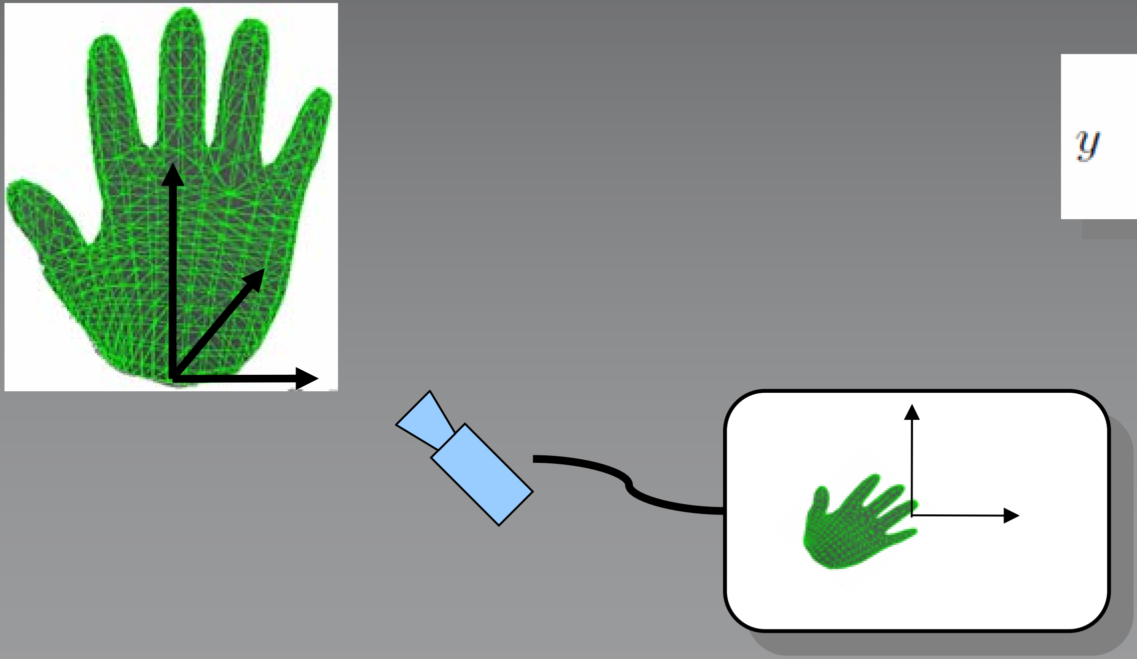

HighAPI
Input
Detector
Tracker
Output
Visual Modalities
Model Projection
Models
CvProcess
Core
Math

Model Projection – Warp

- Mapping from object to screen homogeneous coordinates:

$$\bar{y} = K \cdot T \cdot \bar{x}$$

$$y = \begin{bmatrix} \frac{\bar{y}_1}{\bar{y}_3} & \frac{\bar{y}_2}{\bar{y}_3} \end{bmatrix}^T$$



Model Projection - Warp

- Managing the relationship between object and camera spaces

  → modelprojection::Warp warp(sensVector)

- Update internal camera matrices and Jacobians whenever the state vector changes

  → warp.warpUpdate(stateVec)

Models – Sensor Model

$$\bar{y} \quad = \quad K \cdot T \cdot \bar{x}$$

- models::SensorModel sensor
  → K-Matrix

$$K = \begin{bmatrix} 1 & 0 & 0 & r_x/2 \\ 0 & 1 & 0 & r_y/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2D → focus=1

- Internal camera parameters
  → sensor.setK(focus, xres, yres);

- Support of multiple cameras
  → std::vector<models::SensorModel *> sensVector;

