**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

КУРСОВАЯ РАБОТА

На тему: Тулбокс для захвата и толкания предметов эндеффектором манипулятора с самостоятельным обучением с подкреплением

Выполнили:

Ларченков Артем  
 Холодилов Алексей

Санкт-Петербург

2023 г.

**Оглавление**

[Введение: 3](#_heading=h.30j0zll)

[Формулировка проблемы: 5](#_heading=h.1fob9te)

[Методология решения: 5](#_heading=h.3znysh7)

[Эксперимент: 5](#_heading=h.2et92p0)

[Заключение: 5](#_heading=h.tyjcwt)

[Список используемой литературы: 5](#_heading=h.3dy6vkm)

1. Введение:

Цель грамотного манипулирования эндеффектором заключается в комбинировании толкательными и хватательными операциями. Толкание может помочь отодвинуть предметы заграждающие объекты, которые должны быть захвачены, также, захват предметов может помочь перемещать объекты, чтобы сделать толкающие движения более точными и избежать столкновений. Хотя планированию толкающего и хватающего движений было посвящено значительное исследование, они преимущественно изучались изолированно. Сочетание подталкиваний и захватов для последовательного изменения положения объектов является относительно неизученной задачей. Отталкивание традиционно изучается для решения задачи точного контроля положения объекта. Однако в уже существующих системах, сочетающих, толкание и захват, подталкивание реализовано достаточно слабо, для разделения двух объектов, для освобождения пространства в определенной области или для разрушения группы объектов. Эти цели трудно использовать для обучения с подкреплением. Большинство последних опубликованных работ используют только метод захвата, ввиду его наглядности для машинного обучения и высокой повторяемости.

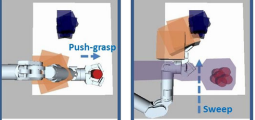


Рис. 1 Хватательное и толкательное движения эндеффектора .

Однако остается неясным, как планировать последовательности действий, сочетающие захваты и отталкивания, ведь каждое из этих движений изучается изолированно. В этой работе мы предлагаем обнаружить и изучить способы взаимодействия между подталкиванием и захватом на основе опыта с помощью обучения с глубоким подкреплением с open source моделями такими как Q-learning.

Основными аспектами системы, над которой мы работаем являются:

• Изучаем методологию совместного толкания и захвата методом проб и ошибок с самообучением. Толкающие действия полезны только в том случае, если со временем они позволяют захватывать. Это противоречит предыдущим подходам, которые определяют взаимодействие методов или жестко запрограммированные алгоритмы для толкающих движений.

• Мы обучаем нашу модель от начала до конца с помощью глубокой нейронной сети, которая принимает визуальные наблюдения и выдает оценку в виде значений Q для потенциальных действий по подталкиванию и захвату. Затем алгоритм совместной работы выбирает действие с наибольшим значением Q, то есть то, которое максимизирует ожидаемый успех текущих / будущих захватов.

Это контрастирует с явным восприятием отдельных объектов и планированием действий над ними на основе разработанных вручную функций. Эти формулировки позволяет нашей системе выполнять сложные последовательные манипуляции с толканием и захватом объектов в неструктурированных сценариях выбора и позволяет включать в себя новые объекты, те, которые не использовались при обучении.

Обучение при помощи глубоких нейронных сетей алгоритмам совмещения двух типов взаимодействия с объектами с подкреплением в физических системах всегда будет слишком дорогостоящим и продолжительным по времени из-за очень высокой сложности выборки и её разнообразия. Чтобы сделать обучение управляемым на реальном роботе, мы упростили пространство действий до набора движущих простейших объектов, управляемых эндеффектором. Мы формулируем задачу как проблему маркировки по пикселям: где каждый пиксель изображения и ориентация изображения соответствует определенному типу движения робота толкание или захват. Всё выполняется в 3D-пространстве этого пикселя в сцене. При толкании это положение представляет собой начальное положение толкающего движения, а при захвате - среднее положение между «пальцами». Эта пиксельная параметризация простейших действий робота, которые мы называем полностью сверточными функциями взаимодействия, позволяют нам эффективно обучать методологию толкания и захвата на одной руке робота менее чем за несколько робо-часов. Основным вкладом этой статьи является новый взгляд на объединение управляемых данными толкательных и хватательных манипуляций.

