МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра Вычислительной техники

ОТЧЁТ

по лабораторной работе № 2

«Управление манипулятором»

по дисциплине: «Основы теории управления автономными системами»

|  |  |
| --- | --- |
| Выполнили:  Студенты гр. АПИМ-25, АВТФ:  Клименко К. В.  Епифанцев Г. А.  Вариант № 2, 17 | Преподаватель:  доцент каф. ВТ,  к.т.н. Яковина И.Н. |

Новосибирск, 2025

# Описание проделанной работы

## Часть 1.1. Знакомство с манипулятором

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1. Параметры поворотных узлов | | | |
| Имя узла | Pos. min, град. | Pos. range, град. | Pos. max, град. |
| rjoint\_0 | -180° | 360° | +180°/0° |
| rjoint\_1 | -110° | 200° | +90° |
| rjoint\_2 | -50° | 280° | +230° |
| rjoint\_3 | -120° | 240° | +120° |



рисунок 1. Информация о границах положений узлов манипулятора

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 2. Соответствие внешнего вида манипулятора и значений угловых положений для поворотных узлов | | | |
|  | рис 2.1 | рис 2.2 | рис 2.3 |
| Имя узла | Угловое положение, град | Угловое положение, град | Угловое положение, град |
| rjoint\_0 | 0° | +45° | -180° |
| rjoint\_1 | +45° | -30° | -60° |
| rjoint\_2 | -45° | +180° | +40° |
| rjoint\_3 | +90° | -90° | +50° |

## Часть 1.2. Установка актуатора

Заключение о результатах проверки захватов на удержание груза:

1. BaxterGripperWithGUI — успешно поднял груз и обеспечил его устойчивое удержание;
2. FrankaGripper — при срабатывании применил чрезмерное усилие, в результате чего груз (кубик) был вытеснен из захвата ещё до подъёма и остался на поверхности, поднятие не выполнено;
3. RG2 — корректно захватил груз, обеспечил надёжное удержание и успешно поднял его без срывов;
4. ROBOTIQ85 — не смог удержать груз: объект выпал из захвата сразу при попытке подъёма, удержание не обеспечено;
5. YuMiGripper — захватил и поднял груз, однако удержание оказалось нестабильным: через ~1 секунду груз выскользнул из захвата и упал.

|  |
| --- |
| рисунок 3. Результат проверки захватов на удержание груза |

## Часть 1.4. Индивидуальные задания с инструментами

**Щипковый манипулятор:**

Координаты манипулятора:

x: +0.51898 y: +0.00001 z: +0.65161

Название актуатора: BaxterGripperWithGUI

Координаты актуатора относительно элемента ForceSensor:

x: 0.000 y: 0.000 z: +0.035

Координаты объекта:

x: +0.09134 y: +0.51801 z: +0.015

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.1 Перечень целевых и промежуточных точек при выполнении движения | | | |
| № | Положение целевой точки (x, y, z), м | Ориентация целевой точки (Alpha, Beta, Gamma), град. | Внешний вид манипулятора и объекта |
| 1 | [0.09004, 0.51062, 0.215] | [0, 0, 80] |  |
| 2 | [0.09004, 0.51062, 0.135] | [0, 0, 80] |  |
| 3 | [0.09134, 0.51801, 0.315] | [0, 0, 0] |  |
| 4 | [0.39955, 0.46654, 0.35] | [0, 0, 0] |  |
| 5 | [0.39955, 0.46654, 0.236] | [0, 0, 0] |  |
| 6 | [-0.66636, 0.08904, 0.45] | [0, 0, 0] |  |
| 7 | [-0.66751, 0.08534, 0.436] | [0, 0, 0] |  |

