Appendix

Script 1: Lua script file: ugv-04–vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 04 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- # @todo Fix VelodyneScan publishing -- The 'packets' field is not assigned ▼12
    12▲ properly despite proper assignment when VelodynePacket is used directly, ▼12
    12▲ probably needs conversion from Lua floats to uint8
-- # @done Implement switch for publishing pointcloud2 and raw messages
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼24
    24▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 ▼42
    42▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 = ▼43
    43▲ pointCloud2 messages;
  -- Publisher switch
  publishSwitch = 1 -- 0 = off; +1 = PointCloud2 (not implemented, on by default); ▼45
    45▲ +2 = raw messages scan (not working properly because of incorrect packets ▼45
    45▲ assignment); +4 = raw messages packet;
```

```
-- Counters
counter = 0

-- Handles
-- Frames
worldHandle = simGetObjectHandle('frame00')
robot01Handle = simGetObjectHandle('robot01')
-- vlp16Handle = simGetObjectHandle('vlp16')
vlp16Handle = simGetObjectHandle('vpl16')

-- Tranform (TF)
tfLinks = {}
-- for i,link in ipairs {'vlp16'} do
for i, link in ipairs {'vpl16'} do
  tfLinks[link] = simGetObjectHandle(link)
end

-- Base TF
tf = {
  header = {
    stamp = 0,
    frame_id = 'frame00'
  },
  child_frame_id = '',
  transform = {
    -- ROS has definition x = front y = side z = up
    translation = {x = 0, y = 0, z = 0}, -- V-rep
    rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
  }
}


-- VLP16
pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
visionSensorHandles = {}
for i = 1, 4, 1 do
  visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
end

-- Setting VLP16 parameters
frequency = 10 -- Default model 5 Hz
-- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only  ▼87
  87▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼87
  87▲  (8)=displayed points are emissive
options = 1 -- No display
pointSize = 2
coloringCloseAndFarDistance = {1, 4}
displayScaling = 0.999 -- so that points do not appear to disappear in objects

-- Reading VLP16 sensor data
vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles,  ▼94
  94▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼94
  94▲  pointCloudHandle)
```

```
95
    -- Init publishers and subscribers
    -- Init publishers
-- #S#
100 -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1, ▼100
      100▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
    -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
    -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼102
      102▲ PointCloud2')
    -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼103
      103▲ VelodyneScan')
    -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼104
      104▲ VelodynePacket')
105 -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼105
      105▲ VelodyneScan')
-- #E#
    if bitand(publishSwitch, 1) == 1 then
      vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2 ▼108
      108▲ ')
      simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
110 end
--  --[[ #DS#
    if bitand(publishSwitch, 2) == 2 then
      vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼113
      113▲ VelodyneScan')
      -- simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
115 elseif bitand(publishSwitch, 4) == 4 then
      vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼116
      116▲ VelodynePacket')
      simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
    end
-- #DE# --]]
120 end

-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
    -- Retrieve sensor data
125 data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼125
      125▲ simGetSimulationTimeStep())
    -- Retrieve sensor transformation matrix
    vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

    -- Debug options
130 -- print('----------------')
    -- print(bitand(debugOutput, 1))
    -- print(bitand(debugOutput, 2))
    -- print(bitand(debugOutput, 4))
    -- print('----------------')
135
    -- Display the detected points
    if displayPointCloud == 1 then
      if pointCloud then
```

```
    simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
  else
    pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
  end
end

-- Construct the point cloud from raw data: x, y and z coordinates for every  ▼145
 145▲ point
-- Reference: From the original VPL_16 model script
local pointCloudData = {}
for i = 0, #data / 3 - 1, 1 do
  -- Temporary assignment to a vector d
  d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
  -- Transformation to sensor frame
  -- No transformation results in point cloud rotated by 90 deg around z
  d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
  data[3 * i + 1] = d[1]
  data[3 * i + 2] = d[2]
  data[3 * i + 3] = d[3]
  table.insert(pointCloudData, d)
  -- print(d)
end

-- Display the detected points
if displayPointCloud == 1 then
  simInsertPointsIntoPointCloud(pointCloud, 0, data)
end

if bitand(publishSwitch, 2) == 2 or bitand(publishSwitch, 4) == 4 then
-- --[[ #DS# -- Comment this line for raw message handling, currently only a  ▼167
 167▲ dummy message publishing implemented
  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402,  ▼176
 176▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
```

```
    -- for i = 1, 402, 1 do
190     for i = 1, 1, 1 do -- works forcing one entry in table
          table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
  -- #S#
        -- table.insert(tpcl, {1, 1, 1})
        -- table.insert(tpcl, {1.0, 1.0, 1.0})
195     -- table.insert(tpcl, {2, 2, 2})
  -- #E#
      end

      local pointCloudRawData = {}
200   -- CRASH with multiple objects in data
      if bitand(publishSwitch, 2) == 2 then
        pointCloudRawData['header'] = {seq = 0, stamp = simGetSimulationTimeStep() ▼202
  202▲ , frame_id = "frame00"}
        -- Apparently neither indirect nor direct assignment of packets object  ▼203
  203▲ works
        pointCloudRawData['packets']  = {stamp = simGetSimulationTimeStep(), data  ▼204
  204▲ = simPackFloats(tpcl)}
205     -- Lua runtime error: [string "[embScript_51284011.lua] SCRIPT  ▼205
  205▲ velodyneVPL_16"]:253: read__velodyne_msgs__VelodyneScan: field packets:  ▼205
  205▲ expected array (simExtRosInterface_publish @ 'RosInterface' plugin)
  -- #S#
        -- pointCloudRawData['packets']  = {stamp = simGetSimulationTimeStep(),  ▼207
  207▲ data = simPackUInts(tpcl)}
        -- local pointCloudRawDataPackets = {}
        -- pointCloudRawDataPackets = {stamp = simGetSimulationTimeStep(), data =  ▼209
  209▲ simPackFloats(tpcl)}
210     -- pointCloudRawData['packets'] = pointCloudRawDataPackets
        -- Equivalent to above
        -- pointCloudRawData = { packets  = {
          -- stamp = simGetSimulationTimeStep(), data = simPackFloats(tpcl)
          -- }
215     -- }
  -- #E#
      elseif bitand(publishSwitch, 4) == 4 then
        pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼218
  218▲ simPackFloats(tpcl)}
      end
220 -- #S#
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼221
  221▲ simPackUInts(tpcl)}
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼222
  222▲ simPackUInts(data)} -- CRASH
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼223
  223▲ simPackUInts(tdata)} -- CRASH
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼224
  224▲ simPackUInts(pointCloudData)} -- CRASH
225 -- #E#

      if bitand(debugOutput, 2) == 2 then
        print('-*- VLP16 Raw START')
        print('tpcl')
```

```
        print(#tpcl)
        print('tdata')
        print(#tdata)
    -- #S#
        -- print('h')
        -- print(type(h))
        -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua]  ▼236
    236▲ SCRIPT velodyneVPL...":140: attempt to get length of global 'h' (a  ▼236
    236▲ number value)
        -- print(type(tpcl))
        -- print(type(simPackUInts(tpcl)))
        -- print(type(simPackFloats(tpcl)))
        -- print(#simPackUInts(tpcl))
        -- print(type(tdata))
        -- print(tdata)
        -- for i, j in pairs (tdata) do
          -- print(i, j)
        -- end
        -- for i, j in pairs (data) do
          -- print(i, j)
        -- end
        -- print(#simPackUInts(tdata)) -- nil
        -- print('raw')
        -- print(type(pointCloudRawData.data))
        -- print(#pointCloudRawData.data)
        -- print('-*- VLP16 Raw END')
    -- #E#
      end
      simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
      rmcounter = rmcounter + 1
    end
    if bitand(debugOutput, 2) == 2 then
      print('rmcounter')
      print(rmcounter)
      print('-*- VLP16 Raw END')
    end
    -- #S#
    -- #E#
  end
  -- #DE# --]]

  if bitand(publishSwitch, 1) == 1 then
    -- Publish point cloud

    -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
    -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/  ▼273
    273▲ Software/V-REP/robot.lua
    -- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
    local pointCloudData = {}
    --[[
    -- The frame of VLP16 shifts all the points to that frame
    -- If frame_id set to sensor frame in current configuration results in point  ▼278
    278▲ cloud shifted to the sensor position
```

```
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼279
    279▲ frame_id = "vlp16"}
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼280
    280▲ frame_id = "vpl16"}
    --]]
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼282
    282▲ frame_id = "frame00"}
    -- Describes the channels and their layout in the binary data blob
    local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
    local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
    local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
    pointCloudData['fields'] = {field_x, field_y, field_z}
    pointCloudData['is_bigendian'] = false
    pointCloudData['is_dense'] = true
    pointCloudData['data'] = simPackFloats(data) --  Works
    -- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the  ▼291
    291▲ data field being of type uint8
    -- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼292
    292▲ and it differs in number of received points
    --[[
    -- Theoretically this should work, but the bytes offset still drop the  ▼294
    294▲ messages
    pointCloudData['height'] = 16 -- Ordered, 16 scan lines
    pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData ▼296
    296▲ .height)
    pointCloudData['point_step'] = math.floor(#pointCloudData.data /  ▼297
    297▲ pointCloudData.width / pointCloudData.height)
    --]]
    -- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
    pointCloudData['height'] = 1 -- Unordered
    -- Effectively number of cloud points, each having 3 coordinates
    pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
    -- Length of a point in bytes
    pointCloudData['point_step'] = math.floor(#pointCloudData.data /  ▼304
    304▲ pointCloudData.width)
    -- Length of a row in bytes
    pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼306
    306▲ point_step']

    if bitand(debugOutput, 4) == 1 then
      print('-- VLP16 PointCloud2 START')
      print(#pointCloudData.data)
      print(pointCloudData.width * pointCloudData.point_step)
      print(pointCloudData.point_step)
      print(#pointCloudData.data - math.floor(#pointCloudData.data /  ▼313
    313▲ pointCloudData.width) * pointCloudData.width)
      print(#pointCloudData.data - pointCloudData.width * pointCloudData. ▼314
    314▲ point_step)
      print(pointCloudData.width)
  -- #S#
    -- print(type(pointCloudData))
    -- print(type(tdata))
  -- #E#
```

```
320       print('-- VLP16 PointCloud2 END')
        end

        simExtRosInterface_publish(vlp16Pub, pointCloudData)
      end
325
      -- --[[
      -- Send transforms
      tf.header.stamp = simGetSimulationTimeStep()
      for link, handle in pairs(tfLinks) do
330     -- Get parent
        -- if (link == 'vlp16') then
        if (link == 'vpl16') then
          tf.header.frame_id = 'frame00' -- Absolute frame reference
          -- tf.header.frame_id = 'robot01' -- Relative to robot frame reference
335       -- Get pose relative to worldframe
          position = simGetObjectPosition(handle, worldHandle) -- Absolute frame ▼336
        336▲ reference
          orientation = simGetObjectQuaternion(handle, worldHandle) -- Absolute frame ▼337
        337▲ reference
          -- position = simGetObjectPosition(handle, robot01Handle) -- Relative to ▼338
        338▲ robot frame reference
          -- orientation = simGetObjectQuaternion(handle, robot01Handle) -- Relative ▼339
        339▲ to robot frame reference
340     else
          -- tf.header.frame_id = 'vlp16'
          tf.header.frame_id = 'vpl16'
          -- Get pose relative to sensor
          position = simGetObjectPosition(handle, vlp16Handle)
345       orientation = simGetObjectQuaternion(handle, vlp16Handle)
        end

        -- Update TF
        tf.child_frame_id = link
350     tf.transform.translation.x = position[1]
        tf.transform.translation.y = position[2]
        tf.transform.translation.z = position[3]
        tf.transform.rotation.x = orientation[1]
        tf.transform.rotation.y = orientation[2]
355     tf.transform.rotation.z = orientation[3]
        tf.transform.rotation.w = orientation[4]
        simExtRosInterface_sendTransform(tf)
      end
      --]]
360
      -- Update counter
      counter = counter + 1

      if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
365     simAddStatusbarMessage('VLP16 counter: ' .. counter)
        print('--- VLP16 START')
        print('pcl data')
        print(#pointCloudData)
```