Мы считаем, что можно обучать сверточные глубокие сети для взаимодополняющих движений, которые вместе дополняют друг друга. Мы предлагаем несколько моделей системы, которые в условиях симуляции служат для оценки ключевых параметров нашей системы. Наши результаты показывают, что использование метод подталкивания совместно с захватами расширяет набор сценариев и что оба типа взаимодействия с объектами, работающие вместе, создают сложные взаимодействия, в том числе и нами изначально не прогнозируемые, такие как: одновременное перемещение нескольких блоков, разделение двух объектов, разбиение группы объектов с помощью цепочки реакций.

1. **Исследование уже существующих решений:**

Наша работа лежит на пересечении стандартных для мехатроники задач: компьютерного зрения и машинного обучения мы кратко пройдемся по этим областям без излишнего углубления.

Планирование движений, не связанных с хватательными движениями, таких как толкание, является одной из старейших проблем, она стала актуальна с первой волной просчётов основных роботизированных манипуляций. Литература в этой области обширна, она рано появилась на основе классических решений, которые явно моделируют динамику толчка силами трения, хотя многие из этих методов и кажутся достаточно рабочими, но они основаны на допущениях моделирования, которые не выполняются на практике, а следовательно мы не можем использовать их в работе. Например, неравномерное распределение трения по поверхностям объектов и изменчивость трения — это лишь некоторые из факторов, которые могут привести к ошибочным прогнозам решений для моделирования трения в реальных условиях. В то время как сейчас набирают популярность работы, в которых исследуются алгоритмы, основанные на данных, для изучения динамики нажатия, многие из этих работ в значительной степени сосредоточены на выполнении стабильных нажатий для одного объекта за раз. Моделирование крупномасштабных последствий расталкивания в условиях сильного беспорядка и изменения трения по-прежнему остается сложной проблемой.

Захваты эндеффектором также хорошо изучены в области рассуждений, основанных на моделях; от моделирования контактных сил и их сопротивления внешним рывкам до характеристик захватов по их способности ограничивать подвижность объекта. Общий подход к реализации этих методов в реальных системах включает предварительное вычисление захватов из базы данных известных 3D-моделей объектов и индексирование их во время выполнения с регистрацией облака точек для оценки положения объекта. Однако эти методы обычно предполагают знание форм объектов, поз, динамики и точек соприкосновения – информации, которая редко известна для новых объектов в неструктурированных средах. Более современные методы, основанные на данных, исследуют перспективы обучения не зависящим от модели стратегиям глубокого понимания которые обнаруживают захваты, используя изученные визуальные особенности, и без явного использования знаний, специфичных для объекта таких как: форма, поза, динамика. Если повысить эффективность этих глубоких политик, используя модели предварительно обучены вспомогательным задачам, таким как касание.

Есть исследования, которые демонстрируют, что использование движений касания (схожее с толканием) эффективное моделирования этих методов взаимодействия с учетом доступности может значительно сократить время выполнения. Аналогично этим методам, наша платформа, основанная на данных, не зависит от модели, но с добавлением улучшения производительности захвата за счет включения толкательных действий. Толкание и захватом сочетание этих методов манипулирования представляет интерес, хотя это область исследований, которая была гораздо менее изучена. Существует работа Догара1 «A planning framework for nonprehensile manipulation under clutter and uncertainty» представляет надежную структуру планирования для толкающих захватов (толкающих движениях, ограниченных в простейших движениях захвата), чтобы уменьшить неопределенность захвата, а также дополнительный элемент движения «подметание» или расталкивание для беспорядочного перемещения объектов, представляющих собой препятствия. Однако методы исследования Догара по-прежнему в значительной степени разрабатывается вручную. В отличие от этого, наш метод основан на данных и изучается онлайн путем самообучения.