|  |
| --- |
| Листинг управляющей программы (моноширинный шрифт Consolas, 10 кегль)  def sysCall\_thread():    sphere\_id = sim.getObject('.')  force\_sensor\_id = sim.getObject('/IRB140/connection')  actuator\_id = sim.getObject('/IRB140/connection/BaxterGripperWithGUI')  cube\_id = sim.getObject("/DummyL/Cuboid")  cylinder\_1 = sim.getObject("/DummyL/Cylinder[0]")  cylinder\_2 = sim.getObject("/DummyL/Cylinder[1]")    sphere\_or = sim.getObjectOrientation(sphere\_id, -1)  cube\_pos = sim.getObjectPosition(cube\_id, -1)  cube\_or = sim.getObjectOrientation(cube\_id, -1)    print(f"Position: {[round(x, 5) for x in sim.getObjectPosition(sphere\_id, -1)]}")  print(f"Position: {[round(x, 5) for x in sim.getObjectPosition(actuator\_id, force\_sensor\_id)]}")  print(f"Position: {[round(x, 5) for x in sim.getObjectPosition(cube\_id, -1)]}")    p1 = [-0.0075, 0, 0.2]  p2 = [-0.0075, 0, 0.12]  p3 = [cube\_pos[0], cube\_pos[1], cube\_pos[2] + 0.3]  p4 = [0, 0.0075, 0.3]  p5 = [0, 0.0075, 0.186]  p6 = [0.005, 0.005, 0.3]  p7 = [0.005, 0.005, 0.286]    interpolate\_smoothly('move', sphere\_id, p1, 7, cube\_id)  interpolate\_smoothly('turn', sphere\_id, cube\_or, 2, None)  interpolate\_smoothly('move', sphere\_id, p2, 3, cube\_id)  sim.setInt32Signal ("BaxterGripperWithGUI\_\_44\_\_\_close", 1)  sim.wait(3)  interpolate\_smoothly('move', sphere\_id, p3, 3, None)  interpolate\_smoothly('turn', sphere\_id, sphere\_or, 2, None)  interpolate\_smoothly('move', sphere\_id, p4, 3, cylinder\_1)  interpolate\_smoothly('move', sphere\_id, p5, 3, cylinder\_1)  sim.setInt32Signal ("BaxterGripperWithGUI\_\_44\_\_\_close", 0)  sim.wait(3)  sim.setInt32Signal ("BaxterGripperWithGUI\_\_44\_\_\_close", 1)  sim.wait(3)  interpolate\_smoothly('move', sphere\_id, p6, 8, cylinder\_2)  interpolate\_smoothly('move', sphere\_id, p7, 1, cylinder\_2)  sim.setInt32Signal ("BaxterGripperWithGUI\_\_44\_\_\_close", 0)      def to\_deg(radians:float) -> float:  return math.degrees(radians)    def print\_position\_and\_orientation(obj\_id:int):  position = [round(x, 5) for x in sim.getObjectPosition(obj\_id, -1)]  print(f"Position: {position}")  orientation = [round(to\_deg(rad)) for rad in sim.getObjectOrientation(obj\_id, -1)]  print(f"Orientation: {orientation}")    def interpolate\_smoothly(action, obj\_id, target, duration, parent\_id):  if parent\_id == None:  parent\_id = -1  steps = 50  current = []    if action == 'move':  current = sim.getObjectPosition(obj\_id, parent\_id)  elif action == 'turn':  current = sim.getObjectOrientation(obj\_id, parent\_id)  else:  raise ValueError('No such action')    dx = (target[0] - current[0]) / steps  dy = (target[1] - current[1]) / steps  dz = (target[2] - current[2]) / steps  step\_time = duration / steps  for i in range(1, steps + 1):  new = [  current[0] + dx \* i,  current[1] + dy \* i,  current[2] + dz \* i  ]  if action == 'move':  sim.setObjectPosition(obj\_id, parent\_id, new)  elif action == 'turn':  sim.setObjectOrientation(obj\_id, parent\_id, new)  sim.wait(step\_time)    print\_position\_and\_orientation(obj\_id) |

**Вакуумный манипулятор:**

Координаты манипулятора:

x: +0.51898 y: +0.00001 z: +0.5621

Название актуатора: BaxterVacuumCupWhithGUI

Координаты объекта:

x: +0.67844 y: -0.42394 z: +0.500

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 4 Перечень целевых и промежуточных точек при выполнении движения | | | |
| № | Положение целевой точки (x, y, z), м | Ориентация целевой точки (Alpha, Beta, Gamma), град. | Внешний вид манипулятора и объекта |
| 1 | [0.51898, 1e-05, 0.65161] | [-90, -58, -90] |  |
| 2 | [0.55123, 0.34445, 0.49] | [-90, -58, -90] |  |
| 3 | [-0.4982, 0.03048, 0.5234] | [3, 45, -5] |  |
| 4 | [0.01832, 0.34952, 0.647] | [90, 3, -90] |  |