Core

$$\bar{y} \quad = \quad K \cdot T \cdot \bar{x}$$

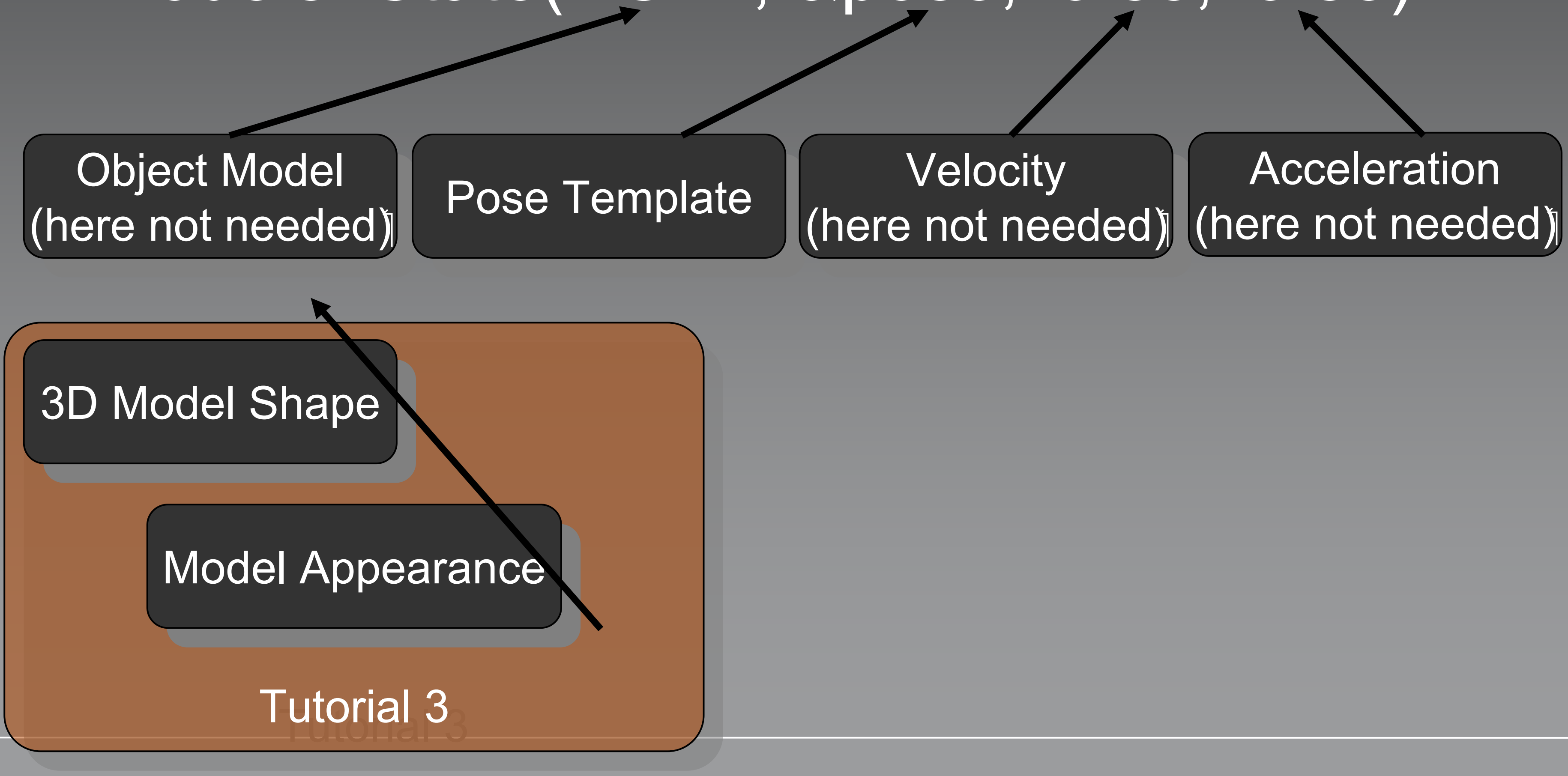
- cvdata::Pose2d1ScaleTranslation pose
  → T-Matrix

$$T = \begin{bmatrix} s & 0 & 0 & t_x \\ 0 & s & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 3 degrees of freedom
  → 1 scale, 2 translation

Models – Object State

- models::State(NULL, &pose, false, false)



Object Model
(here not needed)

Pose Template

Velocity
(here not needed)

Acceleration
(here not needed)

3D Model Shape

Model Appearance

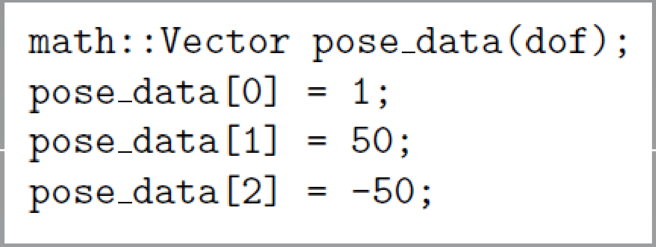
Tutorial 3

Models – Object State

- Support of multiple objects
  → Multiple States!