Так же нельзя упомянуть методы, которые исследуют планирование толкающих движений без модели для перемещения объектов в целевые положения, которые более благоприятны для заранее разработанных алгоритмов захвата – поведение которых, как правило, создается вручную, фиксируется и хорошо известно заранее. Эти знания в первую очередь используются для определения конкретных целей (например, целевых позиций), которые могут помочь в разработке или обучении политике продвижения. Однако попытка определить аналогичные цели для политик, основанных на данных, не зависящих от модели, где оптимальное поведение определяется опытом, становятся нереалистичными, поскольку эти методы управления постоянно обучаются, изменяются и адаптируют поведение с течением времени с большим количеством данных. Более тесно связана с нашей работой «Learning to manipulate unknown objects in clutter by reinforcement» В ней исследуется использование обучения с подкреплением для обучения стратегиям контроля для выбора между толчком и захватом. В этой работе, рассматривается возможность создания «конвейера», который сначала сегментирует изображения на объекты, предлагает действия подталкивания и/или захвата, извлекает настроенные вручную функции для каждого действия, а затем выполняет действие с наивысшим ожидаемым вознаграждением. Будучи реализованным на практике, их метод основан на создании модели для прогнозирования движения толкаемых объектов и вывода его в наилучшее положение для будущего захвата, такие прогнозы являются двумя “функциями”, предоставляемыми типу толкательного движения, такой алгоритм рассчитан на работу в основном с выпуклыми объектами, и продемонстрирован только в одном сценарии только с двумя объектами, цилиндр рядом с коробкой. Напротив, мы обучаем методологии восприятия и контроля с помощью сквозных глубоких сетей, мы не делаем никаких предположений о формах или динамике объектов и демонстрируем, что наш алгоритм работает не только для множества тестовых случаев с многочисленными объектами вплоть до 30, но и что он способен быстро обобщаться на новые объекты и сценарии. Насколько нам известно, наша работа — это первая система без шаблонных моделей движения, позволяющая выполнять обучение с подкреплением, основой которой являются подталкивания и захваты, которые работают от начала до конца, от визуальных наблюдений до действий.

Методология решения:

Использование в коде программы возможность выбора алгоритма обучения: обучение с учителем или обучение с подкреплением (Q-learning). Что мы делаем (плюс немного кода для наглядности):

Мы моделируем каждое состояние st как RGB-D изображение с высотой представление сцены в момент времени t. Чтобы вычислить эту карту высот, мы снимаем RGB-D изображения с камеры, установленной неподвижно, проецируем данные на 3D облако точек и ортографически обратно проецируем вверх в направлении силы тяжести, чтобы построить представление карты высот с цветным (RGB) и высотой от низа (D) каналами. Края карты высот предопределены относительно границ рабочей области агента для сбора. В наших экспериментах эта область охватывает поверхность столешницы размером 0,4482 м. Поскольку наши карты высот имеют пиксельное разрешение 224×224, каждый пиксель пространственно представляет 2x2 мм вертикальный столбец 3D-пространства в рабочем пространстве агента.

Далее, мы параметризуем каждое действие как примитив движения поведение ψ (например, толкание или хватание), выполняемое в 3D местоположении q, спроецированном на пиксель p изображения карты высот представления состояния st:

a = (ψ, q) | ψ ∈ {push, grasp}, q → p ∈

Наши примитивы движений определяются следующим образом.

Толкание: q обозначает начальную позицию 10-сантиметрового толчка n в одном из k = 16 направлений. Траектория толчка прямая. При моделировании он физически выполняется с помощью кончика закрытого двухпальцевого захвата.

Хватание: q обозначает среднюю позицию хватания сверху вниз параллельно-челюстным захватом в одной из k = 16 ориентаций. Во время попытки захвата оба пальца пытаются переместиться на 3см ниже q (в направлении силы тяжести), прежде чем сомкнуть пальцы. В обоих примитивах планирование движения руки робота выполняется автоматически с помощью стабильных, свободных от столкновений IK-решений.

Мы расширяем глубокие Q-сети (DQN), моделируя нашу Q-функцию как две полностью сверточные сети (FCN) и ; по одной для каждого примитива движения (толкание и хватание соответственно). Каждая отдельная FCN φ, ψ принимает на вход изображение карты высоты состояния и выдает плотную попиксельную карту значений Q с тем же размером и разрешением изображения, что и st, где каждое отдельное предсказание значения Q в пикселе p представляет собой будущее ожидаемое вознаграждение за выполнение примитива . примитива ψ в 3D месте q, где q→p ∈ . Заметим, что эта формулировка является прямым объединением Q-обучения с визуального манипулирования на основе доступности.