```
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 2: Lua script file: ugv-01–robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Robot01 -- Prioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message format. UGV is immobile. The raw message format is implemented but not used, since data formatting is not working. An additional object was added to test the filter.
-- ###


-- ## Functions
-- # Callback for desired velocity
function controlCallback(msg)
  -- Reads desired linear and angular velocities
  if(table.getn(msg.name) == table.getn(msg.position)) then
    for i, joint in ipairs(msg.name) do
      simSetJointTargetVelocity(jointHandles[joint], msg.position[i])
    end
  end
end

-- # Callback for current path computation
function pathCallback(msg)
  pathCurrent = pathHandles['Path' .. msg.data]
  pathCurrent_pose = simGetObjectPosition(pathCurrent, -1)
end

-- # Callback for current pose to path projection
function poseCallback(pose)
dist = simGetClosestPositionOnPath(pathCurrent, {pose.x-pathCurrent_pose[1], pose.y -pathCurrent_pose[2], 0}) + delta
  --simAddStatusbarMessage('delta: ' .. delta)
  if dist > 1 or dist < 0 then
    delta = -delta
  end
  xyz = simGetPositionOnPath (pathCurrent, dist)
  rpy = simGetOrientationOnPath(pathCurrent, dist)
  simExtRosInterface_publish(posePub, {x = xyz[1], y = xyz[2], theta = rpy[3]})
  simSetObjectPosition(proj, -1, xyz)
  --simAddStatusbarMessage('on path @ (' .. xyz[1] .. ', ' .. xyz[2] .. ', ' .. rpy [3] .. ')')

  -- Put the robot there
  simSetObjectPosition(req, -1, {pose.x, pose.y, 4})
end
```

```
-- # Function for TF publications
function getTransformStamped(objHandle, name, relTo, relToName)
  t = simGetSystemTime()
  p = simGetObjectPosition(objHandle, relTo)
  o = simGetObjectQuaternion(objHandle, relTo)
  return {
    header={
      stamp = t,
      frame_id = relToName
    },
    child_frame_id = name,
    transform = {
      -- ROS has definition x=front y=side z=up
      translation = {x = p[1], y = p[2], z = p[3]}, --V-rep
      rotation = {x = o[1], y = o[2], z = o[3], w = o[4]} --V-rep
    }
  }
end

-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  counter = 0
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  poseprefix = 'pose/'
  -- poseprefix = ''
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼83
    83▲ dylib):
  while moduleName do
    moduleName,moduleVersion = simGetModuleName(index)
-- TODO which one is it? Ros or RosInterface ?
--    if (moduleName == 'RosInterface') then
    if (moduleName == 'Ros') then
      pluginNotFound = false
    end
    index = index + 1
  end

  -- Conversion deg / rad
  d2r = math.pi / 180.
  r2d = 180. / math.pi

  -- Paths
  delta = 0.01
```

```
  delta_sign = 1

  if (pluginNotFound) then

    -- Display an error message if the plugin was not found:
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼105
    105▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼105
    105▲ {0.5,0,0,1,1,1})

  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    camera01Handle = simGetObjectHandle('camera01')
    object01Handle = simGetObjectHandle('object01') -- Cylinder
    object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- Frames
    worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers

    -- Init publishers
    camera01Pub = simExtROS_enablePublisher('image01', 1, ▼122
    122▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    -- Cylinder
    object01Pub = simExtROS_enablePublisher(poseprefix .. 'object01', 1, ▼124
    124▲ simros_strmcmd_get_object_pose, object01Handle, robot01Handle, '')
    -- Cuboid
    object02Pub = simExtROS_enablePublisher(poseprefix .. 'object02', 1, ▼126
    126▲ simros_strmcmd_get_object_pose, object02Handle, camera01Handle, '')
    robot01jointsPub = simExtROS_enablePublisher('robot01joints', 1, ▼127
    127▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')
    -- PCLPub = simExtROS_enablePublisher('pcl', 1, simros_strmcmd_get_joint_state ▼128
    128▲ , sim_handle_all, -1, '')

    -- Init subscribers
    robot01StatesSub = simExtROS_enableSubscriber('robot01jointscontrol', 1, ▼131
    131▲ simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

  -- PASTE

--[[
    req = simGetObjectHandle('pose_request')
    proj = simGetObjectHandle('pose_response')
    pathHandles = {}
    for index=1,1 do
      pathHandles['Path' .. index] = simGetObjectHandle('Path' .. index)
      simAddStatusbarMessage('Found path #' .. index)
    end
--]]
```

```
145     -- Some handles for TF
        tfLinks = {}
        for i,link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼147
       147▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link'} do
          tfLinks[link] = simGetObjectHandle(link)
        end
150     -- Base TF
        tf = {
          header = {
            stamp = 0,
            frame_id ='frame00'
155       },
          child_frame_id = '',
          transform = {
            -- ROS has definition x=front y=side z=up
            translation = {x = 0, y = 0, z = 0},-- V-rep
160         rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
          }
        }

    --[[
165     -- Publish joint states
        -- simExtROS_enablePublisher("/joint_states", 4102,  ▼166
       166▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, "")

        -- Get joint handles
        jointHandles={}
170     for j,lr in ipairs{'L','R'} do
          for i,fr in ipairs{'F','R'} do
            -- Steering
            joint = 'steering' .. fr .. lr
            jointHandles[joint] = simGetObjectHandle(joint)
175         simSetJointTargetVelocity(jointHandles[joint],0)
          end
          -- Motor for F wheels
          joint = 'motorF' .. lr
          jointHandles[joint] = simGetObjectHandle(joint)
180       simSetJointTargetVelocity(jointHandles[joint],0)
        end

        -- Joint subscriber
        velocitySub = simExtRosInterface_subscribe('/joint_setpoint', 'sensor_msgs/ ▼184
       184▲ JointState', 'controlCallback')
185
        -- Advertise  pseudo-service = subscribe and publish topic
        pathCurrent = pathHandles['Path1']
        pathCurrent_pose = simGetObjectPosition(pathCurrent, -1)
        pathSub = simExtRosInterface_subscribe('path_nb', 'std_msgs/Int32', ' ▼189
       189▲ pathCallback')
190     poseSub = simExtRosInterface_subscribe('pose_request', 'geometry_msgs/Pose2D', ▼190
       190▲  'poseCallback')
        posePub = simExtRosInterface_advertise('pose_response', 'geometry_msgs/Pose2D' ▼191
       191▲ )
```

```
--]]

    -- Publish robot twist
    twistPub = simExtRosInterface_advertise('twist', 'geometry_msgs/Twist')

    -- Simulation time
    clockPub = simExtRosInterface_advertise('/clock', 'rosgraph_msgs/Clock')

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Update counter
  counter = counter + 1
  if isint(counter / 100) then
    simAddStatusbarMessage('Counter: ' .. counter)
  end
  if isint(counter / 10) then
    -- simAddStatusbarMessage('Point cloud: ' .. ptCloud)
  end

  -- Publish simulation time
  simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'Pioneer_p3dx_visible') then
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
```

```
  end

  -- Publish twist
  o = simGetObjectOrientation(robot01Handle, worldHandle)
  v, w = simGetObjectVelocity(robot01Handle)
  c = math.cos(o[3])
  s = math.sin(o[3])
  simExtRosInterface_publish(twistPub, {linear =
                    {x = c * v[1] + s * v[2],
                     y = c * v[2] - s * v[1],
                     z = v[3]},
  angular={x = w[1], y = w[2], z = w[3]}})

end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼263
    263▲ down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 3: Text file: ugv-05.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Text
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP scene - UGV 05 scenario

# General description
The UGV collects readings from VLP16 and publishes it in PointCloud2 message format ▼9
    9▲ . UGV is mobile and follows waypoints.