Для решения задачи были добавлены функции расчета центра составного объекта и перевод относительных координат в мировые с учетом вектора смещения(для позиционирования ровно по середине коробки).

|  |
| --- |
| Листинг управляющей программы  def sysCall\_thread():  cube = sim.getObject("/Cuboid")  root = sim.getObject('/BoxSrc')  root1 = sim.getObject('/BoxDst[0]')  root2 = sim.getObject('/BoxDst[1]')  sphere = sim.getObject('/manipulationSphere')    zComponent = ('z', +1, 0.122)    interpolate\_smoothly('turn', sphere, [0, -math.pi / 2.0,0], 10, root)    p = get\_subtree\_point\_with\_offset(root,  multi\_axes=[('x', +1, -0.15), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p, 10, -1)    sim.setIntegerSignal('BaxterVacuumCupWhithGUI\_\_57\_\_\_active', 1)    p = get\_subtree\_point\_with\_offset(root,  multi\_axes=[('x', +1, -0.3), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p, 10, -1)  interpolate\_smoothly('turn', sphere, [0, math.pi / 2.0, 0], 10, root1)    p = get\_subtree\_point\_with\_offset(root,  multi\_axes=[('x', +1, -0.3), ('y', -1, 0.5), zComponent])  interpolate\_smoothly('move', sphere, p, 10, -1)    p2 = get\_subtree\_point\_with\_offset(root1,  multi\_axes=[('x', +1, 0.3), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p2, 10, -1)    p2 = get\_subtree\_point\_with\_offset(root1,  multi\_axes=[('x', +1, 0.15), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p2, 10, -1)  #sim.setObjectPosition(sphere, sim.handle\_world, p2)    p2 = get\_subtree\_point\_with\_offset(root1,  multi\_axes=[('x', +1, 0.3), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p2, 10, -1)    interpolate\_smoothly('turn', sphere, [0, math.pi / 2.0,0], 2, root2)    p3 = get\_subtree\_point\_with\_offset(root2,  multi\_axes=[('x', +1, 0.3), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p3, 10, -1)  p3 = get\_subtree\_point\_with\_offset(root2,  multi\_axes=[('x', +1, 0.15), ('y', +1, 0.00), zComponent])  interpolate\_smoothly('move', sphere, p3, 10, -1)  sim.setIntegerSignal('BaxterVacuumCupWhithGUI\_\_57\_\_\_active', 0)    p3 = get\_subtree\_point\_with\_offset(root2,  multi\_axes=[('x', +1, 0.30), ('y', +1, 0.00), zComponent]  )  interpolate\_smoothly('move', sphere, p3, 10, -1)    def to\_deg(radians:float) -> float:  return math.degrees(radians)    def print\_position\_and\_orientation(obj\_id:int):  position = [round(x, 5) for x in sim.getObjectPosition(obj\_id, -1)]  print(f"Position: {position}")  orientation = [round(to\_deg(rad)) for rad in sim.getObjectOrientation(obj\_id, -1)]  print(f"Orientation: {orientation}")    def asVec3(x):  while isinstance(x, (list, tuple)) and len(x) == 1 and isinstance(x[0], (list, tuple)):  x = x[0]  if not isinstance(x, (list, tuple)) or len(x) < 3:  raise ValueError(f"Expected vec3, got: {x!r}")  return [float(x[0]), float(x[1]), float(x[2])]  def shapeWorldCorners(shape\_h):  try:  sx, sy, sz = asVec3(sim.getShapeBB(shape\_h))  except Exception:  