基于神经网络的 GOTURN: Generic Object Tracking Using Regression Networks 是发表在 ECCV 2016 的一篇有关 Tracking 的文章，达到了 Tracking 中效果上的 state-of-the-art，尤其在检测速度上达到了 100FPS（第一个达到 100FPS 的深度学习方法），Code 用 Caffe 编写。该算法对遮挡敏感，但是对视角、形变、光照变化具有鲁棒性。由于冰球游戏场景很少有遮挡情况，故将这种理论模型应用于桌面冰球的球体追踪中。

整个文章的关键点就是 Regression，回归的是什么？当然是 bounding-box 的坐标，那么回归的输入变量就是 current frame，输出为 bounding-box 的坐标。当然前提是知道 previous frame 中 object 的坐标在中心位置。那么这个 Regression Network 学习到的就是：object 在视频中前后帧的 motion 到 object 坐标的变化！知道了 object 在前一帧的中心，找到 object 在当前帧的位置。

利用视频和静态图片训练，优化 predicted bounding-box 和 ground truth bounding-box 之间的 L1 范数损失。训练集中 video 的某些 object 带有 bounding-box，按照前文所述将 search region 给 crop 出来，将两帧和当前帧 bounding-box 的坐标输入网络来优化。在训练的时候也可以在当前帧进行人为的 random crop，以增加 tracker 的鲁棒性、增广数据集。

摄像头数据采集

1. 打开一个文本文件 "config.txt"，读取其中的配置信息。
2. 从文件中读取字符，将其存储在字符数组 aux_str 中。
3. 根据特定字符的位置计算摄像头和串口的配置参数。
4. 打开指定摄像头和串口。
5. 设置摄像头的分辨率和帧率。
6. 创建一个窗口来显示画面。
7. 进入一个无限循环，处理每一帧的图像。
8. 从摄像头获取一帧图像，并进行高斯滤波和颜色空间转换。
9. 根据设定的阈值对图像进行二值化，以分离出冰球和机器人的轮廓。

冰球轨迹追踪

10. trackObjectPuck() trackObjectRobot() 跟踪冰球和机器人的位置。

- 初始化中心坐标
- 划定并框出感兴趣区域
- 找到符合 hsv 阈值的区域
- 遍历所有符合要求的区域，根据像素面积进行筛选
- 计算筛选后的区域的中心点坐标，保存在 puckCenter 中
- 把 puckCenter 坐标从摄像头坐标系转换为机器人坐标系
- 如果是机器人，程序结束；如果是冰球，继续执行以下语句
- 更新冰球当前 XY 坐标，绘制蓝色圆圈表示冰球当前位置
- 读取上一时刻的 XY，画出冰球运动的轨迹

冰球轨迹预测

11. cameraProcess(33.33) 进行冰球轨迹预测。

- 输入参数 time，表示两帧图像之间的时间间隔。
- 计算冰球在两帧图像中的 x 和 y 方向的位移差，存储在 vectorX 和 vectorY 变量中。
- 将当前的冰球速度赋值给上一次的速度变量 puckOldSpeedX 和 puckOldSpeedY。
- 位移差除以时间间隔，更新冰球的速度值 puckSpeedX 和 puckSpeedY
- 将平均速度值赋值给 puckSpeedXAverage 和 puckSpeedYAverage。
- 将 predict_x_attack 初始化为 -1。
- 如果冰球的 y 方向速度足够大，表示冰球纵向移动，进入追踪逻辑。
- 计算冰球的轨迹斜率 slope。
- 用 GOTURN 算法计算冰球最终目标点的坐标 predict_x 和 predict_y，通过将冰球的当前坐标和斜率带入公式得到。
- 如果预测的最终目标点超出了侧边界，表示冰球与侧边发生碰撞。
- 计算冰球与侧边界碰撞点的坐标 bounce_x 和 bounce_y。
- 计算冰球与侧边界碰撞后的预测时间 predict_time。
- 根据碰撞后的斜率、碰撞点和预测目标点的 y 坐标，重新计算预测目标点的坐标。
- 如果预测的目标点仍然超出侧边界，表示冰球与另一侧边界发生了第二次碰撞，此时不进行新的预测，因为反弹两次后一般来说速度很低，没什么威胁
- 如果预测的目标点在侧边界范围内，计算最终预测目标点的坐标，并更新预测时间。
- 在图像上绘制冰球轨迹的线段。
- 如果冰球速度较慢或向另一边移动，将预测状态和相关变量重置为初始状态。

冰球游戏策略（衔接运动控制系统）

12. newDataStrategy() 根据当前状态给机器人制定一个策略。

- 将机器人状态设为回到初始位置（robot_status = 0）。
- 如果冰球预测状态为 1（冰球直冲过来），判断冰球预测的 x 坐标和时间，确定机器人的状态：
  - 如果冰球预测的 x 坐标在 151 和 323 之间，并且预测时间小于 300ms，将机器人状态设置为被动进攻模式（robot_status = 2）。
  - 如果冰球预测的 x 坐标在 151 和 323 之间，将机器人状态设置为防御模式（robot_status = 1）。
  - 如果以上条件均不满足，将机器人的 x 坐标限制在 111 和 363 之间，并将机器人状态设置为防御模式（robot_status = 1）。

- 如果冰球预测状态为 2（冰球碰撞到了墙壁），将机器人的 x 坐标限制在 101 和 373 之间，并将机器人状态设置为防御模式（robot_status = 1）。
- 如果冰球预测状态为 0，并且冰球的 y 坐标小于 180（在机器人区域缓慢移动），则进行主动进攻：
  - 如果攻击时间为 0，则预测冰球 400ms 后的位置，并判断该位置是否在一定区域内。如果在区域内，设置攻击时间为当前时间加上 400ms，并将机器人位置设置到冰球后方，同时将攻击状态设置为 1。如果不在区域内，继续等待。
  - 如果攻击时间不为 0，根据攻击状态进行相应的操作：如果攻击状态为 1（已经在冰球后方，准备进攻），并且攻击剩余时间小于 200ms，进行进攻动作，并将攻击状态设置为 2（进攻完成）。如果攻击状态为 1，但攻击剩余时间大于 200ms，将机器人位置设置到准备进攻区域。如果攻击状态为 2（进攻完成），并且当前时间超过攻击时间加上 150ms，重置攻击时间和机器人状态，并将攻击状态设置为 0。
- 最后，将机器人状态设置为主动进攻模式（robot_status = 3）。

13. 通过串口发送控制指令给机器人。
14. 计算帧率 fps 值并在图像上显示。
15. 显示处理后的图像。
16. 等待用户按键，如果按下了 ESC 键，则退出循环。