# Stereo vision parameters
Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. Both ▼12
    12▲ camera frames are published in TF.
```

Script 4: Lua script file: ugv-02–vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 02 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼22
    22▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4  ▼40
    40▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 =  ▼41
    41▲ pointCloud2 messages;
  -- Counters
  counter = 0

  -- Handles
  -- Frames
  worldHandle = simGetObjectHandle('frame00')
  -- robot01Handle = simGetObjectHandle('robot01')
  -- vlp16Handle = simGetObjectHandle('vlp16')
```

```
  vlp16Handle = simGetObjectHandle('vpl16')

  -- Tranform (TF)
  tfLinks = {}
  -- for i,link in ipairs {'vlp16'} do
  for i, link in ipairs {'vpl16'} do
    tfLinks[link] = simGetObjectHandle(link)
  end

  -- Base TF
  tf = {
    header = {
      stamp = 0,
      frame_id = 'frame00'
    },
    child_frame_id = '',
    transform = {
      -- ROS has definition x = front y = side z = up
      translation = {x = 0, y = 0, z = 0}, -- V-rep
      rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
    }
  }


  -- VLP16
  pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
  visionSensorHandles = {}
  for i = 1, 4, 1 do
    visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
  end

  -- Setting VLP16 parameters
  frequency = 20 -- Default model 5 Hz
  -- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only ▼83
    83▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼83
    83▲  (8)=displayed points are emissive
  options = 1 -- No display
  pointSize = 2
  coloringCloseAndFarDistance = {1, 4}
  displayScaling = 0.999 -- so that points do not appear to disappear in objects

  -- Reading VLP16 sensor data
  vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles, ▼90
    90▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼90
    90▲  pointCloudHandle)


  -- Init publishers and subscribers
  -- Init publishers
-- #S#
  -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1, ▼96
    96▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
  -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
```

```
  -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼98
    98▲ PointCloud2')
  -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼99
    99▲ VelodyneScan')
100 -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼100
    100▲ VelodynePacket')
  -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼101
    101▲ VelodyneScan')
-- #E#
  vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
105 --[[ #S#
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼106
    106▲ VelodynePacket')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
-- #E# --]]
end
110

-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
  -- Retrieve sensor data
  data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼114
    114▲ simGetSimulationTimeStep())
115 -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

  -- Debug options
  -- print('----------------')
120 -- print(bitand(debugOutput, 1))
  -- print(bitand(debugOutput, 2))
  -- print(bitand(debugOutput, 4))
  -- print('----------------')

125 -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
      simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
    else
130   pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
    end
  end

  -- Construct the point cloud from raw data: x, y and z coordinates for every ▼134
    134▲ point
135 -- Reference: From the original VPL_16 model script
  local pointCloudData = {}
  for i = 0, #data / 3 - 1, 1 do
    -- Temporary assignment to a vector d
    d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
140 -- Transformation to sensor frame
    -- No transformation results in point cloud rotated by 90 deg around z
    d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
    data[3 * i + 1] = d[1]
```

```
    data[3 * i + 2] = d[2]
    data[3 * i + 3] = d[3]
    table.insert(pointCloudData, d)
    -- print(d)
  end

  -- Display the detected points
  if displayPointCloud == 1 then
    simInsertPointsIntoPointCloud(pointCloud, 0, data)
  end

--[[ #S#
  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402, ▼164
    164▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
-- #S#
      -- table.insert(tpcl, {1, 1, 1})
      -- table.insert(tpcl, {1.0, 1.0, 1.0})
      -- table.insert(tpcl, {2, 2, 2})
-- #E#
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackFloats( ▼189
    189▲ tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼190
    190▲ simPackUInts(tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼191
    191▲ simPackUInts(data)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼192
```

```
    192▲ simPackUInts(tdata)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼193
    193▲ simPackUInts(pointCloudData)} -- CRASH
-- #S#
-- #E#

    if bitand(debugOutput, 2) == 2 then
      print('-*- VLP16 Raw START')
      print('tpcl')
      print(#tpcl)
      print('tdata')
      print(#tdata)
-- #S#
      -- print('h')
      -- print(type(h))
      -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua] SCRIPT  ▼206
    206▲ velodyneVPL...]:140: attempt to get length of global 'h' (a number value ▼206
    206▲ )
      -- print(type(tpcl))
      -- print(type(simPackUInts(tpcl)))
      -- print(type(simPackFloats(tpcl)))
      -- print(#simPackUInts(tpcl))
      -- print(type(tdata))
      -- print(tdata)
      -- for i, j in pairs(tdata) do
        -- print(i, j)
      -- end
      -- for i, j in pairs(data) do
        -- print(i, j)
      -- end
      -- print(#simPackUInts(tdata)) -- nil
      -- print('raw')
      -- print(type(pointCloudRawData.data))
      -- print(#pointCloudRawData.data)
      -- print('-*- VLP16 Raw END')
-- #E#
    end
    simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
    rmcounter = rmcounter + 1
  end
  if bitand(debugOutput, 2) == 2 then
    print('rmcounter')
    print(rmcounter)
    print('-*- VLP16 Raw END')
  end
-- #S#
-- #E#
-- #E# --]]

  -- Publish point cloud

  -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
  -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼241
```

```
  241▲ Software/V-REP/robot.lua
-- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
local pointCloudData = {}
--[[
-- The frame of VLP16 shifts all the points to that frame
-- If frame_id set to sensor frame in current configuration results in point  ▼246
  246▲ cloud shifted to the sensor position
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼247
  247▲ frame_id = "vlp16"}
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼248
  248▲ frame_id = "vpl16"}
--]]
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼250
  250▲ frame_id = "frame00"}
-- Describes the channels and their layout in the binary data blob
local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
pointCloudData['fields'] = {field_x, field_y, field_z}
pointCloudData['is_bigendian'] = false
pointCloudData['is_dense'] = true
pointCloudData['data'] = simPackFloats(data) --  Works
-- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the  ▼259
  259▲ data field being of type uint8
-- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼260
  260▲ and it differs in number of received points
--[[
-- Theoretically this should work, but the bytes offset still drop the messages
pointCloudData['height'] = 16 -- Ordered, 16 scan lines
pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData. ▼264
  264▲ height)
pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼265
  265▲ width / pointCloudData.height)
--]]
-- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
pointCloudData['height'] = 1 -- Unordered
-- Effectively number of cloud points, each having 3 coordinates
pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
-- Length of a point in bytes
pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼272
  272▲ width)
-- Length of a row in bytes
pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼274
  274▲ point_step']

if bitand(debugOutput, 4) == 1 then
  print('-- VLP16 PointCloud2 START')
  print(#pointCloudData.data)
  print(pointCloudData.width * pointCloudData.point_step)
  print(pointCloudData.point_step)
  print(#pointCloudData.data - math.floor(#pointCloudData.data / pointCloudData. ▼281
  281▲ width) * pointCloudData.width)
  print(#pointCloudData.data - pointCloudData.width * pointCloudData.point_step)
```

```
    print(pointCloudData.width)
-- #S#
    -- print(type(pointCloudData))
    -- print(type(tdata))
-- #E#
    print('-- VLP16 PointCloud2 END')
  end

  simExtRosInterface_publish(vlp16Pub, pointCloudData)

  -- Send transforms
  tf.header.stamp = simGetSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      position = simGetObjectPosition(handle, worldHandle)
      orientation = simGetObjectQuaternion(handle, worldHandle)
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = simGetObjectPosition(handle, vlp16Handle)
      orientation = simGetObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    simAddStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
```

```
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 5: Lua script file: ugv-05–robot01.pose2d-control.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 04 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message format. UGV is mobile using forward and backward movements. The raw message format is implemented but not used, since data formatting is not working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. Both camera frames are published in TF.
-- # * Robot travels along several waypoints in a loop
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  stopSimulation = -1 -- 0 = off; >0 = on, number of full cycles type A; <0 = on, number of full cycles type B
  counterSimulation = 0 -- Simulation counter offset
  -- counterStep = 10 -- Internal script handling step
  counterStep = 4 -- Internal script handling step
  -- Counters
  counter = 0
  -- Prefixes
  vrepPrefix = '/vrep/'
  rosParamPrefix = vrepPrefix .. 'init/'
  posePrefix = vrepPrefix .. 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos.dylib):
```

```
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
--    if (moduleName == 'RosInterface') then -- Use this for V-REP > 3.3.2
    if (moduleName == 'Ros') then -- This works with V-REP == 3.3.2
      pluginNotFound = false
    end
    index = index + 1
  end

  if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼58
    58▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼58
    58▲ {0.5,0,0,1,1,1})
  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    stereoVisionHandle = simGetObjectHandle('stereo_vision') -- Stereo vision  ▼64
    64▲ system
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    -- object01Handle = simGetObjectHandle('object01') -- Cylinder
    -- object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- --[[ #S#
    -- Waypoints
    waypointsHandlesNames = {}
    waypointHandle = simGetObjectHandle('waypoint')
    for i = 1, 10, 1 do
      waypointsHandlesNames[i] = 'waypoint0' .. i - 1
    end
    waypointsHandles = {}
    -- Asign waypoint handles to array
    for i, link in ipairs(waypointsHandlesNames) do
      waypointsHandles[link] = simGetObjectHandle(link)
    end
    -- #E# --]]

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼88
    88▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Initial and gain velocity
    velocity = 10

    -- Robot 01 + stereo vision
```

```
    -- Tranforms
    tfLinks = {}
    -- Publish with rotated camera frames
    for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼99
    99▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼99
    99▲ camera01', 'camera02', 'camera01r', 'camera02r'} do
100 -- Publish without rotated camera frames
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼101
    101▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼101
    101▲ camera01', 'camera02'} do
    -- Publish also the VPL16 sensor
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼103
    103▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼103
    103▲ camera01', 'camera02', 'vpl16'} do
    -- Alternative to publishing poses
105 -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼105
    105▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼105
    105▲ camera01', 'camera02', 'waypoint00', 'waypoint01', 'waypoint02', ' ▼105
    105▲ waypoint03', 'waypoint04', 'waypoint05', 'waypoint06', 'waypoint07', ' ▼105
    105▲ waypoint08', 'waypoint09'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

    -- Frames
110 worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
    camera01Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera01' ▼114
    114▲ , 1, simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
115 camera02Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera02' ▼115
    115▲ , 1, simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher(vrepPrefix .. 'joints/robot01/ ▼117
    117▲ state', 1, simros_strmcmd_get_joint_state, sim_handle_all, -1, '')
    -- Publish robot01 pose
    -- Argument -1 makes values absolute, could be also frame00 as world frame ▼119
    119▲ reference
120 -- robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼120
    120▲ simros_strmcmd_get_object_pose, robot01Handle, -1, '')
    robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼121
    121▲ simros_strmcmd_get_object_pose, robot01Handle, worldHandle, 'frame00')
    -- Publish waypoints poses
    -- waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼123
    123▲ simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')

125

    -- Init subscribers
    robot01StatesSub = simExtROS_enableSubscriber(vrepPrefix .. 'joints/robot01/ ▼127
    127▲ control', 1, simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

    -- Simulation time
130 -- clockPub = simExtROS_enablePublisher('/clock', 1, 'rosgraph_msgs/Clock')
```

```
    clockPub = simExtRosInterface_advertise('/clock', 'rosgraph_msgs/Clock')

    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }


    -- Waypoints 01,05,04,03,02
    -- waypointsloopA = {1, 5, 4, 3, 2}
    -- Waypoints 01,02,03,04,05
    waypointsloopA = {1, 2, 3, 4, 5}
    -- Correct index
    for i in ipairs(waypointsloopA) do
      waypointsloopA[i] = waypointsloopA[i] + 1
    end
    waypointcounter = 1
    -- Tolerance position
    tp = 0.3
    -- Tolerance heading
    to = 15 / 360
    -- Control gains
    -- Control gain position
    kp = 10.0
    -- Control gain orientation
    ko = 18.0
    -- Wheel base
    L = 0.1655
    l = 2 * L
    -- Wheel radius
    r = 0.975
    -- Initial velocities
    v = 0
    w = 0

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})
  simExtRosInterface_publish(clockPub, {clock = simGetSystemTime()})
```

```
185   -- Send transforms
      tf.header.stamp = simGetSimulationTime()
      for link, handle in pairs(tfLinks) do
        -- Get parent
        if (link == 'robot01') then
190       tf.header.frame_id = 'frame00'
          -- Get pose relative to world frame
          p = simGetObjectPosition(handle, worldHandle)
          o = simGetObjectQuaternion(handle, worldHandle)
        -- --[[ -- Uncomment this block comment for publishing all stereo vision  ▼194
        194▲ system handles relative to robot
195     elseif string.find(link, 'camera') then
          tf.header.frame_id = 'stereo_vision'
          -- Get pose relative to stereo vision system
          p = simGetObjectPosition(handle, stereoVisionHandle)
          o = simGetObjectQuaternion(handle, stereoVisionHandle)
200     --]] -- End of block comment
        else
          tf.header.frame_id = 'robot01'
          -- Get pose relative to robot
          p = simGetObjectPosition(handle, robot01Handle)
205       o = simGetObjectQuaternion(handle, robot01Handle)
        end

        -- Update TF
        tf.child_frame_id = link
210     tf.transform.translation.x = p[1]
        tf.transform.translation.y = p[2]
        tf.transform.translation.z = p[3]
        tf.transform.rotation.x = o[1]
        tf.transform.rotation.y = o[2]
215     tf.transform.rotation.z = o[3]
        tf.transform.rotation.w = o[4]
        simExtRosInterface_sendTransform(tf)
      end

220 -- --[[
      -- Get robot position and orientation
      -- rp = simGetObjectPosition(robot01Handle, robot01Handle)
      -- ro = simGetObjectQuaternion(robot01Handle, robot01Handle)
      rp = simGetObjectPosition(robot01Handle, worldHandle)
225   ro = simGetObjectQuaternion(robot01Handle, worldHandle)

      -- Assign waypoint from loop
      waypointHandle = waypointsHandles[ waypointsHandlesNames[ waypointsloopA[ ▼228
        228▲ waypointcounter] ] ]
      -- print(waypointsloopA[waypointcounter])
230

      -- Get waypoint position and orientation
      wp = simGetObjectPosition(waypointHandle, robot01Handle)
      wo = simGetObjectQuaternion(waypointHandle, robot01Handle)
      -- wp = simGetObjectPosition(waypointHandle, worldHandle)
```

```
235   -- wo = simGetObjectQuaternion(waypointHandle, worldHandle)

      -- Pose 2D
      -- Error position
      -- dx = (wp[1] - rp[1])
240   -- dy = (wp[2] - rp[2])
      -- ep = math.sqrt(dx^2 + dy^2)
      ep = math.sqrt(wp[1]^2 + wp[2]^2)
      -- Error heading
      -- eh = math.atan(dy / dx)
245   -- eh = math.atan2(dy, dx)
      -- eh = math.atan2(wp[2], wp[1])
      eh = math.fmod(math.atan2(wp[2], wp[1]) + math.pi, 2 * math.pi) - math.pi
      -- Error final orientation
      eo = wo[3] - ro[3]
250
      -- Control
      -- if math.abs(eh) <= 5 * to and math.abs(eo) <= to and ep <= tp then
      if math.abs(eo) <= to and ep <= tp then
        -- Waypoint reached
255     waypointcounter = waypointcounter + 1
        -- Reset counter if overflow
        if waypointcounter > #waypointsloopA then
          waypointcounter = 0
        end
260     simAddStatusbarMessage('Waypoint reached. Moving to next waypoint: ' ..  ▼260
       260▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
      elseif math.abs(eh) <= 2 * to and math.abs(ep) <= tp then
        -- Correct heading
        simAddStatusbarMessage('Correcting heading towards: ' .. waypointsHandlesNames ▼263
       263▲ [waypointsloopA[waypointcounter]])
        vl = - ko * 2.5 * eo
265     vr = ko * 2.5 * eo
      elseif math.abs(eh) <= 4 * to and math.abs(ep) <= 3 * tp then