return []  hx, hy, hz = 0.5\*sx, 0.5\*sy, 0.5\*sz  corners\_local = [  [-hx,-hy,-hz], [ hx,-hy,-hz], [-hx, hy,-hz], [ hx, hy,-hz],  [-hx,-hy, hz], [ hx,-hy, hz], [-hx, hy, hz], [ hx, hy, hz],  ]  pose\_w = [float(v) for v in sim.getObjectPose(shape\_h, sim.handle\_world)]  out = []  for c in corners\_local:  p = sim.multiplyPoses(pose\_w, [c[0], c[1], c[2], 0.0, 0.0, 0.0, 1.0])  out.append([p[0], p[1], p[2]])  return out  def get\_subtree\_local\_aabb(root\_h):  root\_pose\_w = [float(v) for v in sim.getObjectPose(root\_h, sim.handle\_world)]  inv\_root\_pose = sim.invertPose(root\_pose\_w)  xs, ys, zs = [], [], []  shapes = sim.getObjectsInTree(root\_h, sim.object\_shape\_type, 0) or []  for sh in shapes:  for xw, yw, zw in shapeWorldCorners(sh):  p\_loc = sim.multiplyPoses(inv\_root\_pose, [xw, yw, zw, 0.0, 0.0, 0.0, 1.0])  xs.append(p\_loc[0])  ys.append(p\_loc[1])  zs.append(p\_loc[2])  if not xs:  return [0.0,0.0,0.0], [0.0,0.0,0.0]  return [min(xs),min(ys),min(zs)], [max(xs),max(ys),max(zs)]  def get\_subtree\_point\_with\_offset(root\_h,  gap=0.0, axis='x', direction=+1,  multi\_axes=None,  center\_other\_axes=True,  extra\_offset\_local=None, extra\_offset\_world=None):  idx = {'x':0, 'y':1, 'z':2}  mnL, mxL = get\_subtree\_local\_aabb(root\_h)  cL = [0.5\*(mnL[i]+mxL[i]) for i in range(3)]  pL = [cL[0], cL[1], cL[2]] if center\_other\_axes else [0.0, 0.0, 0.0]  if multi\_axes is None:  ax = axis.lower()  i = idx[ax]  pL[i] = (mxL[i] if direction >= 0 else mnL[i]) + direction \* float(gap)  else:  for ax, dir\_i, gap\_i in multi\_axes:  i = idx[ax.lower()]  pL[i] = (mxL[i] if dir\_i >= 0 else mnL[i]) + dir\_i \* float(gap\_i)  if extra\_offset\_local is not None:  dx, dy, dz = asVec3(extra\_offset\_local)  pL = [pL[0]+dx, pL[1]+dy, pL[2]+dz]  root\_pose\_w = [float(v) for v in sim.getObjectPose(root\_h, sim.handle\_world)]  pW\_pose = sim.multiplyPoses(root\_pose\_w, [pL[0], pL[1], pL[2], 0.0, 0.0, 0.0, 1.0])  pW = [pW\_pose[0], pW\_pose[1], pW\_pose[2]]  if extra\_offset\_world is not None:  DX, DY, DZ = asVec3(extra\_offset\_world)  pW = [pW[0]+DX, pW[1]+DY, pW[2]+DZ]  return pW  def interpolate\_smoothly(action, obj\_id, target, duration, parent\_id):  if parent\_id == None:  parent\_id = -1  steps = 50  current = []    if action == 'move':  current = sim.getObjectPosition(obj\_id, parent\_id)  elif action == 'turn':  current = sim.getObjectOrientation(obj\_id, parent\_id)  else:  raise ValueError('No such action')    dx = (target[0] - current[0]) / steps  dy = (target[1] - current[1]) / steps  dz = (target[2] - current[2]) / steps  step\_time = duration / steps  for i in range(1, steps + 1):  new = [  current[0] + dx \* i,  current[1] + dy \* i,  current[2] + dz \* i  ]  if action == 'move':  sim.setObjectPosition(obj\_id, parent\_id, new)  elif action == 'turn':  sim.setObjectOrientation(obj\_id, parent\_id, new)  sim.wait(step\_time)    print\_position\_and\_orientation(obj\_id) |

