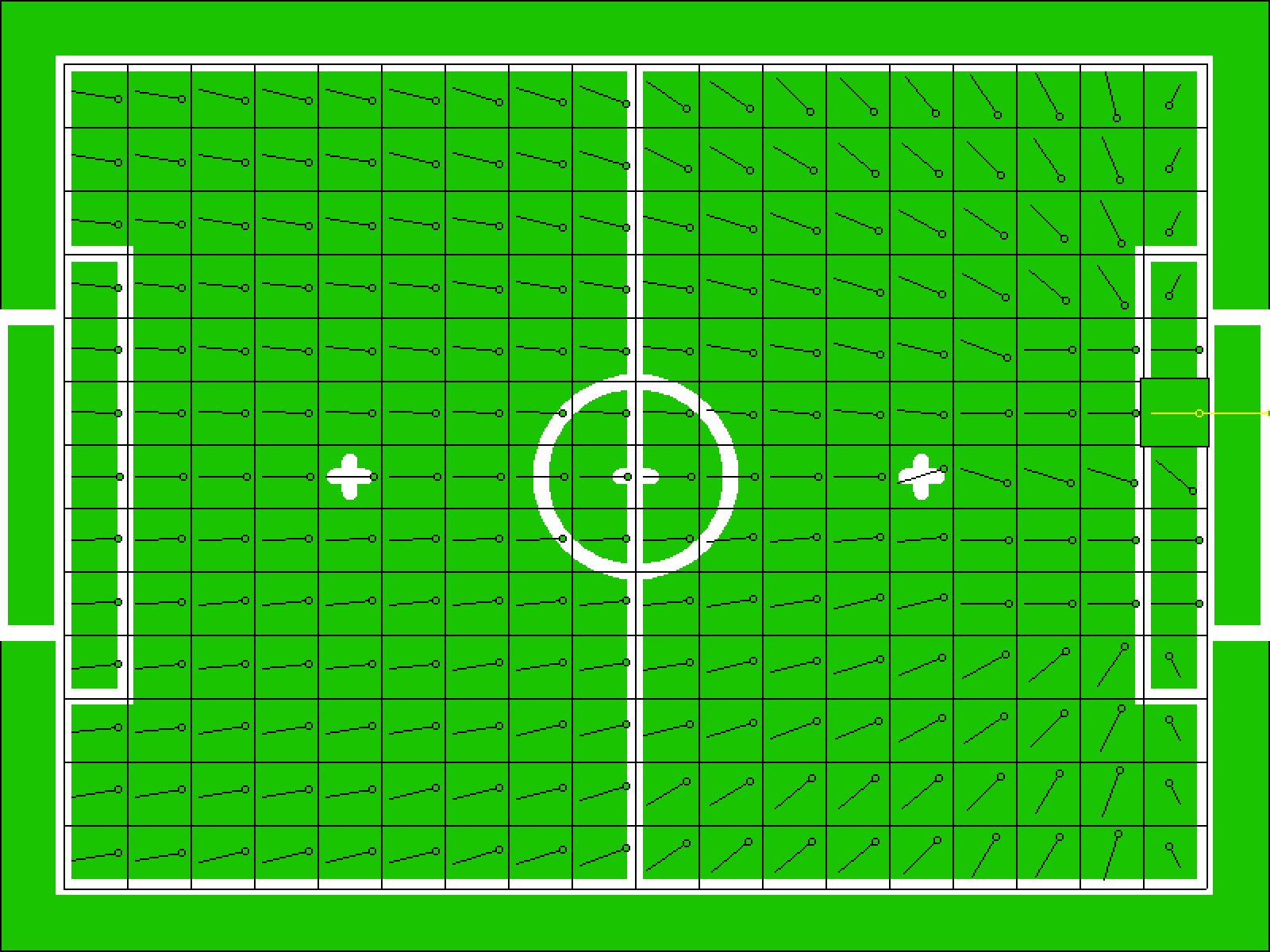
ELSIROS

Entry\_Level\_SImulation\_of\_RObot\_Soccer

Руководство программирования Робота Robokit-1

1. Не рекомендуется играть игру между полностью одинаковыми командами. В большинстве случаем вы будете наблюдать неинтересную игру с малым количеством голов и безрезультатной борьбой. Поэтому мы вам предоставляем 2 разных алгоритма игры: нормальный и старого стиля, который использовался в 2020 году.
2. Реальные роботы используют смарт камеру OpenMV H7 в качестве визуального сенсора и в качестве вычислительного модуля. Это одноядерный контроллер со средой программирования Micropython без операционной системы. Это означает, что контроллер не позволяет одновременно использовать функцию хождения и получения видеокадра. Для обновления информации о нахождении мяча и само-локализации робот должен остановиться в стабилизированной вертикальной позиции и поворачивать голову в разные сектора обзора для осмотра окрестностей. При поворотах головы положение мяча может быть считано, если мяч оказался в видимом секторе камеры. Апертура камеры 46 градусов.
3. В процессе обзора камерой в вертикальной позиции робот также получает информацию, полезную для локализации, такую как стойки ворот, разметка поля и граница зеленого поля. Чем больше кадров обработано с разных ракурсов, тем точнее информация о локализации.
4. Робот может обновлять информацию о препятствиях из полученных кадров. В алгоритм построения траекторий включен алгоритм обхода препятствий, который не совсем совершенен. Нет планирования направления удара с учетом препятствий на пути мяча. Список self.glob.obstacles содержит информацию об обнаруженных и обновленных данных о препятствиях.
5. Коммуникация между игроками разрешена по правилам через UDP сообщения WIFI. Коммуникация не реализована в текущей стратегии, но команды могут её разрабатывать. Коммуникация внутри команды может помочь организовать более эффективную игру. ELSIROS API содержит средства коммуникации между членами команды.
6. Координатное пространство, используемое в стратегии, отличается от абсолютного координатного пространства в симуляции. Для целей стратегии собственные ворота располагаются в части поля с отрицательными значениями координаты X, ворота противника располагаются в части поля с положительными значениями координаты X. Положительные значения Y расположены на левом фланге атаки, отрицательные значения Y расположены на правом фланге атаки. Азимут yaw имеет нулевое значение, если направлен из центра поля к воротам противника. Yaw изменяется от 0 к math.pi при вращении вокруг вертикальной оси влево от нулевого направления. Yaw изменяется от 0 к -math.pi при вращении вокруг вертикальной оси вправо от нулевого направления. Координата Z направлена вверх с нулём на поверхности поля.
