Practical Course “Artificial Intelligence” Requirements Analysis

Thomas Monninger, Stefan Bolz, Zuohao Chen, Ashish Negi, Samhita Ganguly

1. Requirements Analysis **2**

2. Data Collection **2**
- Camera Setup 2
- Table Setup 2
- Recording 3
- Situations 3

3. Preprocessing **3**
- Scenario Generation 3
- Table Coordinate System 3
- Image Processing 4
- Data augmentation 5

4. Modeling **5**
- Framework 5
- Model Architecture 6
- Input/Output 6
- Overcome Data Bottleneck 6
- Internal Representation 7
- Training 8

5. Evaluation **8**
- Evaluation 8
- Rendering 8

1. Requirements Analysis

This document reflects the requirements for Fachpraktikum Artificial Intelligence. It is structured according to the Cross Industry Standard Process for Data Mining (CRISP-DM):

1. Requirement Analysis
2. Data Collection
3. Preprocessing
4. Modeling
5. Evaluation

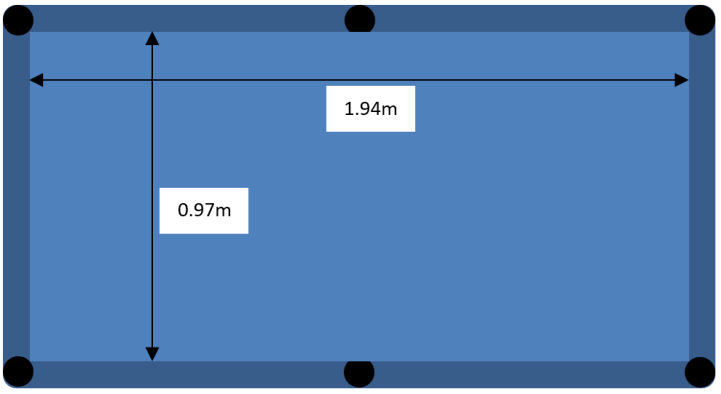
For each step in the CRISP-DM model one section in this requirement analysis is provided.

2. Data Collection

Camera Setup

- Capture full pool table
- Capture top view and avoid shearing
- Camera view shall be steady and give same view across all recording (Maybe slight differences in angle to make a more robust model?)
- Camera shall be mounted to provide fixed and steady field of view across the recording.
- Camera resolution shall be 1920x1080 pixel.
- Camera frame rate shall be 60 frames per second.
- Images of a printed checkerboard pattern shall be recorded from various angles and distances in order to calculate the lense distortion matrix

Table Setup

- Illumination shall be constant for all regions of pool table
- Illumination shall be from top or equally from all sides to minimize shadows
- Occlusions by billiard cue stick, arms, hands and others shall be avoided as good as possible.
- The table shall be a standard billiard table with 6 holes
- The table shall have dimensions 1.94m by 0.97m
- 

- The table shall have a subset of the default billiard configuration (16 balls)

Recording

- The full game play shall be recorded
- The data shall be available as video file
- At least 500 scenarios with a strike shall be recorded
- Before each action the recording shall cover the table free of occlusions
- A new strike shall be executed only after the balls are back in steady state.

Situations

- Realistic (Test set will likely consist of moves from regular game of pool)
- Various number of balls
- Various number of bounces
- Various directions to strike the balls
- Variation of stick usage
- Various forces to strike the balls
- Corner case: Ball falls in pocket (disappear)
- Corner case: Ball hits multiple balls at the same time
- Corner case: Ball hits boundary before it hits ball
- Corner case: Stick or some balls are hidden by hand or arm at strike
- Not in scope: Multiple sticks in frame
- Not in scope: Trick shots (jumps etc.)
- Not in scope: Irregular ball configuration

3. Preprocessing

Scenario Generation

- The video shall be cut into separate scenarios
- Each scenario shall reflect the execution of exactly one strike
- Additionally, for each scenario an initial frame without occlusions shall be provided (see req. in “Recording”)
- A scenario starts at the point of contact between stick and white ball
- One scenario shall have a length of 5 seconds
- Each scenario shall be captured as a series of images with framerate 30 Hz
- Each scenario hence holds 30 frames / second * 5 seconds = 150 frames
- For each of the 150 frames the ball positions shall be detected (see subsection “Image Processing”).
- The final scenario is provided as a list of ball positions per frame.

Table Coordinate System

- A function shall identify the 4 corner points of the billiard table
- A two-dimensional coordinate system shall be defined in the plane of the billiard table
- An affine transformation matrix shall be provided to convert from image coordinate frame to billiard table coordinate frame

- The center of the coordinate system shall be defined in the bottom left corner with positive x in direction of the bottom right corner along the length of the table and positive y in direction of the top left corner along the width of the table (see figure)

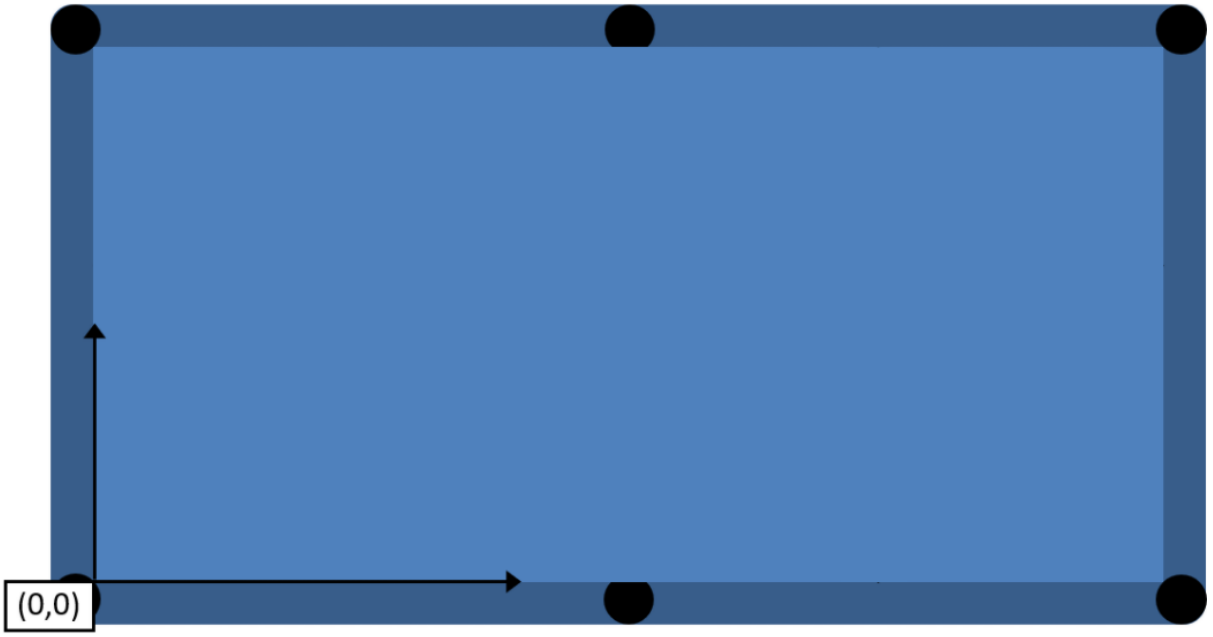
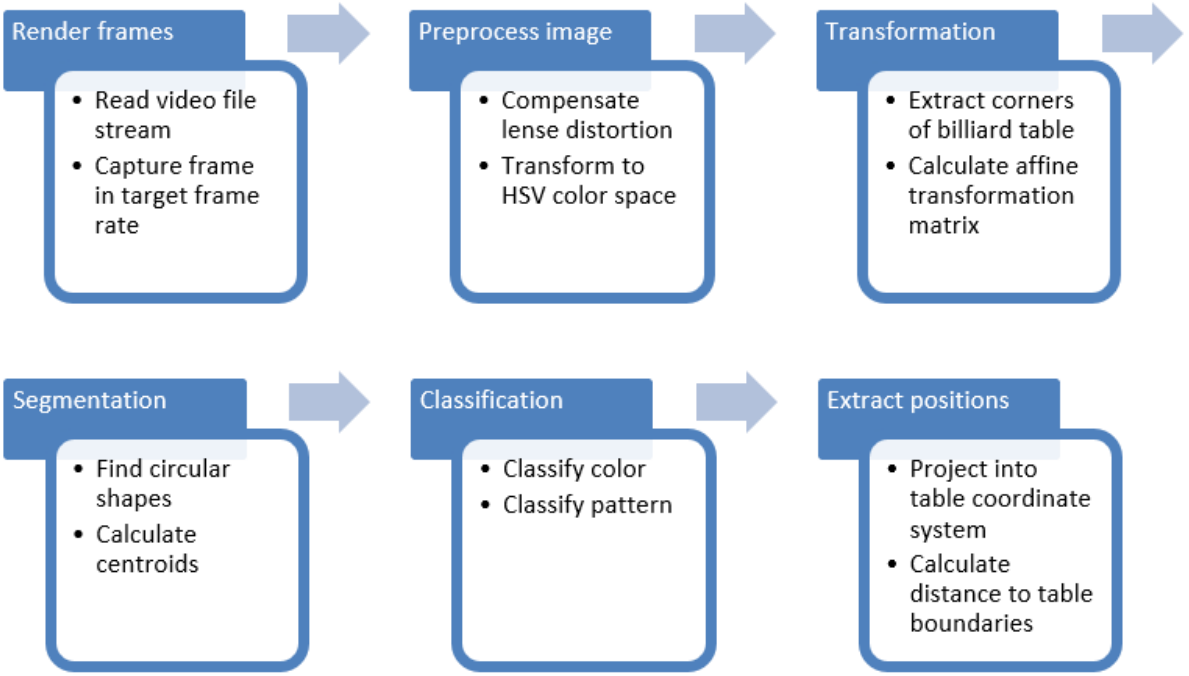


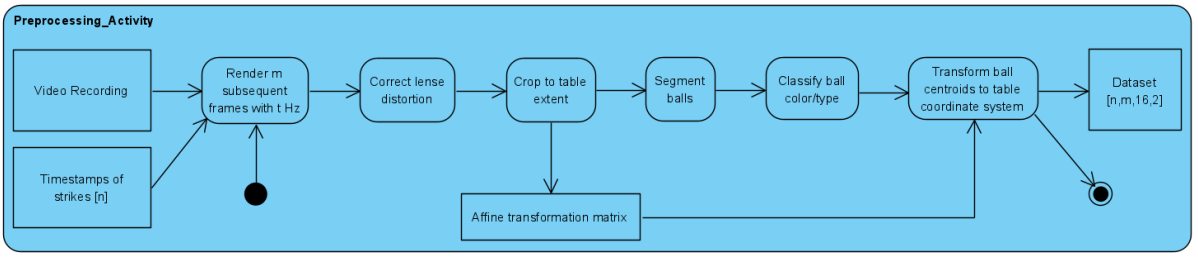
Image Processing

- Based on the initially calculated camera intrinsics (in the first place the lense distortion matrix) the image frame shall be dewarped/rectified, see https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html
- The area around the billiard table shall be cropped
- For each scenario an initial image without occlusions shall be available (see req. in “Recording”) to register all available balls
- Balls not visible in the initial image shall be given a negative position in x and y. This makes them distinguishable from regular ball positions as defined in subsection “Table Coordinate System”
- Occlusion during striking: Occluded balls shall be assumed to be steady
- The positions (as defined in “Table Coordinate System”) of the initially visible cue balls shall be extracted for each frame
- For each detected ball position the color and pattern (solid/stripe) shall be classified.
- The positions from image frame shall be projected to table frame as specified by the affine transformation.

- The pipeline can be described as follows:



- The UML activity diagram looks as follows:



Data augmentation

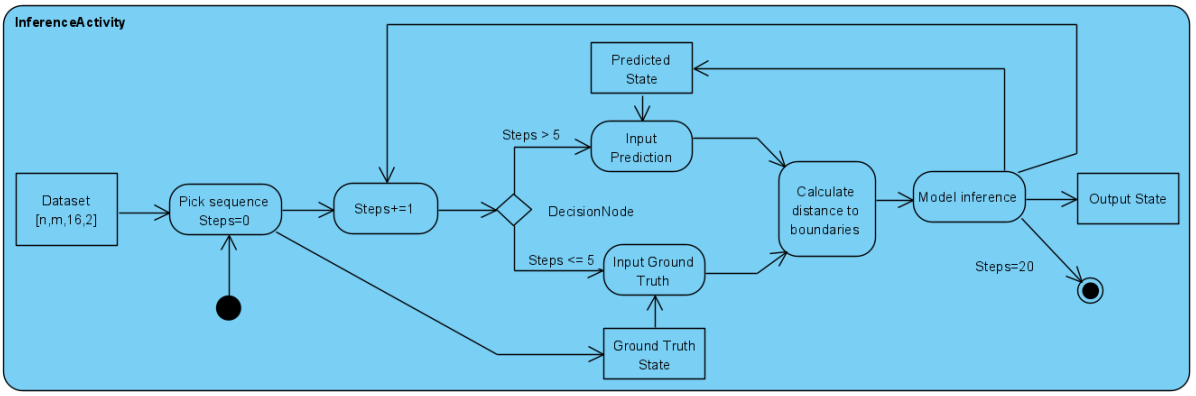
- Mirror dataset horizontally (mirror position values in processed dataset)
- Mirror dataset vertically (mirror position values in processed dataset)
- Swap balls (except white one) → to be implemented in DataLoader
- Change position of balls that are not affected in a scenario to positions which don't have collisions in this scenario (maybe unrealistic, because we don't know the trajectory in the form of a function, but rather encoded in the model. So we can't know where to place balls to not have an effect).

4. Modeling

Framework

- Data loader functions shall be implemented
- Model loader and store functions shall be implemented
- Implement train/test/validation with dataset
- Training functions shall be implemented

- Build model with hyperparameters to enable quick changes in model makeup and later on a parameter search.
- Create dummy dataset to test data loader, train/test/validation split and training functionality.
- Add GPU support to training and inference code
- Function shall be implemented that handles balls not on table (negative x,y positions)
- The UML activity diagram for inference looks as follows:



Model Architecture

- A traditional ML baseline model shall be implemented (SVM, Random Forrest)
- A neural network baseline model shall be implemented (Linear fully connected layer)
- A deep learning model shall be implemented (LSTM)
- The models shall not process images in an end-to-end fashion because:
    - Blurry estimates (sharp object contours become ellipsoids)
    - Contacting objects might fuse
    - The training data set is too small to learn extracting image data
    - The color distance between balls is different
- The deep learning model shall process a tensor of ball positions as an input and provide a tensor of predicted ball positions as an output (details see subsection “Input/Output”)
- A recurrent neural network shall be implemented to process the time series
- As recurrent cell, an LSMT layer shall be used with 64 hidden units
- Additionally, an input linear layer and an output linear layer shall be applied

Input/Output

- Input 1: Position for 16 balls [6x16] (zeros for balls in pockets)
- Output 1: Position difference x,y for 16 balls [2x16] (position invariance)
- Output 2: Value that defines whether ball is on table for 16 balls [1x16]

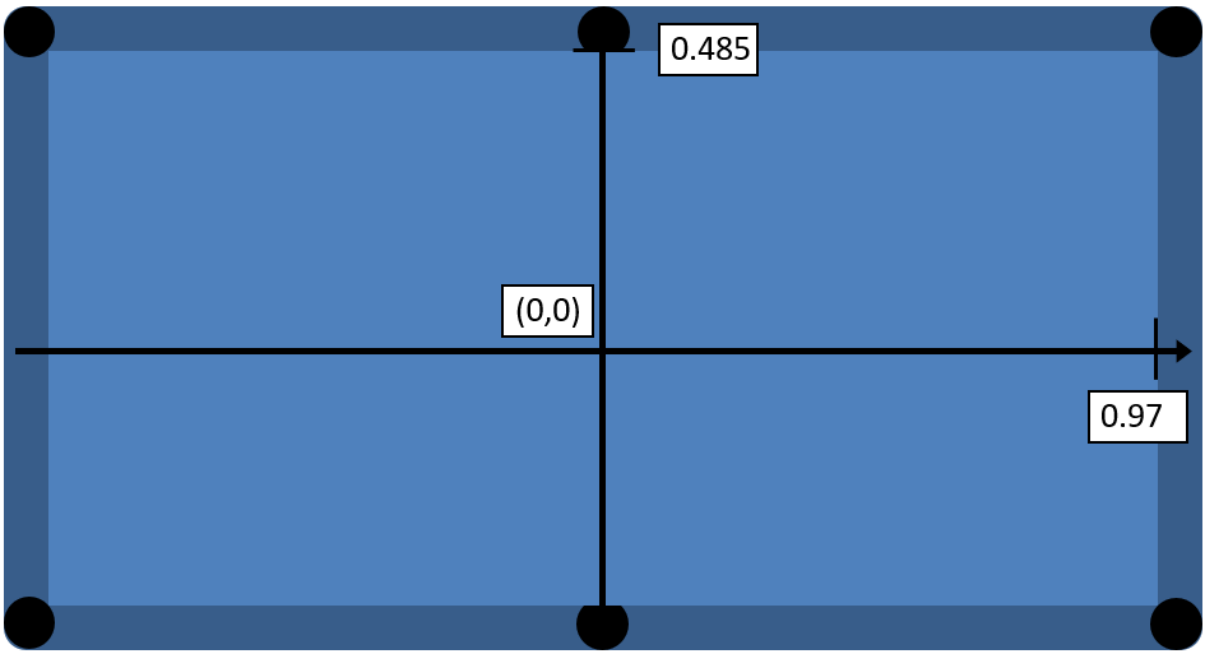
Overcome Data Bottleneck

- Collect as much data as reasonable (at least 500 strikes)
- Apply data augmentation as described in chapter “Preprocessing”
- Apply bias correction to the input data (see subsection “Internal Representation”)

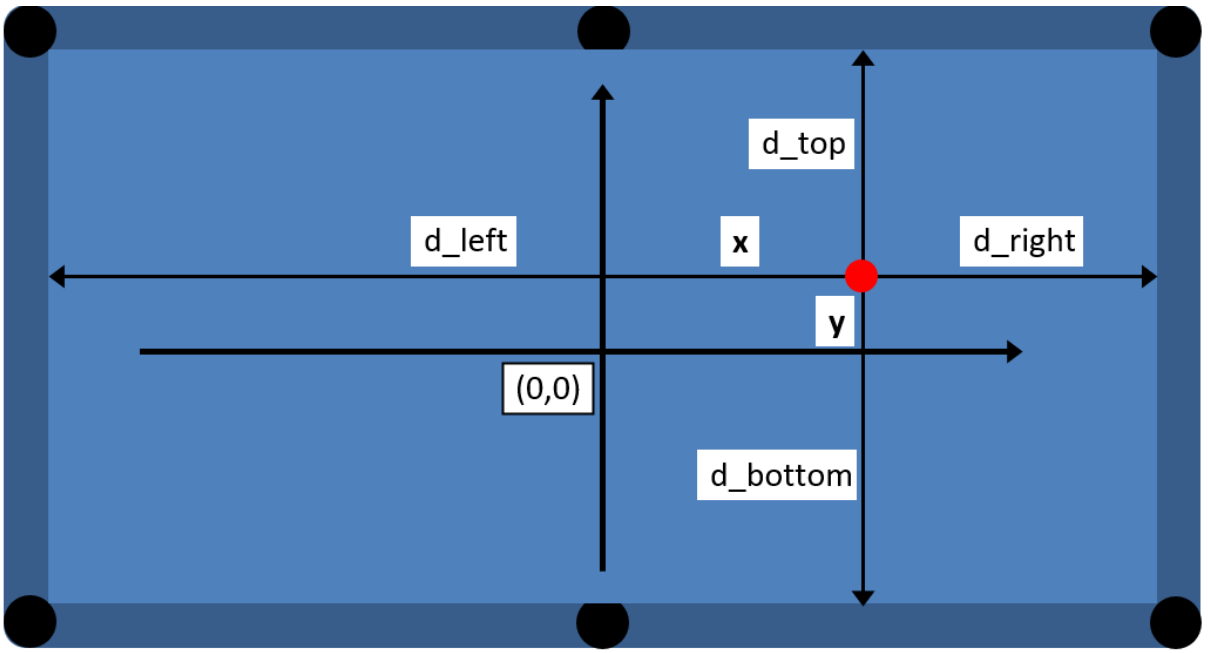
- Provide positions as distance to boundaries (see subsection “Internal Representation”)
- Encode domain knowledge: Define learning task on sparse representation of ball positions instead of dense image representation
- Use neural network models with few parameters (no Transformer)
- Use advanced training techniques (see subsection “Training”)
- Pre-train with synthetic dataset generated from random motion

Internal Representation

- The defined “Table coordinate system” in chapter 3 undergoes changes in representation for the model. Those changes shall be applied when positions are fed into the model and shall be reverted for the positions that the model provides as an output.
- Bias correction: The center of the coordinate system shall be defined in the center of the billiard table with the x axis oriented in direction of the table length and the y axis oriented in direction of the table width (see figure)



- Towards position invariance: A position on the table shall be defined with 6 values (d_top meaning distance to top boundary of table): x, y, d_top, d_right, d_bottom, d_left (see figure)

- 

Training

- L2 norm (euclidean distance, RMSE) shall be used as part one of the loss function
- Cross entropy between predicted and real on_table information shall be used as part two of the loss function
- Exclude from L2 loss the balls that are correctly classified as not on table
- Interpretability: Additionally to the combined loss a loss function shall be provided that represents the average euclidean error distance per ball in x and y dimension
- Optional: Semi teacher forcing: Recursive prediction across 5 time steps based on previous prediction output, then comparison with GT labels to maximize loss and increase punishment for steady state output

5. Evaluation

Evaluation

- The dataset shall be split into 70% training data, 20% validation data and 10% test data
- A cross-fold-validation shall be done to improve hyperparameters
- Evaluation functions shall be implemented
- F1, Accuracy, Root Mean Square Error … metrics shall be implemented
- Comparison with mathematical model shall be carried out

Rendering

- A tool shall render an image of the table view given a list of ball positions
- A tool shall generate a list of images based on a prediction time series
- A tool shall generate a GIF file from the list of images