Обе нейросети и имеют одинаковую архитектуру сети: два параллельных 121-слойных DenseNet, предварительно обученных на ImageNet, затем поканальная конкатенация и 2 дополнительных 1 × 1 сверточных слоя, чередующихся с нелинейными функциями активации (ReLU) и пространственной пакетной нормализацией, затем проводится билинейное апсемплирование. Одна DenseNet принимает на вход цветовые каналы (RGB) карты высот, в то время как другая принимает на вход канальный клонированный канал глубины (DDD) (нормализованный путем вычитания среднего и деления стандартного отклонения) карты высот.

Чтобы упростить обучение ориентированных примитивов движения для толкания и хватания, мы учитываем различные ориентации путем поворота входной карты высот st на k = 16 ориентаций (все кратные 22.5 градусам) и рассматриваем только горизонтальные толчки (вправо) и хватания на повернутых картах высот. Таким образом, на вход каждой FCN φ, ψ поступает k = 16 повернутых карт высот, а на выходе получается 32 пиксельные карты значений Q (16 для толчков в разных направлениях и 16 для захватов в разных ориентациях). Действие, которое максимизирует Q-функцию - это примитив и пиксель с наибольшим значением Q по всем 32-м pixel-wise картам.

Pixel-wise параметризация пространств состояний и действий позволяет использовать FCN в качестве аппроксиматоров Q-функций, что дает несколько преимуществ. Во-первых, прогнозирование значения Q для каждого действия теперь имеет явное понятие пространственной локальности по отношению к другим действиям, а также к входному наблюдения состояния . Во-вторых, FCN эффективны для вычислений по пикселям.

Отдельно необходимо рассмотреть вопрос с наградами в Q-обучении.

Мы назначаем (, ) = 1 в случае успешного захвата (вычисляется с помощью порогового значения расстояний между пальцами гриппера после попытки захвата) и (, ) = 0,5 за толчки, которые вносят обнаруживаемые изменения в среду (изменения обнаруживаются, если сумма различий между картами высот превышает некоторый порог τ , т.е. P((- ) > τ ). Важно, что внутреннее вознаграждение (, ) не учитывает в явном виде, обеспечивает ли толчок будущие хватания. Скорее, оно просто поощряет систему делать толчки, которые вызывают изменения. Синергия между толканием и хватанием усваивается в основном через обучение с подкрепление.

Ниже представлен код, двух сеток для толкания и хватания написанный на PyTorch.

class reactive\_net(nn.Module):

def \_\_init\_\_(self, use\_cuda): # , snapshot=None

super(reactive\_net, self).\_\_init\_\_()

self.use\_cuda = use\_cuda

# Initialize network trunks with DenseNet pre-trained on ImageNet

self.push\_color\_trunk = torchvision.models.densenet.densenet121(pretrained=True)

self.push\_depth\_trunk = torchvision.models.densenet.densenet121(pretrained=True)

self.grasp\_color\_trunk = torchvision.models.densenet.densenet121(pretrained=True)

self.grasp\_depth\_trunk = torchvision.models.densenet.densenet121(pretrained=True)

self.num\_rotations = 16

# Construct network branches for pushing and grasping

self.pushnet = nn.Sequential(OrderedDict([

('push-norm0', nn.BatchNorm2d(2048)),

('push-relu0', nn.ReLU(inplace=True)),

('push-conv0', nn.Conv2d(2048, 64, kernel\_size=1, stride=1, bias=False)),

('push-norm1', nn.BatchNorm2d(64)),

('push-relu1', nn.ReLU(inplace=True)),

('push-conv1', nn.Conv2d(64, 3, kernel\_size=1, stride=1, bias=False))

# ('push-upsample2', nn.Upsample(scale\_factor=4, mode='bilinear'))

]))

self.graspnet = nn.Sequential(OrderedDict([

('grasp-norm0', nn.BatchNorm2d(2048)),

('grasp-relu0', nn.ReLU(inplace=True)),

('grasp-conv0', nn.Conv2d(2048, 64, kernel\_size=1, stride=1, bias=False)),

('grasp-norm1', nn.BatchNorm2d(64)),

('grasp-relu1', nn.ReLU(inplace=True)),

('grasp-conv1', nn.Conv2d(64, 3, kernel\_size=1, stride=1, bias=False))

# ('grasp-upsample2', nn.Upsample(scale\_factor=4, mode='bilinear'))

]))

