Lab 9 – Segmentation

Proposal 1 – Color based object detection

Estimate a multivariate normal distribution for a set of pixels

- Capture an image that samples the colors of an object/surface
- Represent the sampled pixels as feature vectors with R-, G- and B-values
- Estimate a multivariate normal distribution for the feature vectors
  - Estimate the expectation vector $\boldsymbol{\mu}$ and the covariance matrix Σ (see the example in Lecture 9.3)

Compare new images with the estimated model

- Capture a new image
- Compute the corresponding probability density image by comparing each pixel with the multivariate normal distribution (Mahalanobis distance)
- Threshold the probability density image, e.g. by using Otsu's method
- Experiment with morphological operations to clean up the thresholded image
- Present the final result

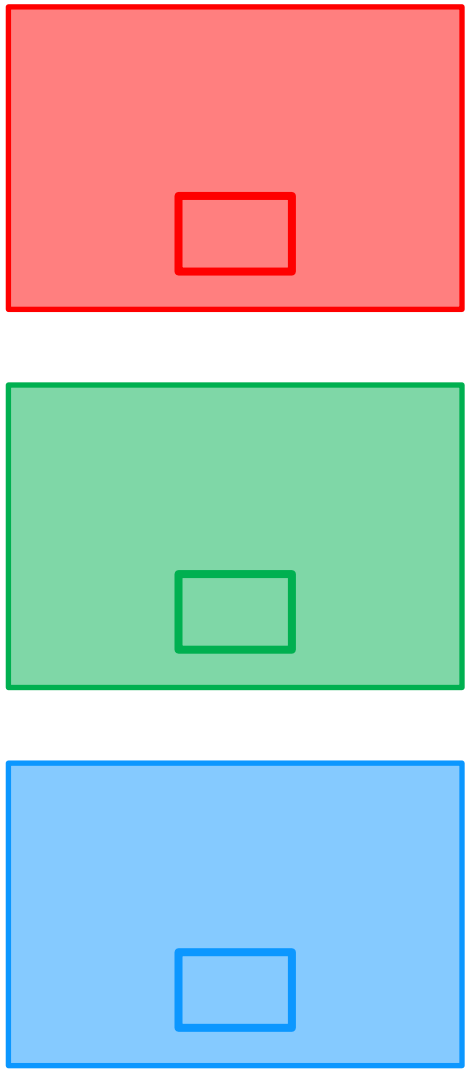
Proposal 2 – Detection of moving objects in image sequences

Implement a simple object detector based on change detection

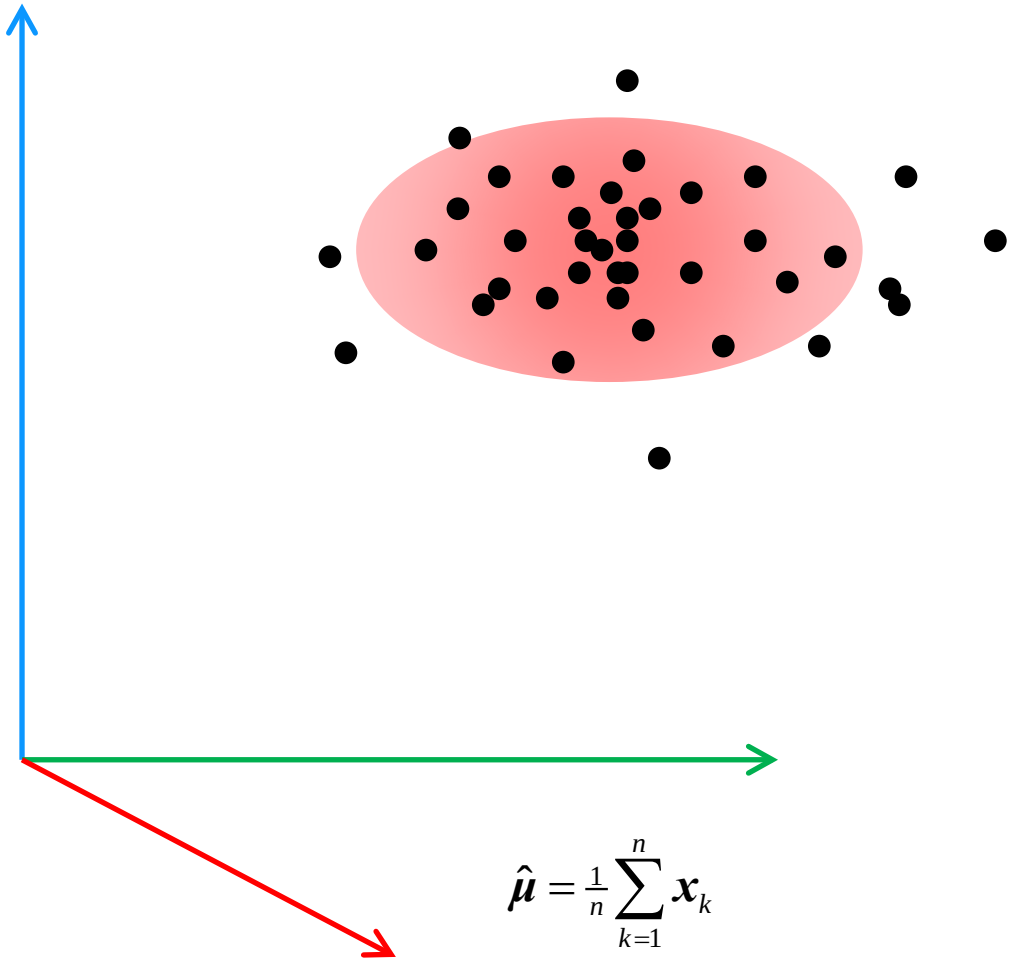
- Compute a time-averaged background (reference) image using a static camera
- Detect changes in the scene by computing the difference between a new image and the reference
- Threshold the difference image using e.g. Otsu’s method.
- Experiment with morphological operations for cleaning up the thresholded image (erosion, dilation, opening, closing).
- Classify moving objects by designing a simple decision rule based on aspect ratio (e.g. Height vs. Width ratio).