7. Нормальный игрок “forward” использует предварительно запрограммированную стратегию, заложенную в матрицу векторов. Матрица закодирована в файле strategy\_data.json . Этот файл является читаемым и редактируемым так же как и обычный текстовый файл. В файле содержится словарь с одним ключом “strategy\_data”. Под ключом “strategy\_data” содержится список с числом элементов по умолчанию 234. Каждый элемент списка представляет квадратный сектор поля размерами 20cmX20cm. Каждому сектору назначен вектор представляющий из себя направление удара в ситуации когда мяч оказался в этом секторе. Сила удара закодирована в виде ослабляющего коэффициента: 1 – стандартный удар, 2 – удар ослабленный в 2 раза, 3- удар ослабленный в 3 раза. Каждый элемент списка закодирован следующим: [столбец, ряд, сила, yaw]. Футбольное поле разбито на 13 рядов и 18 столбцов. Столбец 0 находится возле своих ворот, столбец 17 находится возле ворот противника. Ряд 0 находится в области с положительной Y координатой, ряд 12 находится в области с отрицательной Y координатой.



1. В процессе игры игрок может принимать 4 разные роли: ‘forward’, ‘goalkeeper’, ‘penalty\_Shooter’, ‘penlaty\_Goalkeeper’. Для каждой роли программа стратегии разная. Модуль launcher выбирает роль игрока для запуска в зависимости от номера игрока и вторичного состояния игры. В случае если номер игрока 1 назначаемая роль будет ‘goalkeeper’ или ‘penalty\_Goalkeeper’. В случае, если номер игрока отличается от 1, тогда назначаемая роль ‘forward’ или ‘penalty\_Shooter’. В случае secondary game state = ‘STATE\_PENALTYSHOOT’ тогда игрок с номером 1 будет назначен ‘penlaty\_Goalkeeper’, а игрок с номером, отличающимся от 1 будет назначен на роль ‘penalty\_Shooter’. Котроллер игры, поставляемый по умолчанию будет назначать игроку с номером 1 ‘goalkeeper’, а игроку с номером, отличающимся от 1 роль ‘forward’ во всех остальных значениях secondary game state. Команды могут менять стратегию игры путем назначения различных ролей в зависимости от величины secondary game state, поставляемой геймконтроллером. Текущий геймконтроллер может поставлять следующие значения secondary game state: STATE\_NORMAL=0, STATE\_PENALTYSHOOT=1, STATE\_OVERTIME=2, STATE\_TIMEOUT=3, STATE\_DIRECT\_FREEKICK=4, STATE\_INDIRECT\_FREEKICK=5, STATE\_PENALTYKICK=6, STATE\_CORNERKICK=7, STATE\_GOALKICK=8, STATE\_THROWIN=9, DROPBALL=128, UNKNOWN=255