Image Processing/Ball Detection

The code for the live detection of the ball in the camera feed is located in the files *detection.cpp* and *detection.h* and uses constants from *magic.h* that are being determined from calibration.

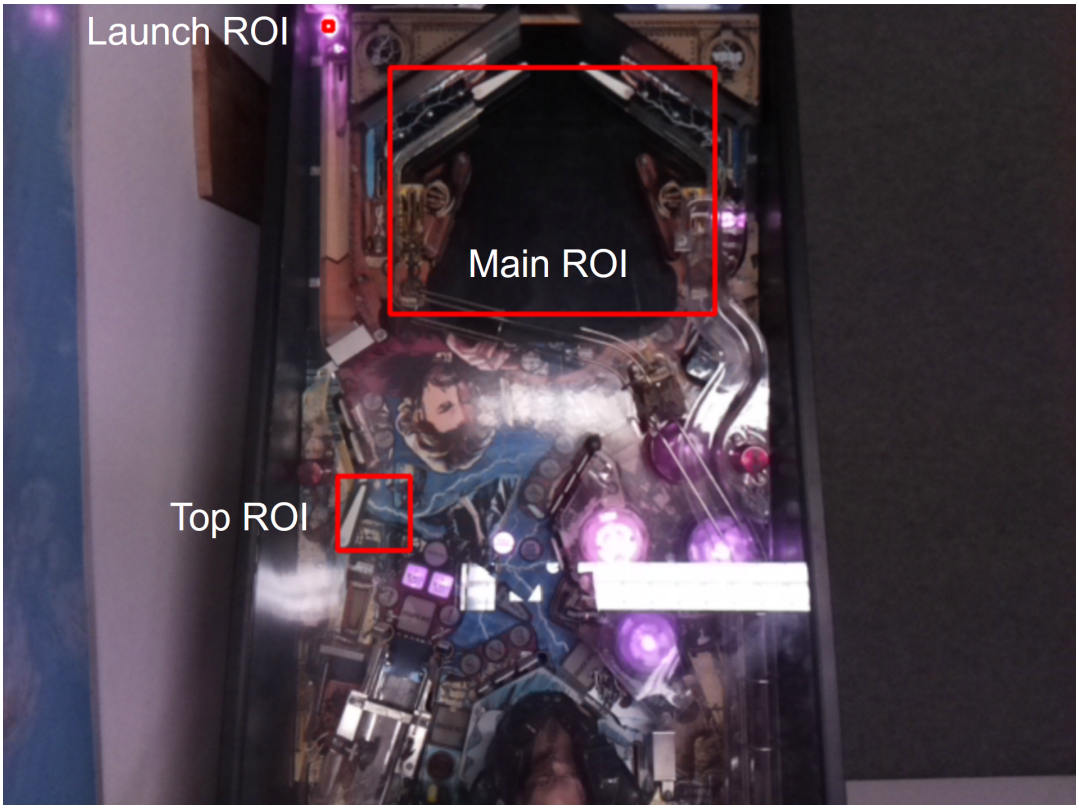
The purpose of this code is:

- detect the ball in one of multiple regions of the camera frame
- output the balls real world coordinates (in meters)
- determine if the launch lamp is flashing (can the ball be launched to resume the game?)
- determine if the flippers are currently up (for timing measurements)

Camera Settings

- 30 Hz refresh rate 33.3 ms between frames
- 640x480 resolution
- 10 ms shutter speed (to avoid deformation and blurring of the ball)

Regions of Interest



There are three so called regions of interest (ROIs) defined in the algorithm.

The main ROI is the important region for keeping the ball in the game. The purpose of the top region is to be able to use the top flipper to deflect the ball into parts of the playfield that are otherwise hard to reach.

The ball detection algorithm can be run on any defined ROI. However, the ROIs currently in use are chosen to include as few lamps as possible (which must be masked or the detection algorithm will be confused) and fit the time window of at most 33.3 ms that is available well. Too many or too large regions will become problematic at some point.