# Initialize network weights

for m in self.named\_modules():

if 'push-' in m[0] or 'grasp-' in m[0]:

if isinstance(m[1], nn.Conv2d):

nn.init.kaiming\_normal(m[1].weight.data)

elif isinstance(m[1], nn.BatchNorm2d):

m[1].weight.data.fill\_(1)

m[1].bias.data.zero\_()

# Initialize output variable (for backprop)

self.interm\_feat = []

self.output\_prob = []

def forward(self, input\_color\_data, input\_depth\_data, is\_volatile=False, specific\_rotation=-1):

if is\_volatile:

output\_prob = []

interm\_feat = []

# Apply rotations to images

for rotate\_idx in range(self.num\_rotations):

rotate\_theta = np.radians(rotate\_idx\*(360/self.num\_rotations))

# Compute sample grid for rotation BEFORE neural network

affine\_mat\_before = np.asarray([[np.cos(-rotate\_theta), np.sin(-rotate\_theta), 0],[-np.sin(-rotate\_theta), np.cos(-rotate\_theta), 0]])

affine\_mat\_before.shape = (2,3,1)

affine\_mat\_before = torch.from\_numpy(affine\_mat\_before).permute(2,0,1).float()

if self.use\_cuda:

flow\_grid\_before = F.affine\_grid(Variable(affine\_mat\_before, requires\_grad=False).cuda(), input\_color\_data.size())

else:

flow\_grid\_before = F.affine\_grid(Variable(affine\_mat\_before, requires\_grad=False), input\_color\_data.size())

# Rotate images clockwise

if self.use\_cuda:

rotate\_color = F.grid\_sample(Variable(input\_color\_data, volatile=True).cuda(), flow\_grid\_before, mode='nearest')

rotate\_depth = F.grid\_sample(Variable(input\_depth\_data, volatile=True).cuda(), flow\_grid\_before, mode='nearest')

else:

rotate\_color = F.grid\_sample(Variable(input\_color\_data, volatile=True), flow\_grid\_before, mode='nearest')

rotate\_depth = F.grid\_sample(Variable(input\_depth\_data, volatile=True), flow\_grid\_before, mode='nearest')

# Compute intermediate features

interm\_push\_color\_feat = self.push\_color\_trunk.features(rotate\_color)

interm\_push\_depth\_feat = self.push\_depth\_trunk.features(rotate\_depth)

interm\_push\_feat = torch.cat((interm\_push\_color\_feat, interm\_push\_depth\_feat), dim=1)

interm\_grasp\_color\_feat = self.grasp\_color\_trunk.features(rotate\_color)

interm\_grasp\_depth\_feat = self.grasp\_depth\_trunk.features(rotate\_depth)

interm\_grasp\_feat = torch.cat((interm\_grasp\_color\_feat, interm\_grasp\_depth\_feat), dim=1)

interm\_feat.append([interm\_push\_feat, interm\_grasp\_feat])

# Compute sample grid for rotation AFTER branches

affine\_mat\_after = np.asarray([[np.cos(rotate\_theta), np.sin(rotate\_theta), 0],[-np.sin(rotate\_theta), np.cos(rotate\_theta), 0]])

affine\_mat\_after.shape = (2,3,1)

affine\_mat\_after = torch.from\_numpy(affine\_mat\_after).permute(2,0,1).float()

if self.use\_cuda:

flow\_grid\_after = F.affine\_grid(Variable(affine\_mat\_after, requires\_grad=False).cuda(), interm\_push\_feat.data.size())

else:

flow\_grid\_after = F.affine\_grid(Variable(affine\_mat\_after, requires\_grad=False), interm\_push\_feat.data.size())

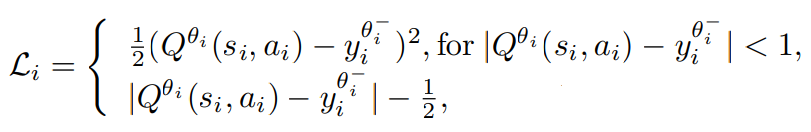
# Forward pass through branches, undo rotation on output predictions, upsample results

output\_prob.append([nn.Upsample(scale\_factor=16, mode='bilinear').forward(F.grid\_sample(self.pushnet(interm\_push\_feat), flow\_grid\_after, mode='nearest')),

nn.Upsample(scale\_factor=16, mode='bilinear').forward(F.grid\_sample(self.graspnet(interm\_grasp\_feat), flow\_grid\_after, mode='nearest'))])

return output\_prob, interm\_feat

Q-обучение проводится на каждой итерации i с использованием функции потерь Хьюбера:



Поскольку наша политика является жадно-детерминированной во время тестирования, есть вероятность застрять на выполнении однотипного действия, в то время как представление состояния (и, следовательно, оценки значений) остаются неизменными, поскольку никаких изменений в среде не происходит.