- Use of a vector of states:
  std::vector<boost::shared_ptr<opentl::models::State> >

- Set the data of the state:
  stateVec[0]->setDataFromSingleVector(&pose data);

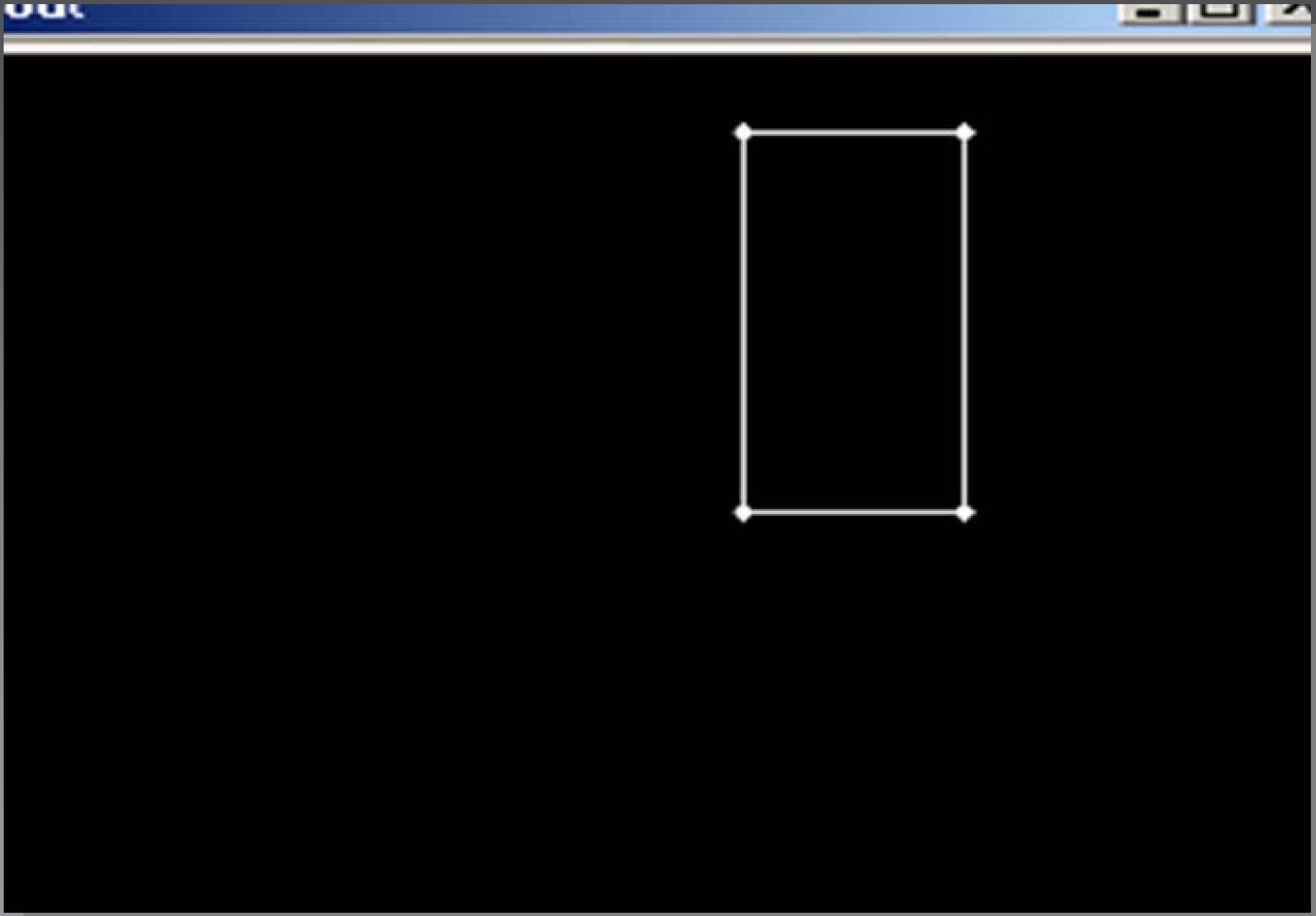


```
math::Vector pose_data(dof);
pose_data[0] = 1;
pose_data[1] = 50;
pose_data[2] = -50;
```

Task – Project 4 object points to the screen

- Define 4 object points in homogeneous coordinates (4 dof)
- Define the 4 resulting screen coordinates (2 dof)
- Warp the points with: *warpPoint*

  → warp.warpPoint(*stateVec[0]->getPos(), &objP1, &screenP1);
- Show the points in a cvdata::Image
  → use OpenCV methods *CvPoint, CvLine*


out