Detection Algorithm

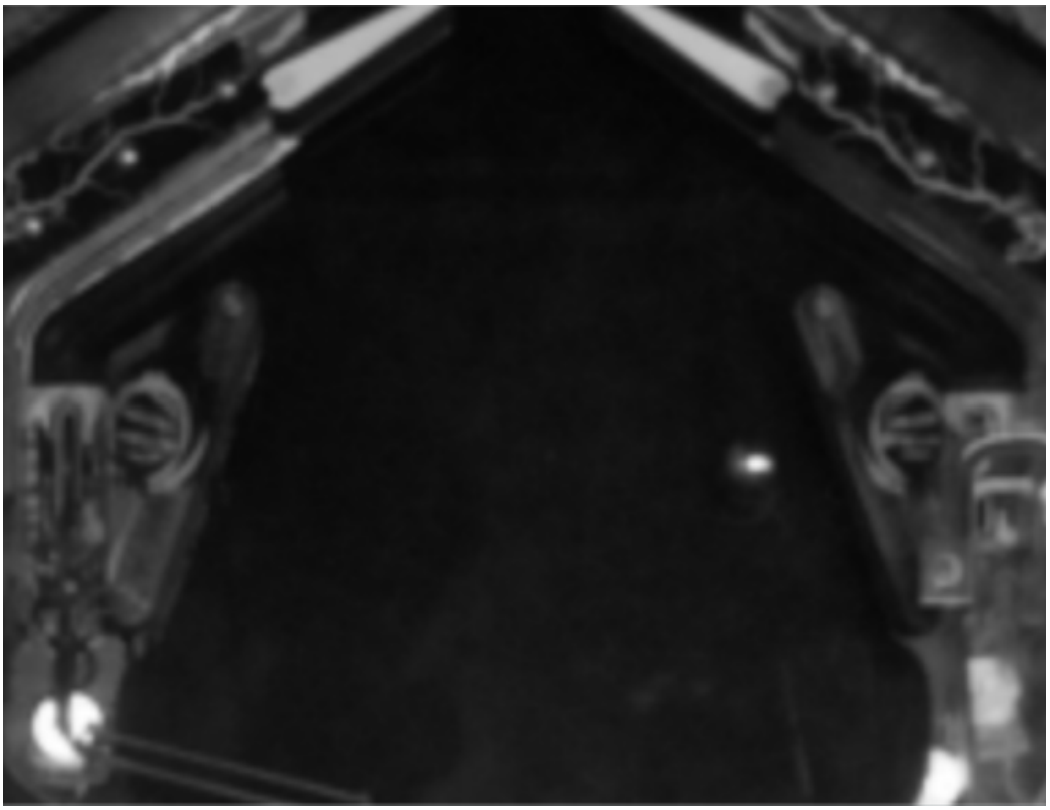
The following is a step by step description and visualization of the actual detection algorithm for the ball.

Cropping

The given ROI is cropped out of the full frame:

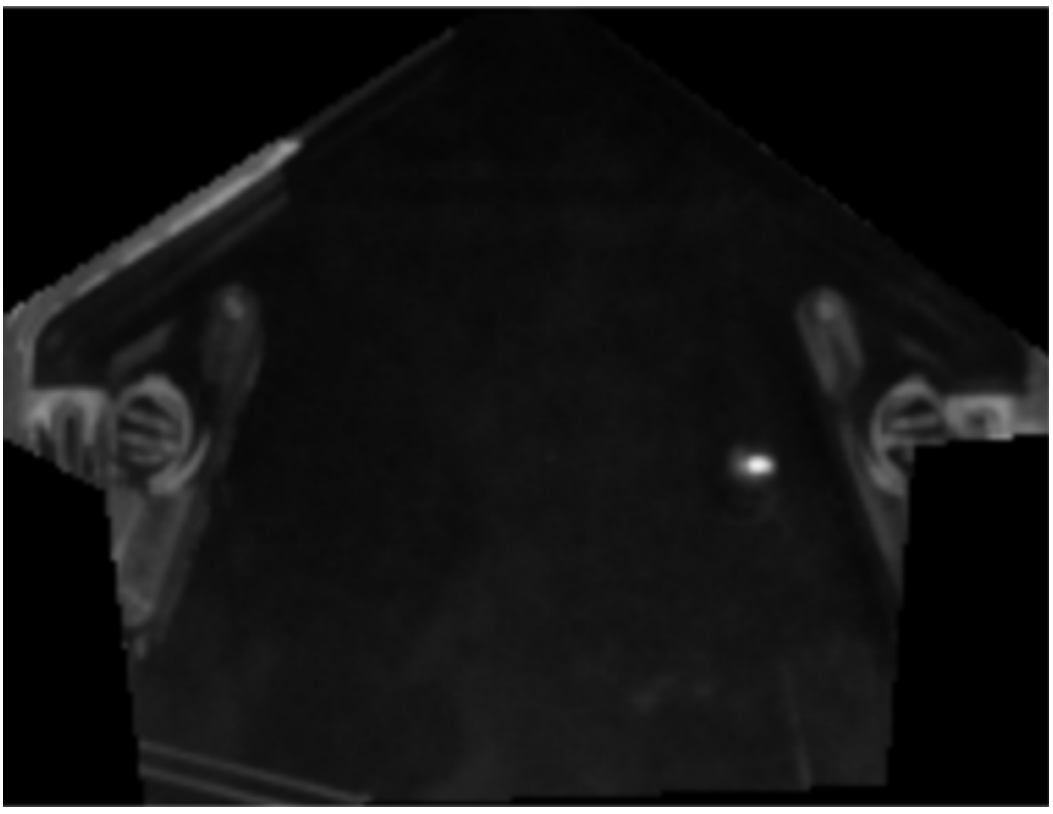
Grey scaling

Three color channels are not required, reduce to single grey channel for more efficient processing.



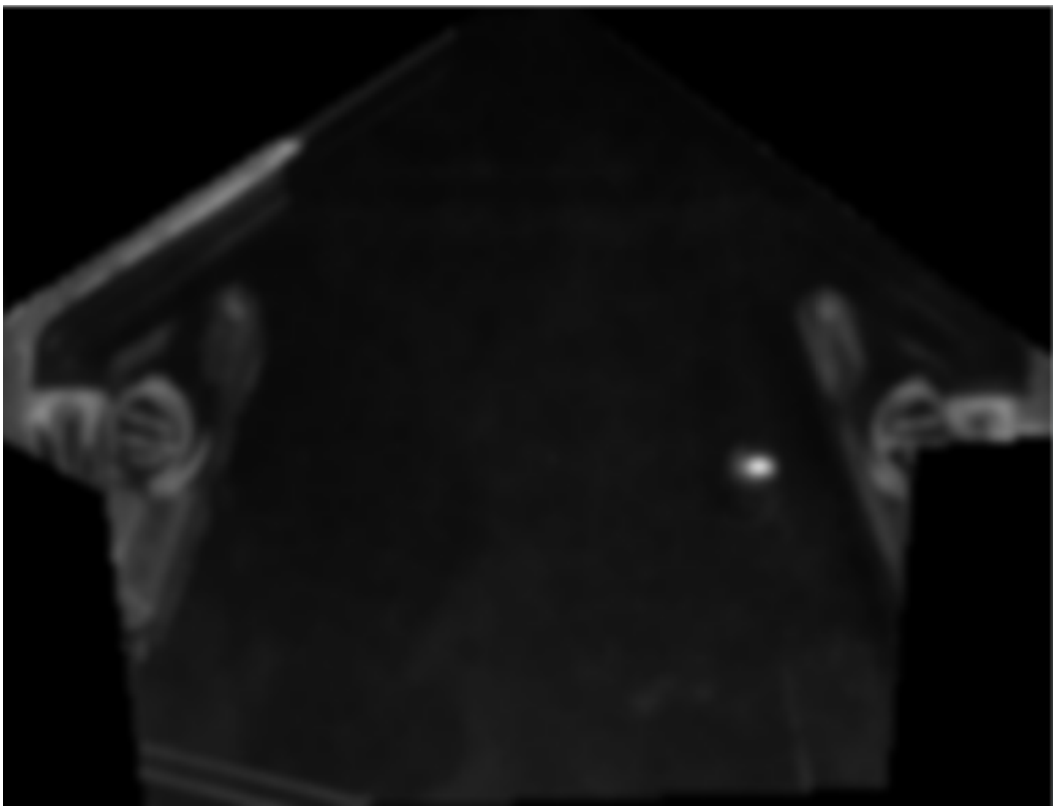
Masking

Ignore lamps and areas outside of the playfield using polygon mask:



Gauss Filter

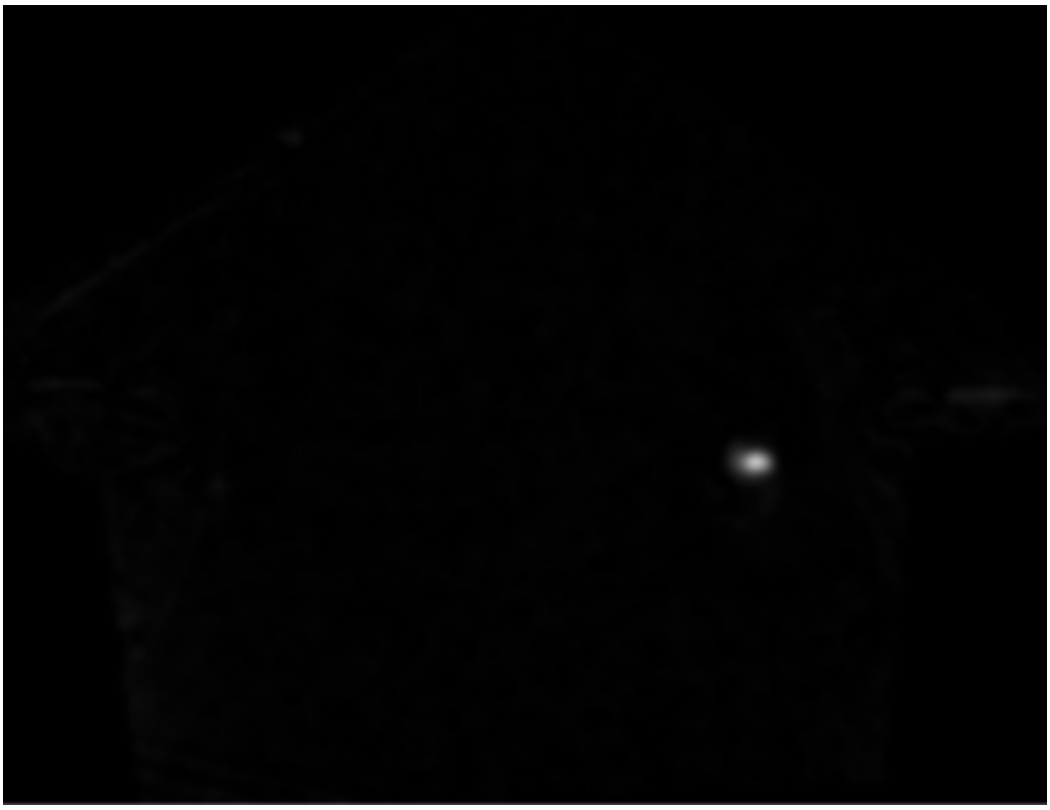
Ignore high frequency noise.



Absolute Difference

Compute pixel wise difference to fixed reference frame, makes movement visible.

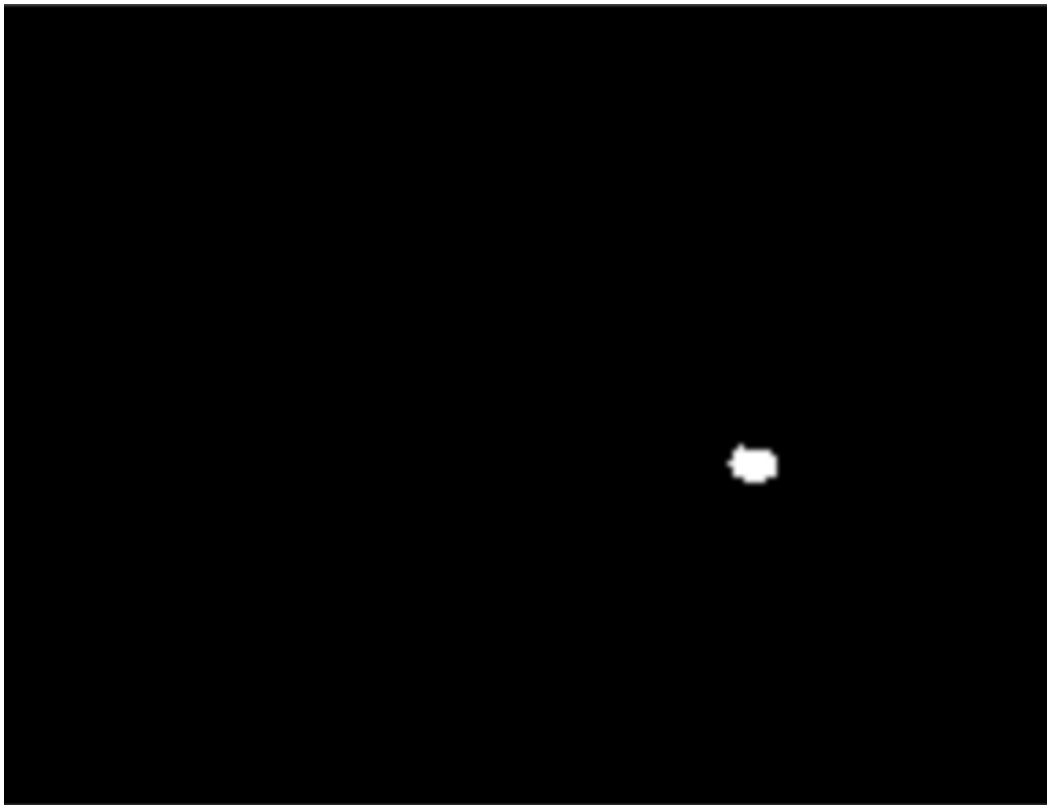
Areas with no movement are black, areas with movement are grey/white:



Thresholding

Set all pixels with gray value above certain threshold to full white, rest to full black

Binary image of moving areas:



Dilation

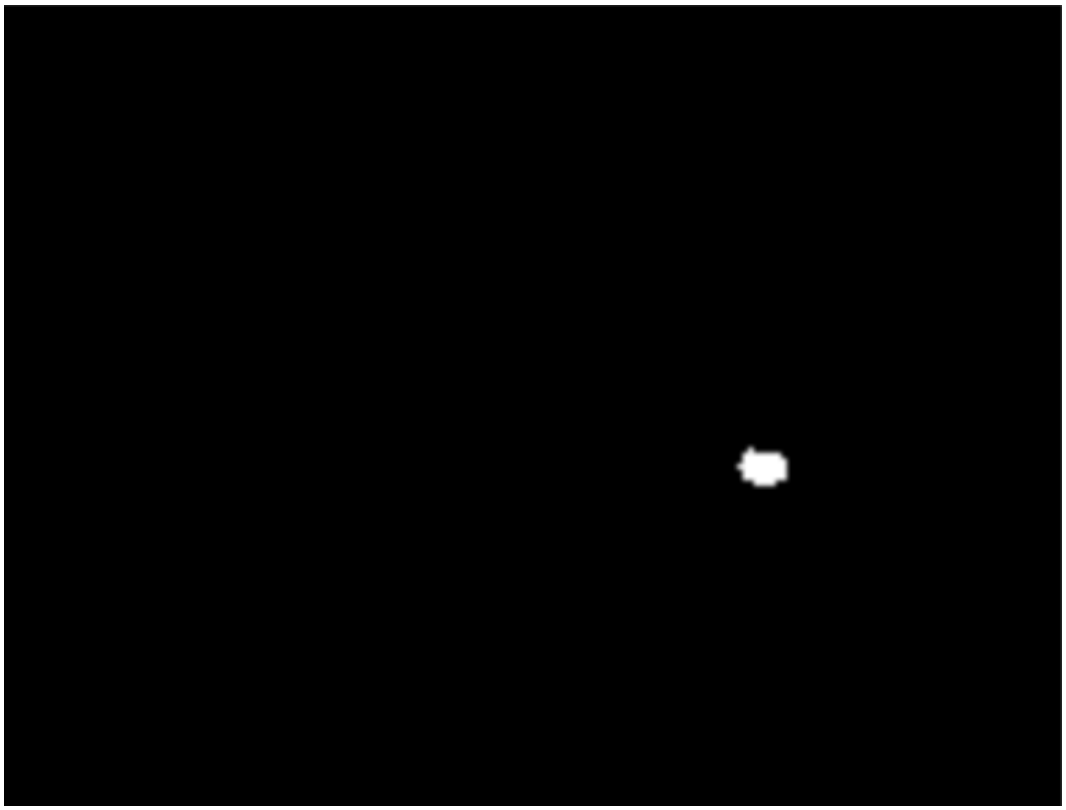
Widen contours by several pixels. Fill in holes, join close areas together.

Erosion

Narrow contours by several pixels. Roughly restores original size of contours but holes remain filled, small gaps remain closed.

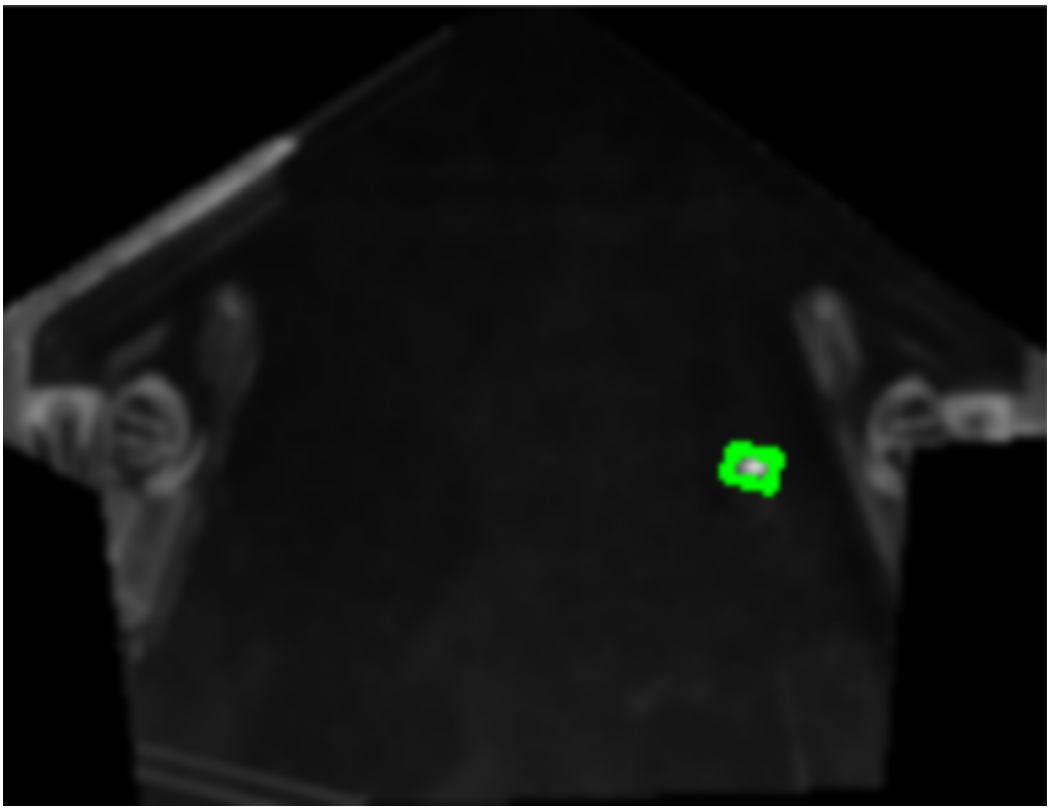
Operation is called “Closing”:

- First Dilation
- Then Erosion



Detection

- Find all contours (here: 1)
- Iterate over contours
- Compute minimal area bounding rectangle
- Drop contours that don’t fit the expected ball contour
    - Contour area
    - Aspect ratio
- Should leave only contour of the ball
- Center of bounding rectangle is ball position in pixel coordinates
- Finally transform pixel coordinates into world coordinates



Additional Processing

Apart from ball detection in the main and top ROI, the algorithm does the following:

Mask Flippers

The algorithm detects if the two bottom flippers are up and will mask them if that's the case. This is done to avoid that the contour of the flipper merges with the ball contour so that the ball isn't detectable anymore.

Detect Launch Lamp Flashing

The launch ROI is a small area at the red lamp that is flashing when a ball is ready to be launched into the game. This is detected by looking at how many pixels in that region are white, if the number is above a certain threshold, the lamp is assumed to be on.

Discarded Approaches

Automatic Masking of Lamps

We tried to observe the whole camera frame with the ball detection algorithm (not only regions) and mask flashing lamps automatically. This has been tried by observing the flashing lamps for a while before the actual game starts and saving the information about the lamps (based on their purple color in the no-IR filter image) in a buffer image that then was used to compute a reference frame free of blinking lamps and also to mask lamps from the current frame. While that did work, it introduced too many blind spots due to the sheer amount of lamps on the playfield, so that not much was gained compared to the approach of just looking at the bottom part of the pinball machine that has a black cover for the lamps. Also the ball was colored in the same purple color as the lamps itself when it passed a flashing lamp, making it almost impossible to detect.