        -- Increased gains near target
        simAddStatusbarMessage('Using increased gains towards: ' ..  ▼268
       268▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
        v = kp * 20 * (wp[1] - tp)
270     w = ko * 40 * eh
        vl = v / r - w * L / r
        vr = v / r + w * L / r
      elseif math.abs(v) < velocity / 5 then
        -- Accelerated control mode
275     v = kp * 3 * (wp[1] - tp)
        w = ko * 2 * eh
        vl = v / r - w * L / r
        vr = v / r + w * L / r
        simAddStatusbarMessage('Accelerated mode reaching waypoint: ' ..  ▼279
       279▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
280   else
        -- Normal control mode
        v = kp * (wp[1] - tp)
        w = ko * eh
```

```
    vl = v / r - w * L / r
    vr = v / r + w * L / r
    simAddStatusbarMessage('Reaching waypoint: ' .. waypointsHandlesNames[ ▼286
    286▲ waypointsloopA[waypointcounter]])
  end

  -- Saturation
  velocities = {}
  -- for i, j in ipairs {'vl', 'vr'} do
    if vl > velocity then
      vl = velocity
    elseif vl < -velocity then
      vl = -velocity
    end
    velocities[1] = vl
    if vr > velocity then
      vr = velocity
    elseif vr < -velocity then
      vr = -velocity
    end
    velocities[2] = vr
  -- end

  if isint(counter / counterStep) then
    simAddStatusbarMessage('### Loop: ' .. counter .. ' ###')
    simAddStatusbarMessage('Robot: Position: X: ' .. rp[1] .. ', Y: ' .. rp[2] .. ▼308
    308▲ ', Z: ' .. rp[3])
    simAddStatusbarMessage('Robot: Orientation: A: ' .. ro[1] .. ', B: ' .. ro[2] ▼309
    309▲ .. ', G: ' .. ro[3])
    simAddStatusbarMessage('Waypoint: Position: X: ' .. wp[1] .. ', Y: ' .. wp[2] ▼310
    310▲ .. ', Z: ' .. wp[3])
    simAddStatusbarMessage('Waypoint: Orientation: A: ' .. wo[1] .. ', B: ' .. wo ▼311
    311▲ [2] .. ', G: ' .. wo[3])
    simAddStatusbarMessage('Velocity: V: ' .. v .. ', W: ' .. w)
    simAddStatusbarMessage('Velocity: L: ' .. vl .. ', R: ' .. vr)
    simAddStatusbarMessage('Errors: H: ' .. eh .. ', P: ' .. ep .. ' (X: ' .. wp ▼314
    314▲ [1] .. ', Y: ' .. wp[2] .. '), O: ' .. eo)
    simAddStatusbarMessage('Tolerance: H: ' .. to .. ', P: ' .. tp .. ', O: ' .. ▼315
    315▲ to)
  end

  -- Get joint handles
  jointHandles = {}
  -- Pioneer_p3dx_leftMotor
  for i, m in ipairs {'left', 'right'} do
    -- Motor velocity for fixed wheels
    joint = 'Pioneer_p3dx_' .. m .. 'Motor'
    jointHandles[joint] = simGetObjectHandle(joint)
    simSetJointTargetVelocity(jointHandles[joint], velocities[i])
    -- simAddStatusbarMessage('Velocity' .. i .. ': ' .. velocities[i])
  end
--]]
```

```
--  --[[
  -- Stop simulation condition
  if waypointcounter == 0 then
    simAddStatusbarMessage('Full cycle achieved')
    counterSimulation = counterSimulation + 1
    if stopSimulation > 0 and counterSimulation >= stopSimulation then
      simAddStatusbarMessage('Stop simulation')
      simStopSimulation()
    end
  end
--]]

  -- Update counter
  counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼349
    349▲  down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 6: Lua script file: ugv-05–robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 05 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
    10▲ format. UGV is mobile using forward and backward movements. The raw ▼10
    10▲ message format is implemented but not used, since data formatting is not ▼10
    10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. ▼11
    11▲ Both camera frames are published in TF.
-- # * Robot optionally travels along several waypoints in a loop using a simple ▼12
    12▲ controller
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  stopSimulation = 1 -- 0 = off; >0 = on, number of full cycles type A; <0 = on, ▼29
    29▲ number of full cycles type B (not implemented yet)
  counterSimulation = 0 -- Simulation counter offset
  followWaypoints = 0 -- 1 = on; 0 = off;
  -- counterStep = 10 -- Internal script handling step
  counterStep = 4 -- Internal script handling step
  -- Counters
  counter = 0
  -- Prefixes
  vrepPrefix = '/vrep/'
  rosParamPrefix = vrepPrefix .. 'init/'
  posePrefix = vrepPrefix .. 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
```

```
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼47
    47▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
--    if (moduleName == 'RosInterface') then -- Use this for V-REP > 3.3.2
    if (moduleName == 'Ros') then -- This works with V-REP == 3.3.2
      pluginNotFound = false
    end
    index = index + 1
  end

  if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼59
    59▲ run properly', sim_dlgstyle_ok, false, nil, {0.8, 0, 0, 0, 0, 0}, {0.5, 0,  ▼59
    59▲ 0, 1, 1, 1})
  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    stereoVisionHandle = simGetObjectHandle('stereo_vision') -- Stereo vision  ▼65
    65▲ system
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    -- object01Handle = simGetObjectHandle('object01') -- Cylinder
    -- object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- --[[ #S#
    -- Waypoints
    waypointsHandlesNames = {}
    waypointHandle = simGetObjectHandle('waypoint')
    for i = 1, 10, 1 do
      waypointsHandlesNames[i] = 'waypoint0' .. i - 1
    end
    waypointsHandles = {}
    -- Asign waypoint handles to array
    for i, link in ipairs(waypointsHandlesNames) do
      waypointsHandles[link] = simGetObjectHandle(link)
    end
    -- #E# --]]

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼89
    89▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Initial and gain velocity
    velocity = 10
```

```
95
    -- Robot 01 + stereo vision
    -- Tranforms
    tfLinks = {}
    -- Publish with rotated camera frames
100 for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼100
    100▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼100
    100▲ camera01', 'camera02', 'camera01r', 'camera02r'} do
    -- Publish without rotated camera frames
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼102
    102▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼102
    102▲ camera01', 'camera02'} do
    -- Publish also the VPL16 sensor
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼104
    104▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼104
    104▲ camera01', 'camera02', 'vpl16'} do
105 -- Alternative to publishing poses
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼106
    106▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼106
    106▲ camera01', 'camera02', 'waypoint00', 'waypoint01', 'waypoint02', ' ▼106
    106▲ waypoint03', 'waypoint04', 'waypoint05', 'waypoint06', 'waypoint07', ' ▼106
    106▲ waypoint08', 'waypoint09'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

110 -- Frames
    worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
115 camera01Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera01' ▼115
    115▲ , 1, simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    camera02Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera02' ▼116
    116▲ , 1, simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher(vrepPrefix .. 'joints/robot01/ ▼118
    118▲ state', 1, simros_strmcmd_get_joint_state, sim_handle_all, -1, '')
    -- Publish robot01 pose
120 -- Argument -1 makes values absolute, could be also frame00 as world frame ▼120
    120▲ reference
    -- robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼121
    121▲ simros_strmcmd_get_object_pose, robot01Handle, -1, '')
    robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼122
    122▲ simros_strmcmd_get_object_pose, robot01Handle, worldHandle, 'frame00')
    -- Publish waypoints poses
    -- waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼124
    124▲ simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')
125


    -- Init subscribers
    robot01StatesSub = simExtROS_enableSubscriber(vrepPrefix .. 'joints/robot01/ ▼128
    128▲ control', 1, simros_strmcmd_set_joint_state, sim_handle_all, -1, '')
```

```
    -- Simulation time
    -- clockPub = simExtROS_enablePublisher('/clock', 1, 'rosgraph_msgs/Clock')
    clockPub = simExtRosInterface_advertise('/clock', 'rosgraph_msgs/Clock')

    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

    -- Waypoints and control initialization
    -- Waypoints 01,05,04,03,02
    -- waypointsloopA = {1, 5, 4, 3, 2}
    -- Waypoints 01,02,03,04,05
    -- waypointsloopA = {1, 2, 3, 4, 5}
    -- Waypoints 01,02,03,04,05,01
    waypointsloopA = {1, 2, 3, 4, 5, 1}
    -- Correct index -- Waypoint names start with 00, Lua has initial index at 1
    for i in ipairs(waypointsloopA) do
      waypointsloopA[i] = waypointsloopA[i] + 1
    end
    waypointcounter = 1
    -- Tolerance position
    tp = 0.3
    -- Tolerance heading
    to = 15 / 360
    -- Control gains
    -- Control gain position
    kp = 8.0
    -- Control gain orientation
    ko = 16.0
    -- Wheel base
    L = 0.1655
    l = 2 * L
    -- Wheel radius
    r = 0.975
    -- r = 0.0975 -- Documentation
    -- Initial velocities
    v = 0
    w = 0

  end
end

-- # Runtime -- Actuation
```

```
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})
  -- simExtRosInterface_publish(clockPub, {clock = simGetSystemTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to world frame
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    -- --[[ -- Uncomment this block comment for publishing all stereo vision  ▼198
    198▲ system handles relative to robot
    elseif string.find(link, 'camera') then
      tf.header.frame_id = 'stereo_vision'
      -- Get pose relative to stereo vision system
      p = simGetObjectPosition(handle, stereoVisionHandle)
      o = simGetObjectQuaternion(handle, stereoVisionHandle)
    --]] -- End of block comment
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

  if followWaypoints == 1 then
    -- --[[
    -- Get robot position and orientation
    -- rp = simGetObjectPosition(robot01Handle, robot01Handle)
    -- ro = simGetObjectQuaternion(robot01Handle, robot01Handle)
    rp = simGetObjectPosition(robot01Handle, worldHandle)
    ro = simGetObjectQuaternion(robot01Handle, worldHandle)

    -- Assign waypoint from loop
    waypointHandle = waypointsHandles[ waypointsHandlesNames[ waypointsloopA[ ▼233
    233▲ waypointcounter] ] ]
```

```
    -- print(waypointsloopA[waypointcounter])
235
    -- Get waypoint position and orientation
    wp = simGetObjectPosition(waypointHandle, robot01Handle)
    wo = simGetObjectQuaternion(waypointHandle, robot01Handle)
    -- wp = simGetObjectPosition(waypointHandle, worldHandle)
240 -- wo = simGetObjectQuaternion(waypointHandle, worldHandle)

    -- Pose 2D
    -- Error position
    ep = math.sqrt(wp[1]^2 + wp[2]^2)
245 -- Error heading
    eh = math.fmod(math.atan2(wp[2], wp[1]) + math.pi, 2 * math.pi) - math.pi
    -- Error final orientation
    eo = wo[3] - ro[3]

250 -- Control
    if math.abs(ep) <= tp then
    -- if math.abs(eo) <= to and math.abs(ep) <= tp then
      -- Waypoint reached
      waypointcounter = waypointcounter + 1
255   -- Reset counter if overflow
      if waypointcounter > #waypointsloopA then
        waypointcounter = 1
        simAddStatusbarMessage('Full cycle achieved')
        counterSimulation = counterSimulation + 1
260   end
      if isint(counter / counterStep) then
        simAddStatusbarMessage('Waypoint reached. Moving to next waypoint: ' .. ▼262
    262▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
      end
    elseif math.abs(ep) <= 3 * tp then
265   -- Increased gains near target
      if isint(counter / counterStep) then
        simAddStatusbarMessage('Using increased gains towards: ' .. ▼267
    267▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
      end
      v = kp * 18 * (wp[1] - tp)
270   w = ko * 40 * eh
      vl = v / r - w * L / r
      vr = v / r + w * L / r
    else
      -- Normal control mode
275   v = kp * (wp[1] - tp)
      w = ko * eh
      vl = v / r - w * L / r
      vr = v / r + w * L / r
      if isint(counter / counterStep) then
280     simAddStatusbarMessage('Reaching waypoint: ' .. waypointsHandlesNames[ ▼280
    280▲ waypointsloopA[waypointcounter]])
      end
    end
```

```
    -- Saturation
    velocities = {}
    -- for i, j in ipairs {'vl', 'vr'} do
      if vl > velocity then
        vl = velocity
      elseif vl < -velocity then
        vl = -velocity
      end
      velocities[1] = vl
      if vr > velocity then
        vr = velocity
      elseif vr < -velocity then
        vr = -velocity
      end
      velocities[2] = vr
    -- end

    if isint(counter / counterStep) then
      simAddStatusbarMessage('### Loop: ' .. counter .. ' ###')
      simAddStatusbarMessage('Robot: Position: X: ' .. rp[1] .. ', Y: ' .. rp[2] ▼303
    303▲ .. ', Z: ' .. rp[3])
      simAddStatusbarMessage('Robot: Orientation: A: ' .. ro[1] .. ', B: ' .. ro ▼304
    304▲ [2] .. ', G: ' .. ro[3])
      simAddStatusbarMessage('Waypoint: Position: X: ' .. wp[1] .. ', Y: ' .. wp ▼305
    305▲ [2] .. ', Z: ' .. wp[3])
      simAddStatusbarMessage('Waypoint: Orientation: A: ' .. wo[1] .. ', B: ' ..  ▼306
    306▲ wo[2] .. ', G: ' .. wo[3])
      simAddStatusbarMessage('Velocity: V: ' .. v .. ', W: ' .. w)
      simAddStatusbarMessage('Velocity: L: ' .. vl .. ', R: ' .. vr)
      simAddStatusbarMessage('Errors: H: ' .. eh .. ', P: ' .. ep .. ' (X: ' .. wp ▼309
    309▲ [1] .. ', Y: ' .. wp[2] .. '), O: ' .. eo)
      simAddStatusbarMessage('Tolerance: H: ' .. to .. ', P: ' .. tp .. ', O: ' .. ▼310
    310▲  to)
    end

    -- Get joint handles
    jointHandles = {}
    -- Pioneer_p3dx_leftMotor
    for i, m in ipairs {'left', 'right'} do
      -- Motor velocity for fixed wheels
      joint = 'Pioneer_p3dx_' .. m .. 'Motor'
      jointHandles[joint] = simGetObjectHandle(joint)
      simSetJointTargetVelocity(jointHandles[joint], velocities[i])
      -- simAddStatusbarMessage('Velocity' .. i .. ': ' .. velocities[i])
    end
  --]]
  end

-- --[[
  -- Stop simulation condition
  if stopSimulation > 0 and counterSimulation >= stopSimulation then
      simAddStatusbarMessage('Stop simulation')
      simStopSimulation()
```

```
  end
--]]

  -- Update counter
  counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼341
    341▲ down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 7: Lua script file: ugv-04–robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 04 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
    10▲ format. UGV is mobile using forward and backward movements. The raw ▼10
    10▲ message format is implemented but not used, since data formatting is not ▼10
    10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. ▼11
    11▲ Both camera frames are published in TF.
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  stopSimulation = 1 -- 0 = off; >0 = on, number of full cycles
  counterSimulation = 0 -- Simulation counter offset
  counterStep = 10 -- Internal script handling step
  -- Counters
  counter = 0
  -- Prefixes
  vrepPrefix = '/vrep/'
  rosParamPrefix = vrepPrefix .. 'init/'
  posePrefix = vrepPrefix .. 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼44
    44▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
--    if (moduleName == 'RosInterface') then -- Use this for V-REP > 3.3.2
```

```
    if (moduleName == 'Ros') then -- This works with V-REP == 3.3.2
      pluginNotFound = false
    end
    index = index + 1
end

if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼56
    56▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼56
    56▲ {0.5,0,0,1,1,1})
else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    stereoVisionHandle = simGetObjectHandle('stereo_vision') -- Stereo vision  ▼62
    62▲ system
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    -- object01Handle = simGetObjectHandle('object01') -- Cylinder
    -- object02Handle = simGetObjectHandle('object02') -- Cuboid

    --[[ #S#
    -- Waypoints
    waypointsHandlesNames = {}
    waypointHandle = simGetObjectHandle('waypoint')
    for i = 1, 10, 1 do
      waypointsHandlesNames[i] = 'waypoint0' .. i - 1
    end
    waypointsHandles = {}
    -- Asign waypoint handles to array
    for i, link in ipairs(waypointsHandlesNames) do
      waypointsHandles[link] = simGetObjectHandle(link)
    end
    -- #E# --]]

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼86
    86▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Initial and gain velocity
    velocity = -5

    -- Robot 01 + stereo vision
    -- Tranforms
    tfLinks = {}
    for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼96
```

```
    96▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼96
    96▲ camera01', 'camera02'} do
    -- Publish also the VPL16 sensor
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼98
    98▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼98
    98▲ camera01', 'camera02', 'vpl16'} do
    -- Alternative to publishing poses
100 -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼100
    100▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼100
    100▲ camera01', 'camera02', 'waypoint00', 'waypoint01', 'waypoint02', ' ▼100
    100▲ waypoint03', 'waypoint04', 'waypoint05', 'waypoint06', 'waypoint07', ' ▼100
    100▲ waypoint08', 'waypoint09'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

    -- Frames
105 worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
    camera01Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera01' ▼109
    109▲ , 1, simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
110 camera02Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera02' ▼110
    110▲ , 1, simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher(vrepPrefix .. 'joints/robot01/ ▼112
    112▲ state', 1, simros_strmcmd_get_joint_state, sim_handle_all, -1, '')
    -- Publish robot01 pose
    -- Argument -1 makes values absolute, could be also frame00 as world frame ▼114
    114▲ reference
115 -- robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼115
    115▲ simros_strmcmd_get_object_pose, robot01Handle, -1, '')
    robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼116
    116▲ simros_strmcmd_get_object_pose, robot01Handle, worldHandle, 'frame00')
    -- Publish waypoints poses
    -- waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼118
    118▲ simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')

120 --[[ -- #S#
    -- Does work
    waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼122
    122▲ simros_strmcmd_get_object_pose, waypointsHandles['waypoint00'], ▼122
    122▲ worldHandle, 'frame00')
    -- Does not work
    waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼124
    124▲ simros_strmcmd_get_object_pose, waypointsHandles, -1, 'frame00')
125 -- Does work, but inefficient
    waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼126
    126▲ simros_strmcmd_get_object_pose, waypointsHandles['waypoint00'], ▼126
    126▲ worldHandle, '')
    waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼127
    127▲ simros_strmcmd_get_object_pose, waypointsHandles['waypoint01'], ▼127
    127▲ waypointsHandles['waypoint02'], waypointsHandles['waypoint03'], ▼127
```

```
    127▲ waypointsHandles['waypoint04'], waypointsHandles['waypoint05'],  ▼127
    127▲ waypointsHandles['waypoint06'], waypointsHandles['waypoint07'],  ▼127
    127▲ waypointsHandles['waypoint08'], waypointsHandles['waypoint09'],  ▼127
    127▲ waypointsHandles['waypoint010'], worldHandle, '')
    waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1,  ▼128
    128▲ simros_strmcmd_get_object_pose, waypointsHandles['waypoint01'],  ▼128
    128▲ waypointsHandles['waypoint00'], '')
    for i, j in ipairs(waypointsHandles) do
      -- Does compile but no display
      waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1,  ▼131
    131▲ simros_strmcmd_get_object_pose, waypointsHandles[j], worldHandle, ' ▼131
    131▲ frame00')
      -- Does compile but no display
      -- print(j)
      waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoint' .. i, 1,  ▼134
    134▲ simros_strmcmd_get_object_pose, waypointsHandles[j], worldHandle, ' ▼134
    134▲ frame00')
    end
    -- #E# --]]

    -- Init subscribers
    robot01StatesSub = simExtROS_enableSubscriber(vrepPrefix .. 'joints/robot01/ ▼139
    139▲ control', 1, simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

    waypointcounter = 0

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
```

```
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to world frame
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    -- --[[ -- Uncomment this block comment for publishing all stereo vision  ▼175
    175▲ system handles relative to robot
    elseif string.find(link, 'camera') then
      tf.header.frame_id = 'stereo_vision'
      -- Get pose relative to stereo vision system
      p = simGetObjectPosition(handle, stereoVisionHandle)
      o = simGetObjectQuaternion(handle, stereoVisionHandle)
    --]] -- End of block comment
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

-- Does not work in non-threaded child script, the syntax is sane
--[[
  if isint(counter / 10) then
    if (waypointcounter == 1) then
      waypointcounter = 0
    end
    deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles['waypoint0'  ▼207
    207▲ .. waypointcounter], 3, 1, 0, nil)
    -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[ ▼208
    208▲ waypointcounter], 3, 1, 0, nil)
    waypointcounter = waypointcounter + 1
  end
--]]

-- --[[
  -- Set a loop based on position
  if isint(counter / counterStep) then
    simAddStatusbarMessage('Velocity: ' .. velocity)
  end

  -- Set velocity based on loop counter
```

```
  if isint(counter / (10 * counterStep)) then
    simAddStatusbarMessage('Counter: ' .. counter)
    velocity = -1 * velocity
    --[[ #S#
    -- if ((counter + 100) % 200 == 0) then
    -- if (counter % 200 == 0) then
      -- velocity = 3.0
    -- else
      -- velocity = -3.0
    -- end
    -- #E# --]]
    -- Get joint handles
    jointHandles = {}
    -- Pioneer_p3dx_leftMotor
    for i, m in ipairs{'left', 'right'} do
      -- Motor velocity for fixed wheels
      joint = 'Pioneer_p3dx_' .. m .. 'Motor'
      jointHandles[joint] = simGetObjectHandle(joint)
      simSetJointTargetVelocity(jointHandles[joint], velocity)
    end
    simAddStatusbarMessage('Velocity: ' .. velocity)
  end
-- ]]

  -- Stop simulation condition
  if counter == (20 * counterStep) then
    simAddStatusbarMessage('Full cycle achieved')
    counterSimulation = counterSimulation + 1
    if stopSimulation > 0 and counterSimulation >= stopSimulation then
      simAddStatusbarMessage('Stop simulation')
      simStopSimulation()
    end
  end

  -- Update counter
  counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼261
    261▲ down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 8: Lua script file: ugv-03–vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 03 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼22
    22▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4  ▼40
    40▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 =  ▼41
    41▲ pointCloud2 messages;
  -- Counters
  counter = 0

  -- Handles
  -- Frames
  worldHandle = simGetObjectHandle('frame00')
  -- robot01Handle = simGetObjectHandle('robot01')
  -- vlp16Handle = simGetObjectHandle('vlp16')
```

```
  vlp16Handle = simGetObjectHandle('vpl16')

  -- Tranform (TF)
  tfLinks = {}
  -- for i,link in ipairs {'vlp16'} do
  for i, link in ipairs {'vpl16'} do
    tfLinks[link] = simGetObjectHandle(link)
  end

  -- Base TF
  tf = {
    header = {
      stamp = 0,
      frame_id = 'frame00'
    },
    child_frame_id = '',
    transform = {
      -- ROS has definition x = front y = side z = up
      translation = {x = 0, y = 0, z = 0}, -- V-rep
      rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
    }
  }


  -- VLP16
  pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
  visionSensorHandles = {}
  for i = 1, 4, 1 do
    visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
  end

  -- Setting VLP16 parameters
  frequency = 10 -- Default model 5 Hz
  -- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only  ▼83
    83▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼83
    83▲  (8)=displayed points are emissive
  options = 1 -- No display
  pointSize = 2
  coloringCloseAndFarDistance = {1, 4}
  displayScaling = 0.999 -- so that points do not appear to disappear in objects

  -- Reading VLP16 sensor data
  vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles,  ▼90
    90▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼90
    90▲  pointCloudHandle)


  -- Init publishers and subscribers
  -- Init publishers
-- #S#
  -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1,  ▼96
    96▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
  -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
```

```
  -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼98
    98▲ PointCloud2')
  -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼99
    99▲ VelodyneScan')
100 -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼100
    100▲ VelodynePacket')
  -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼101
    101▲ VelodyneScan')
-- #E#
  vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
105 --[[ #S#
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼106
    106▲ VelodynePacket')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
-- #E# --]]
end
110
-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
  -- Retrieve sensor data
  data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼114
    114▲ simGetSimulationTimeStep())
115 -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

  -- Debug options
  -- print('----------------')
120 -- print(bitand(debugOutput, 1))
  -- print(bitand(debugOutput, 2))
  -- print(bitand(debugOutput, 4))
  -- print('----------------')

125 -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
      simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
    else
130   pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
    end
  end

  -- Construct the point cloud from raw data: x, y and z coordinates for every ▼134
    134▲ point
135 -- Reference: From the original VPL_16 model script
  local pointCloudData = {}
  for i = 0, #data / 3 - 1, 1 do
    -- Temporary assignment to a vector d
    d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
140 -- Transformation to sensor frame
    -- No transformation results in point cloud rotated by 90 deg around z
    d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
    data[3 * i + 1] = d[1]
```

```
    data[3 * i + 2] = d[2]
    data[3 * i + 3] = d[3]
    table.insert(pointCloudData, d)
    -- print(d)
  end

  -- Display the detected points
  if displayPointCloud == 1 then
    simInsertPointsIntoPointCloud(pointCloud, 0, data)
  end

--[[ #S#
  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402, ▼164
    164▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
-- #S#
      -- table.insert(tpcl, {1, 1, 1})
      -- table.insert(tpcl, {1.0, 1.0, 1.0})
      -- table.insert(tpcl, {2, 2, 2})
-- #E#
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackFloats( ▼189
    189▲ tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼190
    190▲ simPackUInts(tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼191
    191▲ simPackUInts(data)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼192
```

```
    192▲ simPackUInts(tdata)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼193
    193▲ simPackUInts(pointCloudData)} -- CRASH
-- #S#
-- #E#

    if bitand(debugOutput, 2) == 2 then
      print('-*- VLP16 Raw START')
      print('tpcl')
      print(#tpcl)
      print('tdata')
      print(#tdata)
-- #S#
      -- print('h')
      -- print(type(h))
      -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua] SCRIPT  ▼206
    206▲ velodyneVPL...]:140: attempt to get length of global 'h' (a number value ▼206
    206▲ )
      -- print(type(tpcl))
      -- print(type(simPackUInts(tpcl)))
      -- print(type(simPackFloats(tpcl)))
      -- print(#simPackUInts(tpcl))
      -- print(type(tdata))
      -- print(tdata)
      -- for i, j in pairs(tdata) do
        -- print(i, j)
      -- end
      -- for i, j in pairs(data) do
        -- print(i, j)
      -- end
      -- print(#simPackUInts(tdata)) -- nil
      -- print('raw')
      -- print(type(pointCloudRawData.data))
      -- print(#pointCloudRawData.data)
      -- print('-*- VLP16 Raw END')
-- #E#
    end
    simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
    rmcounter = rmcounter + 1
  end
  if bitand(debugOutput, 2) == 2 then
    print('rmcounter')
    print(rmcounter)
    print('-*- VLP16 Raw END')
  end
-- #S#
-- #E#
-- #E# --]]

  -- Publish point cloud

  -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
  -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼241
```

```
  241▲ Software/V-REP/robot.lua
-- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
local pointCloudData = {}
--[[
-- The frame of VLP16 shifts all the points to that frame
-- If frame_id set to sensor frame in current configuration results in point  ▼246
  246▲ cloud shifted to the sensor position
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼247
  247▲ frame_id = "vlp16"}
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼248
  248▲ frame_id = "vpl16"}
--]]
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼250
  250▲ frame_id = "frame00"}
-- Describes the channels and their layout in the binary data blob
local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
pointCloudData['fields'] = {field_x, field_y, field_z}
pointCloudData['is_bigendian'] = false
pointCloudData['is_dense'] = true
pointCloudData['data'] = simPackFloats(data) --  Works
-- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the  ▼259
  259▲ data field being of type uint8
-- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼260
  260▲ and it differs in number of received points
--[[
-- Theoretically this should work, but the bytes offset still drop the messages
pointCloudData['height'] = 16 -- Ordered, 16 scan lines
pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData. ▼264
  264▲ height)
pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼265
  265▲ width / pointCloudData.height)
--]]
-- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
pointCloudData['height'] = 1 -- Unordered
-- Effectively number of cloud points, each having 3 coordinates
pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
-- Length of a point in bytes
pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼272
  272▲ width)
-- Length of a row in bytes
pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼274
  274▲ point_step']

if bitand(debugOutput, 4) == 1 then
  print('-- VLP16 PointCloud2 START')
  print(#pointCloudData.data)
  print(pointCloudData.width * pointCloudData.point_step)
  print(pointCloudData.point_step)
  print(#pointCloudData.data - math.floor(#pointCloudData.data / pointCloudData. ▼281
  281▲ width) * pointCloudData.width)
  print(#pointCloudData.data - pointCloudData.width * pointCloudData.point_step)
```

```
    print(pointCloudData.width)
-- #S#
    -- print(type(pointCloudData))
    -- print(type(tdata))
-- #E#
    print('-- VLP16 PointCloud2 END')
  end

  simExtRosInterface_publish(vlp16Pub, pointCloudData)

  -- Send transforms
  tf.header.stamp = simGetSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      position = simGetObjectPosition(handle, worldHandle)
      orientation = simGetObjectQuaternion(handle, worldHandle)
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = simGetObjectPosition(handle, vlp16Handle)
      orientation = simGetObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    simAddStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
```

```
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 9: Lua script file: ugv-uav-01–robot02.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 07 -- Robot02 -- Quadcopter
-- # Robot script
-- # Notes:
-- # * Preparation for a general UGV and UAV collaborative approach
--! @todo Modify script parameters and settings to overrule external input
--! @todo Add way point navigation mode instead of target at a specific height
--! @reference Based on original V-REP script
-- ###

-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  settingUseInternalCameras = 0 -- 0 = off; 1 = on
  settingFakeShadow = 0 -- 0 = off; 1 = on
  settingAnimatePropellers = 0 -- 0 = off; 1 = on
  settingUseTarget = 0 -- 0 = off; 1 = on
  -- Make sure we have version 2.4.13 or above (the particles are not supported ▼25
   25▲ otherwise)
  v = simGetInt32Parameter(sim_intparam_program_version)
  if (v < 20413) then
    simDisplayDialog('Warning', 'The propeller model is only fully supported from V ▼28
    28▲ -REP version 2.4.13 and above.&&nThis simulation will not run as expected!' ▼28
    28▲ , sim_dlgstyle_ok, false, '', nil, {0.8, 0, 0, 0, 0, 0})
  end

  -- --[[ #S#
  -- Waypoints
  waypointsHandlesNames = {}
  waypointHandle = simGetObjectHandle('waypoint')
  for i = 1, 10, 1 do
    waypointsHandlesNames[i] = 'waypoint0' .. i - 1
  end
  waypointsHandles = {}
  -- Asign waypoint handles to array
  for i, link in ipairs(waypointsHandlesNames) do
    waypointsHandles[link] = simGetObjectHandle(link)
  end
  -- #E# --]]
  -- Waypoints and control initialization
  -- Waypoints 01,05,04,03,02
  waypointsloopA = {1, 5, 4, 3, 2, 1}
  -- Waypoints 01,02,03,04,05
  -- waypointsloopA = {1, 2, 3, 4, 5}
  -- Waypoints 01,02,03,04,05,01
```

```
  -- waypointsloopA = {1, 2, 3, 4, 5, 1}
  -- Correct index -- Waypoint names start with 00, Lua has initial index at 1
  for i in ipairs(waypointsloopA) do
    waypointsloopA[i] = waypointsloopA[i] + 1
  end
  waypointcounter = 1

  -- Detatch the manipulation sphere:
  if settingUseTarget > 0 then
    targetObj = simGetObjectHandle('Quadricopter_target')
  else
    targetObj = simGetObjectHandle('waypoint')
  end
  simSetObjectParent(targetObj, -1, true)

  -- This control algo was quickly written and is dirty and not optimal. It just  ▼65
    65▲ serves as a SIMPLE example

  d = simGetObjectHandle('Quadricopter_base')

  if settingAnimatePropellers > 0 then
    particlesAreVisible = simGetScriptSimulationParameter(sim_handle_self, ' ▼70
    70▲ particlesAreVisible')
    simSetScriptSimulationParameter(sim_handle_tree, 'particlesAreVisible',  ▼71
    71▲ tostring(particlesAreVisible))
    simulateParticles = simGetScriptSimulationParameter(sim_handle_self, ' ▼72
    72▲ simulateParticles')
    simSetScriptSimulationParameter(sim_handle_tree, 'simulateParticles', tostring( ▼73
    73▲ simulateParticles))
  end

  propellerScripts = {-1, -1, -1, -1}
  for i = 1, 4, 1 do
    propellerScripts[i] = simGetScriptHandle('Quadricopter_propeller_respondable'  ▼78
    78▲ .. i)
  end
  heli = simGetObjectAssociatedWithScript(sim_handle_self)

  particlesTargetVelocities = {0, 0, 0, 0}

  pParam = 2
  iParam = 0
  dParam = 0
  vParam = -2

  cumul = 0
  lastE = 0
  pAlphaE = 0
  pBetaE = 0
  psp2 = 0
  psp1 = 0

  prevEuler = 0
```

```
  if settingFakeShadow > 0 then
    fakeShadow = simGetScriptSimulationParameter(sim_handle_self, 'fakeShadow')
    if (fakeShadow) then
      shadowCont = simAddDrawingObject(sim_drawing_discpoints + sim_drawing_cyclic ▼102
    102▲  + sim_drawing_25percenttransparency + sim_drawing_50percenttransparency  ▼102
    102▲ + sim_drawing_itemsizes, 0.2, 0, -1, 1)
    end
  end

  if settingUseInternalCameras > 0 then
    -- Prepare 2 floating views with the camera views:
    floorCam = simGetObjectHandle('Quadricopter_floorCamera')
    frontCam = simGetObjectHandle('Quadricopter_frontCamera')
    floorView = simFloatingViewAdd(0.9, 0.9, 0.2, 0.2, 0)
    frontView = simFloatingViewAdd(0.7, 0.9, 0.2, 0.2, 0)
    simAdjustView(floorView, floorCam, 64)
    simAdjustView(frontView, frontCam, 64)
  end
end

if (sim_call_type == sim_childscriptcall_cleanup) then
  if settingFakeShadow > 0 then
    simRemoveDrawingObject(shadowCont)
  end
  if settingUseInternalCameras > 0 then
    simFloatingViewRemove(floorView)
    simFloatingViewRemove(frontView)
  end
end

if (sim_call_type == sim_childscriptcall_actuation) then
  -- Acquire positions
  targetPos = simGetObjectPosition(targetObj, -1)
  pos = simGetObjectPosition(d, -1)
  sp = simGetObjectPosition(targetObj, d)
 m = simGetObjectMatrix(d, -1)

  if settingUseTarget == 0 then
    -- Control
    if math.abs(math.sqrt((targetPos[1] - pos[1])^2 + (targetPos[2] - pos[2]))) <= ▼136
    136▲  0.5 then
      -- Waypoint reached
      waypointcounter = waypointcounter + 1
    end
    -- Assign waypoint from loop
    waypointHandle = waypointsHandles[waypointsHandlesNames[waypointsloopA[ ▼141
    141▲ waypointcounter]]]
  end

  s = simGetObjectSizeFactor(d)

```

```
  pos = simGetObjectPosition(d, -1)
  if settingFakeShadow > 0 and (fakeShadow) then
    itemData = {pos[1], pos[2], 0.002, 0, 0, 1, 0.2 * s}
    simAddDrawingObjectItem(shadowCont, itemData)
  end

  -- Vertical control:
  l = simGetVelocity(heli)
  e = (targetPos[3] - pos[3])
  cumul = cumul + e
  pv = pParam * e
  thrust = 5.335 + pv + iParam * cumul + dParam * (e - lastE) + l[3] * vParam
  lastE = e

  -- Horizontal control:
  vx = {1, 0, 0}
  vx = simMultiplyVector(m, vx)
  vy = {0, 1, 0}
  vy = simMultiplyVector(m, vy)
  alphaE = (vy[3] - m[12])
  alphaCorr = 0.25 * alphaE + 2.1 * (alphaE - pAlphaE)
  betaE = (vx[3] - m[12])
  betaCorr = -0.25 * betaE - 2.1 * (betaE - pBetaE)
  pAlphaE = alphaE
  pBetaE = betaE
  alphaCorr = alphaCorr + sp[2] * 0.005 + 1 * (sp[2] - psp2)
  betaCorr = betaCorr - sp[1] * 0.005 - 1 * (sp[1] - psp1)
  psp2 = sp[2]
  psp1 = sp[1]

  -- Rotational control:
  euler = simGetObjectOrientation(d, targetObj)
  rotCorr = euler[3] * 0.1 + 2 * (euler[3] - prevEuler)
  prevEuler = euler[3]

  -- Decide of the motor velocities:
  particlesTargetVelocities[1] = thrust*(1 - alphaCorr + betaCorr + rotCorr)
  particlesTargetVelocities[2] = thrust*(1 - alphaCorr - betaCorr - rotCorr)
  particlesTargetVelocities[3] = thrust*(1 + alphaCorr - betaCorr + rotCorr)
  particlesTargetVelocities[4] = thrust*(1 + alphaCorr + betaCorr - rotCorr)

  -- Send the desired motor velocities to the 4 rotors:
  for i = 1, 4, 1 do
    simSetScriptSimulationParameter(propellerScripts[i], 'particleVelocity',  ▼189
    189▲ particlesTargetVelocities[i])
  end
end
```

Script 10: Lua script file: ugv-01–vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼22
    22▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4  ▼40
    40▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 =  ▼41
    41▲ pointCloud2 messages;
  -- Counters
  counter = 0

  -- Handles
  -- Frames
  worldHandle = simGetObjectHandle('frame00')
  -- vlp16Handle = simGetObjectHandle('vlp16')
  vlp16Handle = simGetObjectHandle('vpl16')
```

```
50
  -- Tranform (TF)
  tfLinks = {}
  -- for i,link in ipairs {'vlp16'} do
  for i,link in ipairs {'vpl16'} do
55  tfLinks[link] = simGetObjectHandle(link)
  end

  -- Base TF
  tf = {
60  header = {
      stamp = 0,
      frame_id = 'frame00'
    },
    child_frame_id = '',
65  transform = {
      -- ROS has definition x = front y = side z = up
      translation = {x = 0, y = 0, z = 0}, -- V-rep
      rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
    }
70 }


  -- VLP16
  pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
75 visionSensorHandles = {}
  for i = 1, 4, 1 do
    visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
  end

80 -- Setting VLP16 parameters
  frequency = 5 -- Default model 5 Hz
  -- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only  ▼82
    82▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼82
    82▲  (8)=displayed points are emissive
  options = 1 -- No display
  pointSize = 2
85 coloringCloseAndFarDistance = {1, 4}
  displayScaling = 0.999 -- so that points do not appear to disappear in objects

  -- Reading VLP16 sensor data
  vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles,  ▼89
    89▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼89
    89▲  pointCloudHandle)
90


  -- Init publishers and subscribers
  -- Init publishers
-- #S#
95 -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1,  ▼95
    95▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
  -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
  -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼97
```

```
    97▲ PointCloud2')
  -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼98
    98▲ VelodyneScan')
  -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼99
    99▲ VelodynePacket')
100 -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼100
    100▲ VelodyneScan')
-- #E#
  vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2')
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼103
    103▲ VelodynePacket')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
105 simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
end

-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
110 -- Retrieve sensor data
  data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼111
    111▲ simGetSimulationTimeStep())
  -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

115 -- Debug options
  -- print('----------------')
  -- print(bitand(debugOutput, 1))
  -- print(bitand(debugOutput, 2))
  -- print(bitand(debugOutput, 4))
120 -- print('----------------')

  -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
125   simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
    else
      pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
    end
  end
130

  -- Construct the point cloud from raw data: x, y and z coordinates for every ▼131
    131▲ point
  -- Reference: From the original VPL_16 model script
  local pointCloudData = {}
  for i = 0, #data / 3 - 1, 1 do
135   -- Temporary assignment to a vector d
    d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
    -- Transformation to sensor frame
    -- No transformation results in point cloud rotated by 90 deg around z
    d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
140 data[3 * i + 1] = d[1]
    data[3 * i + 2] = d[2]
    data[3 * i + 3] = d[3]
    table.insert(pointCloudData, d)
```

```
    -- print(d)
  end

  -- Display the detected points
  if displayPointCloud == 1 then
    simInsertPointsIntoPointCloud(pointCloud, 0, data)
  end

  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402,  ▼160
    160▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
-- #S#
      -- table.insert(tpcl, {1, 1, 1})
      -- table.insert(tpcl, {1.0, 1.0, 1.0})
      -- table.insert(tpcl, {2, 2, 2})
-- #E#
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackFloats( ▼185
    185▲ tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼186
    186▲ simPackUInts(tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼187
    187▲ simPackUInts(data)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼188
    188▲ simPackUInts(tdata)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼189
    189▲ simPackUInts(pointCloudData)} -- CRASH
-- #S#
```

```
-- #E#

    if bitand(debugOutput, 2) == 2 then
      print('-*- VLP16 Raw START')
      print('tpcl')
      print(#tpcl)
      print('tdata')
      print(#tdata)
-- #S#
      -- print('h')
      -- print(type(h))
      -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua] SCRIPT ▼202
    202▲ velodyneVPL...]:140: attempt to get length of global 'h' (a number value ▼202
    202▲ )
      -- print(type(tpcl))
      -- print(type(simPackUInts(tpcl)))
      -- print(type(simPackFloats(tpcl)))
      -- print(#simPackUInts(tpcl))
      -- print(type(tdata))
      -- print(tdata)
      -- for i, j in pairs(tdata) do
        -- print(i, j)
      -- end
      -- for i, j in pairs(data) do
        -- print(i, j)
      -- end
      -- print(#simPackUInts(tdata)) -- nil
      -- print('raw')
      -- print(type(pointCloudRawData.data))
      -- print(#pointCloudRawData.data)
      -- print('-*- VLP16 Raw END')
-- #E#
    end
    simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
    rmcounter = rmcounter + 1
  end
  if bitand(debugOutput, 2) == 2 then
    print('rmcounter')
    print(rmcounter)
    print('-*- VLP16 Raw END')
-- #S#
-- #E#
  end

  -- Publish point cloud

  -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
  -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼236
    236▲ Software/V-REP/robot.lua
  -- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
  local pointCloudData = {}
  --[[
  -- The frame of VLP16 shifts all the points to that frame
```

```
  -- If frame_id set to sensor frame in current configuration results in point  ▼241
    241▲ cloud shifted to the sensor position
  pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼242
    242▲ frame_id = "vlp16"}
  pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼243
    243▲ frame_id = "vpl16"}
  --]]
  pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼245
    245▲ frame_id = "frame00"}
  -- Describes the channels and their layout in the binary data blob
  local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
  local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
  local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
  pointCloudData['fields'] = {field_x, field_y, field_z}
  pointCloudData['is_bigendian'] = false
  pointCloudData['is_dense'] = true
  pointCloudData['data'] = simPackFloats(data) --  Works
  -- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the  ▼254
    254▲ data field being of type uint8
  -- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼255
    255▲ and it differs in number of received points
  --[[
  -- Theoretically this should work, but the bytes offset still drop the messages
  pointCloudData['height'] = 16 -- Ordered, 16 scan lines
  pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData. ▼259
    259▲ height)
  pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼260
    260▲ width / pointCloudData.height)
  --]]
  -- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
  pointCloudData['height'] = 1 -- Unordered
  -- Effectively number of cloud points, each having 3 coordinates
  pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
  -- Length of a point in bytes
  pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼267
    267▲ width)
  -- Length of a row in bytes
  pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼269
    269▲ point_step']

  if bitand(debugOutput, 4) == 1 then
    print('-- VLP16 PointCloud2 START')
    print(#pointCloudData.data)
    print(pointCloudData.width * pointCloudData.point_step)
    print(pointCloudData.point_step)
    print(#pointCloudData.data - math.floor(#pointCloudData.data / pointCloudData. ▼276
    276▲ width) * pointCloudData.width)
    print(#pointCloudData.data - pointCloudData.width * pointCloudData.point_step)
    print(pointCloudData.width)
-- #S#
    -- print(type(pointCloudData))
    -- print(type(tdata))
-- #E#
```

```
    print('-- VLP16 PointCloud2 END')
  end

  simExtRosInterface_publish(vlp16Pub, pointCloudData)

  -- Send transforms
  tf.header.stamp = simGetSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      position = simGetObjectPosition(handle, worldHandle)
      orientation = simGetObjectQuaternion(handle, worldHandle)
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = simGetObjectPosition(handle, vlp16Handle)
      orientation = simGetObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    simAddStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
```

```
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 11: Lua script file: ugv-03–robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 03 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
    10▲  format. UGV is mobile using forward and backward movements. The raw  ▼10
    10▲ message format is implemented but not used, since data formatting is not  ▼10
    10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis.
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  -- Counters
  counter = 0
  -- Prefixes
  rosParamPrefix = '/init/'
  posePrefix = 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼40
    40▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
-- TODO which one is it? Ros or RosInterface ? #D#
--    if (moduleName == 'RosInterface') then
    if (moduleName == 'Ros') then
      pluginNotFound = false
    end
    index = index + 1
```

```
  end

  if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼53
    53▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼53
    53▲ {0.5,0,0,1,1,1})
  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    object01Handle = simGetObjectHandle('object01') -- Cylinder
    object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼68
    68▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Robot 01
    -- Tranforms
    tfLinks = {}
    for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼75
    75▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

    -- Frames
    worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
    camera01Pub = simExtROS_enablePublisher('image01', 1, ▼84
    84▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    camera02Pub = simExtROS_enablePublisher('image02', 1, ▼85
    85▲ simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher('/joints/robot01/state', 1, ▼87
    87▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')

    -- Init subscribers
    -- robot01StatesSub = simExtROS_enableSubscriber('/joints/robot01/control', 1, ▼90
    90▲ simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

    -- Base TF
    tf = {
```

```
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

  if isint(counter / 100) then
    simAddStatusbarMessage('Counter: ' .. counter)
    -- if ((counter + 100) % 200 == 0) then
    if (counter % 200 == 0) then
```

```
      velocity = 2.0
    else
      velocity = -2.0
    end
    -- Get joint handles
    jointHandles = {}
    -- Pioneer_p3dx_leftMotor
    for i, m in ipairs{'left', 'right'} do
      -- Motor velocity for fixed wheels
      joint = 'Pioneer_p3dx_' .. m .. 'Motor'
      jointHandles[joint] = simGetObjectHandle(joint)
      simSetJointTargetVelocity(jointHandles[joint], velocity)
    end
    simAddStatusbarMessage('Velocity: ' .. velocity)
  end

  -- if isint(counter / 10) then
    -- simAddStatusbarMessage('Point cloud: ' .. ptCloud)
  -- end

  -- Update counter
  counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼174
    174▲ down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 12: Text file: ugv-01.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Text
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP scene - UGV 01 scenario

# General description
The UGV collects readings from VLP16 and publishes it in PointCloud2 message format ▼9
    9▲ . UGV is immobile. The raw message format is implemented but not used, since  ▼9
    9▲ data formatting is not working. An additional object was added to test the  ▼9
    9▲ filter.
```

Script 13: Text file: ugv-uav-01.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Text
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP scene - UGV UAV 01 scenario

# General description
The UGV collects readings from VLP16 and publishes it in PointCloud2 message format ▼9
    9▲ . UGV is mobile and follows waypoints if specified. UAV is currently immobile ▼9
    9▲ .

# Stereo vision parameters
Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. Both  ▼12
    12▲ camera frames are published in TF.
```

Script 14: Lua script file: ugv-05–vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 05 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- # @todo Fix VelodyneScan publishing -- The 'packets' field is not assigned ▼12
    12▲ properly despite proper assignment when VelodynePacket is used directly, ▼12
    12▲ probably needs conversion from Lua floats to uint8
-- # @done Implement switch for publishing pointcloud2 and raw messages
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼24
    24▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 ▼42
    42▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 = ▼43
    43▲ pointCloud2 messages;
  -- Publisher switch
  publishSwitch = 1 -- 0 = off; +1 = PointCloud2 (not implemented, on by default); ▼45
    45▲ +2 = raw messages scan (not working properly because of incorrect packets ▼45
    45▲ assignment); +4 = raw messages packet;
```

```
-- Counters
counter = 0

-- Handles
-- Frames
worldHandle = simGetObjectHandle('frame00')
robot01Handle = simGetObjectHandle('robot01')
-- vlp16Handle = simGetObjectHandle('vlp16')
vlp16Handle = simGetObjectHandle('vpl16')

-- Tranform (TF)
tfLinks = {}
-- for i,link in ipairs {'vlp16'} do
for i, link in ipairs {'vpl16'} do
  tfLinks[link] = simGetObjectHandle(link)
end

-- Base TF
tf = {
  header = {
    stamp = 0,
    frame_id = 'frame00'
  },
  child_frame_id = '',
  transform = {
    -- ROS has definition x = front y = side z = up
    translation = {x = 0, y = 0, z = 0}, -- V-rep
    rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
  }
}


-- VLP16
pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
visionSensorHandles = {}
for i = 1, 4, 1 do
  visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
end

-- Setting VLP16 parameters
frequency = 10 -- Default model 5 Hz
-- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only ▼87
 87▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼87
 87▲  (8)=displayed points are emissive
options = 1 -- No display
pointSize = 2
coloringCloseAndFarDistance = {1, 4}
displayScaling = 0.999 -- so that points do not appear to disappear in objects

-- Reading VLP16 sensor data
vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles, ▼94
 94▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼94
 94▲  pointCloudHandle)
```

```
95
    -- Init publishers and subscribers
    -- Init publishers
-- #S#
100 -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1, ▼100
      100▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
    -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
    -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼102
      102▲ PointCloud2')
    -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼103
      103▲ VelodyneScan')
    -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼104
      104▲ VelodynePacket')
105 -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼105
      105▲ VelodyneScan')
-- #E#
    if bitand(publishSwitch, 1) == 1 then
      vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2 ▼108
      108▲ ')
      simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
110 end
--  --[[ #DS#
    if bitand(publishSwitch, 2) == 2 then
      vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼113
      113▲ VelodyneScan')
      -- simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
115 elseif bitand(publishSwitch, 4) == 4 then
      vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼116
      116▲ VelodynePacket')
      simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
    end
-- #DE# --]]
120 end

-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
    -- Retrieve sensor data
125 data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼125
      125▲ simGetSimulationTimeStep())
    -- Retrieve sensor transformation matrix
    vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

    -- Debug options
130 -- print('----------------')
    -- print(bitand(debugOutput, 1))
    -- print(bitand(debugOutput, 2))
    -- print(bitand(debugOutput, 4))
    -- print('----------------')
135
    -- Display the detected points
    if displayPointCloud == 1 then
      if pointCloud then
```

```
    simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
  else
    pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
  end
end

-- Construct the point cloud from raw data: x, y and z coordinates for every  ▼145
 145▲ point
-- Reference: From the original VPL_16 model script
local pointCloudData = {}
for i = 0, #data / 3 - 1, 1 do
  -- Temporary assignment to a vector d
  d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
  -- Transformation to sensor frame
  -- No transformation results in point cloud rotated by 90 deg around z
  d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
  data[3 * i + 1] = d[1]
  data[3 * i + 2] = d[2]
  data[3 * i + 3] = d[3]
  table.insert(pointCloudData, d)
  -- print(d)
end

-- Display the detected points
if displayPointCloud == 1 then
  simInsertPointsIntoPointCloud(pointCloud, 0, data)
end

if bitand(publishSwitch, 2) == 2 or bitand(publishSwitch, 4) == 4 then
-- --[[ #DS# -- Comment this line for raw message handling, currently only a  ▼167
 167▲ dummy message publishing implemented
  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402,  ▼176
 176▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
```

```
    -- for i = 1, 402, 1 do
190     for i = 1, 1, 1 do -- works forcing one entry in table
          table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
  -- #S#
        -- table.insert(tpcl, {1, 1, 1})
        -- table.insert(tpcl, {1.0, 1.0, 1.0})
195     -- table.insert(tpcl, {2, 2, 2})
  -- #E#
      end

      local pointCloudRawData = {}
200   -- CRASH with multiple objects in data
      if bitand(publishSwitch, 2) == 2 then
        pointCloudRawData['header'] = {seq = 0, stamp = simGetSimulationTimeStep() ▼202
  202▲ , frame_id = "frame00"}
        -- Apparently neither indirect nor direct assignment of packets object  ▼203
  203▲ works
        pointCloudRawData['packets']  = {stamp = simGetSimulationTimeStep(), data  ▼204
  204▲ = simPackFloats(tpcl)}
205     -- Lua runtime error: [string "[embScript_51284011.lua] SCRIPT  ▼205
  205▲ velodyneVPL_16"]:253: read__velodyne_msgs__VelodyneScan: field packets:  ▼205
  205▲ expected array (simExtRosInterface_publish @ 'RosInterface' plugin)
  -- #S#
        -- pointCloudRawData['packets']  = {stamp = simGetSimulationTimeStep(),  ▼207
  207▲ data = simPackUInts(tpcl)}
        -- local pointCloudRawDataPackets = {}
        -- pointCloudRawDataPackets = {stamp = simGetSimulationTimeStep(), data =  ▼209
  209▲ simPackFloats(tpcl)}
210     -- pointCloudRawData['packets'] = pointCloudRawDataPackets
        -- Equivalent to above
        -- pointCloudRawData = { packets  = {
          -- stamp = simGetSimulationTimeStep(), data = simPackFloats(tpcl)
          -- }
215     -- }
  -- #E#
      elseif bitand(publishSwitch, 4) == 4 then
        pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼218
  218▲ simPackFloats(tpcl)}
      end
220 -- #S#
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼221
  221▲ simPackUInts(tpcl)}
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼222
  222▲ simPackUInts(data)} -- CRASH
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼223
  223▲ simPackUInts(tdata)} -- CRASH
      -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼224
  224▲ simPackUInts(pointCloudData)} -- CRASH
225 -- #E#

      if bitand(debugOutput, 2) == 2 then
        print('-*- VLP16 Raw START')
        print('tpcl')
```

```
        print(#tpcl)
        print('tdata')
        print(#tdata)
    -- #S#
        -- print('h')
        -- print(type(h))
        -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua]  ▼236
    236▲ SCRIPT velodyneVPL...":140: attempt to get length of global 'h' (a  ▼236
    236▲ number value)
        -- print(type(tpcl))
        -- print(type(simPackUInts(tpcl)))
        -- print(type(simPackFloats(tpcl)))
        -- print(#simPackUInts(tpcl))
        -- print(type(tdata))
        -- print(tdata)
        -- for i, j in pairs (tdata) do
          -- print(i, j)
        -- end
        -- for i, j in pairs (data) do
          -- print(i, j)
        -- end
        -- print(#simPackUInts(tdata)) -- nil
        -- print('raw')
        -- print(type(pointCloudRawData.data))
        -- print(#pointCloudRawData.data)
        -- print('-*- VLP16 Raw END')
    -- #E#
      end
      simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
      rmcounter = rmcounter + 1
    end
    if bitand(debugOutput, 2) == 2 then
      print('rmcounter')
      print(rmcounter)
      print('-*- VLP16 Raw END')
    end
    -- #S#
    -- #E#
  end
  -- #DE# --]]

  if bitand(publishSwitch, 1) == 1 then
    -- Publish point cloud

    -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
    -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/  ▼273
    273▲ Software/V-REP/robot.lua
    -- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
    local pointCloudData = {}
    --[[
    -- The frame of VLP16 shifts all the points to that frame
    -- If frame_id set to sensor frame in current configuration results in point  ▼278
    278▲ cloud shifted to the sensor position
```

```
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼279
    279▲ frame_id = "vlp16"}
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼280
    280▲ frame_id = "vpl16"}
    --]]
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼282
    282▲ frame_id = "frame00"}
    -- Describes the channels and their layout in the binary data blob
    local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
    local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
    local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
    pointCloudData['fields'] = {field_x, field_y, field_z}
    pointCloudData['is_bigendian'] = false
    pointCloudData['is_dense'] = true
    pointCloudData['data'] = simPackFloats(data) --  Works
    -- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the  ▼291
    291▲ data field being of type uint8
    -- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼292
    292▲ and it differs in number of received points
    --[[
    -- Theoretically this should work, but the bytes offset still drop the  ▼294
    294▲ messages
    pointCloudData['height'] = 16 -- Ordered, 16 scan lines
    pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData ▼296
    296▲ .height)
    pointCloudData['point_step'] = math.floor(#pointCloudData.data /  ▼297
    297▲ pointCloudData.width / pointCloudData.height)
    --]]
    -- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
    pointCloudData['height'] = 1 -- Unordered
    -- Effectively number of cloud points, each having 3 coordinates
    pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
    -- Length of a point in bytes
    pointCloudData['point_step'] = math.floor(#pointCloudData.data /  ▼304
    304▲ pointCloudData.width)
    -- Length of a row in bytes
    pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼306
    306▲ point_step']

    if bitand(debugOutput, 4) == 1 then
      print('-- VLP16 PointCloud2 START')
      print(#pointCloudData.data)
      print(pointCloudData.width * pointCloudData.point_step)
      print(pointCloudData.point_step)
      print(#pointCloudData.data - math.floor(#pointCloudData.data /  ▼313
    313▲ pointCloudData.width) * pointCloudData.width)
      print(#pointCloudData.data - pointCloudData.width * pointCloudData. ▼314
    314▲ point_step)
      print(pointCloudData.width)
  -- #S#
      -- print(type(pointCloudData))
      -- print(type(tdata))
  -- #E#
```

```
320       print('-- VLP16 PointCloud2 END')
        end

        simExtRosInterface_publish(vlp16Pub, pointCloudData)
      end
325
      -- --[[
      -- Send transforms
      tf.header.stamp = simGetSimulationTimeStep()
      for link, handle in pairs(tfLinks) do
330     -- Get parent
        -- if (link == 'vlp16') then
        if (link == 'vpl16') then
          tf.header.frame_id = 'frame00' -- Absolute frame reference
          -- tf.header.frame_id = 'robot01' -- Relative to robot frame reference
335       -- Get pose relative to worldframe
          position = simGetObjectPosition(handle, worldHandle) -- Absolute frame ▼336
        336▲ reference
          orientation = simGetObjectQuaternion(handle, worldHandle) -- Absolute frame ▼337
        337▲ reference
          -- position = simGetObjectPosition(handle, robot01Handle) -- Relative to ▼338
        338▲ robot frame reference
          -- orientation = simGetObjectQuaternion(handle, robot01Handle) -- Relative ▼339
        339▲ to robot frame reference
340     else
          -- tf.header.frame_id = 'vlp16'
          tf.header.frame_id = 'vpl16'
          -- Get pose relative to sensor
          position = simGetObjectPosition(handle, vlp16Handle)
345       orientation = simGetObjectQuaternion(handle, vlp16Handle)
        end

        -- Update TF
        tf.child_frame_id = link
350     tf.transform.translation.x = position[1]
        tf.transform.translation.y = position[2]
        tf.transform.translation.z = position[3]
        tf.transform.rotation.x = orientation[1]
        tf.transform.rotation.y = orientation[2]
355     tf.transform.rotation.z = orientation[3]
        tf.transform.rotation.w = orientation[4]
        simExtRosInterface_sendTransform(tf)
      end
      --]]
360
      -- Update counter
      counter = counter + 1

      if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
365     simAddStatusbarMessage('VLP16 counter: ' .. counter)
        print('--- VLP16 START')
        print('pcl data')
        print(#pointCloudData)
```

```
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 15: Lua script file: ugv-05–scripts–threaded-move-waypoint.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 05 -- Scripts
-- # Waypoint pose threaded script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
   10▲ format. UGV is mobile using forward and backward movements. The raw ