# Robotiks

## 1. Connect to the model car via SSH

**Screenshots machen**

- Verbinde mittels Ethernet/WLAN
- Verbinde mittels SSH `ssh root@192.168.1.199` Passwort `elfmeter`
- erstelle ein `vim /root/hello_car_GRUPPENNAME`
- schreibe aktuelles Datum/Uhrzeit in dei Datei
- Neuer terminal, kopiere obige Datei auf lokalen Rechner:
  - `scp root@192.168.1.199:/root/hello_car_GRUPPENNAME ~/hello_car_GRUPPENNAME`

## 2. Fork des AutoModelCar

- https://github.com/AutoModelCar/catkin_ws_user
- übermittle die URL des forks

## 3 Fahrbahn vorbereiten

- 6 weiße Punkte aufkleben
- Foto davon machen
- *Koordinaten* sichern (Abstände zum Auto merken)

```
*   *
*   *
*   *

qTp
| |
d_b
```

## 4. Monochrome/Grayscale image

- Subscribe to `…sub("/app/camera/rgb/image_raw",Image,self.callback, queue_size=1)`
- wandle RGB in grayscale um
- publish to a new topic

# 5. SW Bild

- verwende `cv::threshold` um SW-Bild zu erstellen. - screenshot
- publish to a new topic

# 6. (4.) finde die 6 Punkt im SW Bild

- finde die 6 Punkte im SW bild,
- print coordinates to terminal - screenshot

# 7. (5.)

- verwende `cv::solvePNP` um realworld Koordinaten zu berechnen:

```
  //6 Klebe-Punkte auf dem SW-Bild (2D)
  std::vector<cv::Point2f> imagePoints = Generate2DPoints(bi_gray_img);
  //6 Klebe-Punkte in realer Welt (3D gemessen)
  std::vector<cv::Point3f> objectPoints = Generate3DPoints();

  // Stelle Matrix auf mit Werten aus Assignment3.pdf
  /*fx,0,cx,0,fy,cy,0,0,1*/
  Mat intrinsics = (Mat_<double>(3,3) << 614.1699, 0, 329.9491,
  0, 614.9002, 237.2788, 0, 0, 1);

  //erstelle 4D Vektor mit Werten aus Assigmnent3.pdf
  cv::Mat distCoeffs(4,1,cv::DataType<double>::type);
  distCoeffs.at<double>(0) = 0.1115;
  distCoeffs.at<double>(1) = -0.1089;
  distCoeffs.at<double>(2) = 0;
  distCoeffs.at<double>(3) = 0;

  //erstelle Output fuer solvePnP
  cv::Mat rvec(3,1,cv::DataType<double>::type);
  cv::Mat tvec(3,1,cv::DataType<double>::type);

  cv::solvePnP(objectPoints, imagePoints, intrinsics, distCoeffs, rvec, tvec);

  std::cout << "rvec: \n" << rvec << std::endl;
std::cout << "tvec: \n" << tvec << std::endl;
```