Эксперимент:

В рамках данного исследования были проведены серия экспериментов, для проверки предложенного подхода. Цели эксперимента показать в какой мере помогает толкание при дальнейшем захвате объекта.

В нашей установке для моделирования используется манипуляционный робот UR5 с гриппером RG2 в симуляционной среде CoppeliaSim (рис. 2). Для динамики используется Bullet Physics 2.83 и внутренний модуль инверсной кинематики V-REP для планирования движения робота. Объекты, используемые в симуляции включают в себя 9 различных трехмерных игрушечных блоков, формы и цвета которых выбираются случайным образом в ходе экспериментов. Большинство параметров динамики остаются по умолчанию, кроме трения, который был изменен для достижения синтетического поведение взаимодействия с объектом, приближенного к реальному миру. В симуляционную среду также в статичном положении добавлена 3D-камера из которой собираются данные о восприятии. RGB-D изображения с разрешением 640×480 визуализируются с помощью OpenGL с камеры, без каких-либо шумовых моделей глубины или цвета.

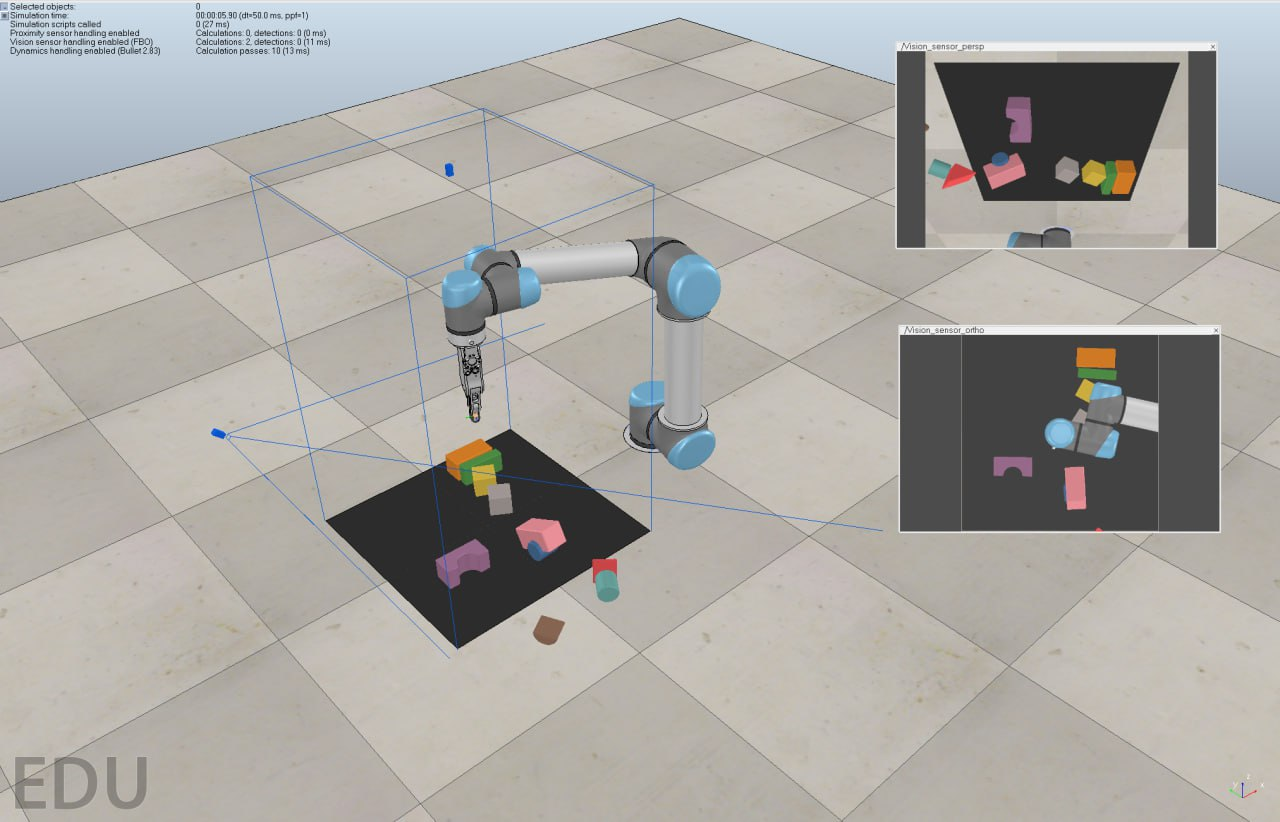


Рис. 2 Среда моделирования. Политики обучаются в случаях, где случайно расположены 10 объектов, затем тестируется в случаях с различной степени беспорядка.

Для достижения нашей цели модели были обучены на PyTorch с NVIDIA GeForce RTX 3070 на процессоре AMD Ryzen 5000 с тактовой частотой 3.7 ГГц. Для сравнения после обучения были построены графики, где пунктирами обозначен показатель уцпешного захвата после толкания и без него соответственно.

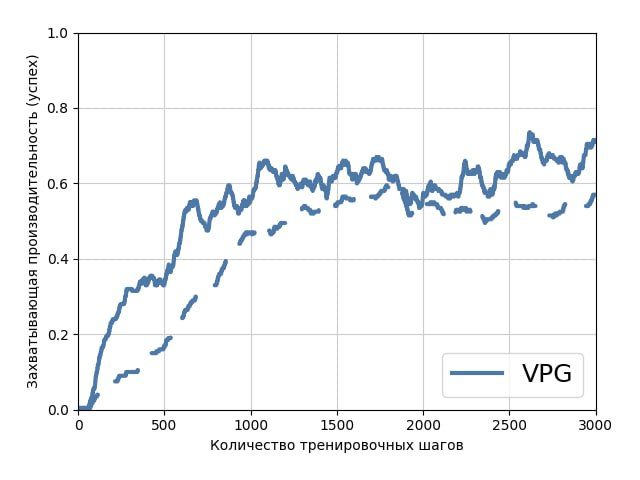


Рис. 2 Сплошные линии показывают процент успешного захвата, а пунктирные линии показывают процент успешных попыток после толкания а затем хватания.

Заключение:

В этой работе мы представляем тулбокс для обучения захвата и толкания предметов с помощью обучения с подкреплением. В данном исследовании показали, что синергия между планированием толкающих и хватающих действий может быть извлечена из опыта. Наш метод основан на пиксельной версии deep networks, которая сочетает в себе глубокое обучение с подкреплением и манипулирование, основанное на доступности. Результаты показывают, что наша система учится выполнять сложные последовательности толчков и захватов на реальном роботе за приемлемое время обучения. Однако его ограничения указывают на направления будущей работы. Во–первых, примитивы движения определяются с параметрами, указанными в обычной сетке (карта высот), что обеспечивает эффективность обучения с помощью глубоких сетей, но ограничивает выразительность - было бы интересно изучить другие параметризации, которые позволяют более выразительные движения (без чрезмерной сложности выборки), включая более динамичные толчки, параллельные, а не последовательные комбинации толчка и захвата, а также использования более разнообразных контактных поверхностей робота. Второе ограничение заключается в том, что мы тренируем нашу систему только с блоками и тестируем с ограниченным набором форм – более перспективно обучить на более крупных вариантах форм и дополнительно оценить возможности обобщения изученных политик. Вкратце, мы изучаем только синергию между толчком и захватом, которые являются всего лишь двумя примерами более широкого семейства примитивных манипулятивных действий, например, перекатывание, опрокидывание, сдавливание, укладка. Исследование пределов этого подхода к обучению с глубоким подкреплением в других многоэтапных взаимодействиях является важной темой для будущей работы.

Список используемой литературы:

1. M. R. Dogar and S. S. Srinivasa, “A planning framework for no prehensile manipulation under clutter and uncertainty,” Autonomous Robots, 2012. [Электронный ресурс] – Режим доступа: https://personalrobotics.cs.washington.edu/publications/dogar2012planning.pdf