**Покраска объекта:**

Координаты манипулятора:

x: +0.51898 y: +0.00001 z: +0.65161

Название актуатора: PaintGun

Координаты объекта:

x: +0.750 y: +0.025 z: +0.075

|  |
| --- |
| рисунок 4 Сцена покраски |

В ходе выполнения задачи столкнулись с проблемой недостаточной длинны манипулятора. Для решения проблемы вдохновились виниловым проигрывателем. Добавили вращательный стол для монеты, покраска осуществлялась от края к центру

|  |
| --- |
| рисунок 5 Результат прокраски |

|  |
| --- |
| Листинг управляющей программы  def sysCall\_thread():  paint\_gun\_id = sim.getObject('.')  target\_surface\_id = sim.getObject("/RotaryTable/RotationTable/Coin")  rotation\_joint = sim.getObject("/RotaryTable/RotationTable")    surface\_center = sim.getObjectPosition(target\_surface\_id, -1)  surface\_radius = 0.4  paint\_height = 0.45    work\_pos = [surface\_center[0] - surface\_radius, surface\_center[1], surface\_center[2] + paint\_height]  interpolate\_smoothly('move', paint\_gun\_id, work\_pos, 3)    sim.wait(0.5)    paint\_vinyl(paint\_gun\_id, rotation\_joint, surface\_center, surface\_radius, paint\_height)    sim.setInt32Signal("PaintGun\_active", 0)  sim.wait(0.5)    def paint\_vinyl(paint\_gun\_id, rotation\_joint, center, radius, height):  rotation\_speed = 1.5  paint\_duration = 17.0  steps = 400    sim.setJointTargetVelocity(rotation\_joint, rotation\_speed)    for i in range(steps + 1):  t = i / steps  current\_radius = -radius \* (1 - t)    x = center[0] + current\_radius  y = center[1]  z = center[2] + height    sim.setObjectPosition(paint\_gun\_id, -1, [x, y, z])  sim.wait(paint\_duration / steps)    sim.setJointTargetVelocity(rotation\_joint, 0)    def to\_rad(degrees: float) -> float:  return math.radians(degrees)    def interpolate\_smoothly(action, obj\_id, target, duration):  steps = 50  current = []    if action == 'move':  current = sim.getObjectPosition(obj\_id, -1)  elif action == 'turn':  current = sim.getObjectOrientation(obj\_id, -1)  else:  raise ValueError('No such action')    dx = (target[0] - current[0]) / steps  dy = (target[1] - current[1]) / steps  dz = (target[2] - current[2]) / steps  step\_time = duration / steps    for i in range(1, steps + 1):  new = [  current[0] + dx \* i,  current[1] + dy \* i,  current[2] + dz \* i  ]  if action == 'move':  sim.setObjectPosition(obj\_id, -1, new)  elif action == 'turn':  sim.setObjectOrientation(obj\_id, -1, new)  sim.wait(step\_time) |

# Выводы

В ходе лабораторной работы для выполнения задания был выбран захват BaxterGripperWithGUI, поскольку он продемонстрировал наилучшую надёжность: успешно захватил кубик, поднял его и удерживал стабильно на всём протяжении перемещения. Остальные захваты показали недостатки: FrankaGripper вытолкнул груз из-за избыточного усилия, ROBOTIQ85 не удержал объект при подъёме, YuMiGripper выронил кубик через секунду, а RG2, несмотря на корректную работу, в данной сцене не применялся.

Индивидуальное задание реализовано с помощью Python-скрипта в CoppeliaSim. Управление манипулятором осуществляется через целевую сферу — объект manipulationSphere, с помощью которой решалась задача обратной кинематики. Вместо ручного задания углов джойнтов скрипт последовательно устанавливает желаемые позицию и ориентацию этой сферы. Чтобы движения были плавными, все переходы выполняются через линейную интерполяцию: траектория разбивается на 50 шагов с небольшими паузами между ними. Захват управляется отправкой целочисленного сигнала (BaxterGripperWithGUI\_\_44\_\_\_close), а фиксированные задержки обеспечивают завершение механических действий до следующего этапа.

Такой подход — управление через целевую сферу — позволил точно и стабильно выполнить все этапы задания: подъезд к грузу с учётом его ориентации, захват, перемещение в зону разгрузки и аккуратное отпускание.

Для решения задачи позиционирования вакуумного инструмента были добавлены: функция вычисления геометрического центра составного объекта (с учётом дочерних мешей и их трансформаций) и преобразование относительных координат в мировые с учётом вектора смещения. Это обеспечило точное наведение по середине коробки и надёжный контакт присоски с плоскостью объекта.

Для выполнения покрасочных операций столкнулись с ограничением рабочей зоны.Длины манипулятора не хватало для равномерного покрытия поверхности. Вдохновившись конструкцией винилового проигрывателя, был добавлен вращательный стол для монеты. Процесс покраски реализован по спирали — от края к центру — при постоянной подаче и фиксированном зазоре сопла. Такой приём позволил компенсировать недостаток вылета и обеспечить равномерность.