Color based segmentation

RGB- image with sampling rectangle



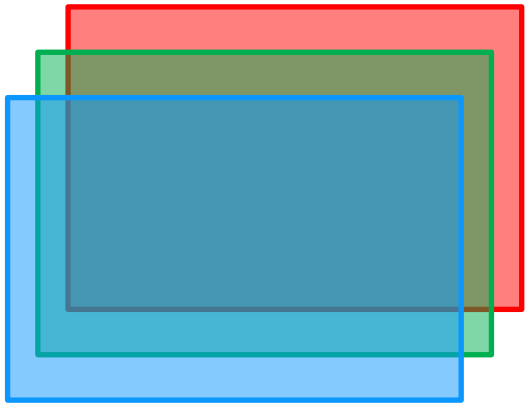
Estimate multivariate normal distribution for sample pixels

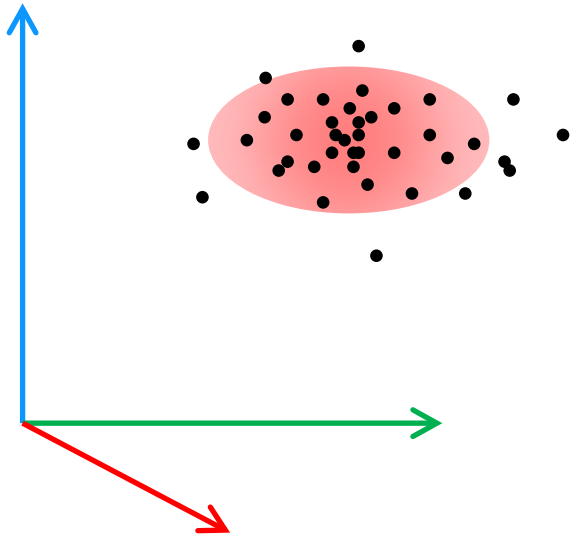


$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{x}_k$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n} \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}\right)\left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}\right)^T$$

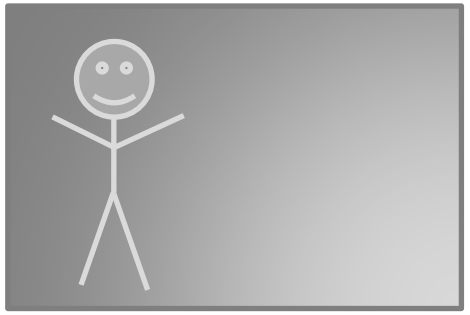
Test image





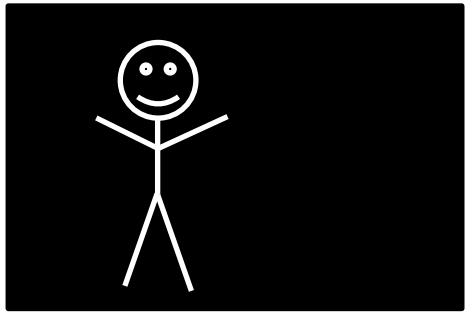
Compare pixels with multivariate normal distribution

«Similarity» image



Segmentation

Binary image



TODO 1

In `multivarate_normal_model.cpp`

```
void lab9::MultivariateNormalModel::performTraining(const cv::Mat& samples)
{
    // TODO 1: Estimate mu_ and inv_covar_(Inverse covariance matrix) from the training samples
    // Hint: See cv::calcCovarMatrix
    is_trained_ = false;
}
```

TODO 2

In `multivarate_normal_model.cpp`

```
cv::Mat lab9::MultivariateNormalModel::computeDensities(const cv::Mat& image)
{
    if (!is_trained_)
    { throw std::runtime_error("You forgot to train the model before trying to compute
    probability densities!"); }

    cv::Mat density;
    // TODO 2: Calculate density value for each pixel in image.
    // Hint: computeMahalanobisDistances is part of the formula (see below)
    return density;
}
```

TODO 3

In `multivarate_normal_model.cpp`

```
cv::Mat lab9::MultivariateNormalModel::computeMahalanobisDistances(const cv::Mat& image)
{
    cv::Mat image_64;
    image.convertTo(image_64, CV_64FC3);

    cv::Mat mahalanobis_dist(image.size(), CV_64FC1);

    // TODO 3: Calculate mahalanobis distance for each pixel. You will need mu_ and inv_covar_
    // Hint: see cv::Mahalanobis

    return mahalanobis_dist;
}
```

TODO 4

In `multivarate_normal_model.cpp`

```
cv::Mat lab9::extractTrainingSamples(const cv::Mat& source_image, const cv::Rect
sampling_rectangle)
{
  cv::Mat patch = source_image(sampling_rectangle).clone();
  // TODO 4: Experiment with other color spaces and other features
  // Hint: see cv::cvtColor
  cv::Mat samples = patch.reshape(1, patch.rows * patch.cols).t();
  return samples;
}
```

Suggestions for further work

- Use morphological operations to clean up the segmented image
  - Take a look at cv::morphologyEx()
- Try to continously update the model by swapping out some of the old feature vectors with new ones
- Try to expand the feature vectors with other properties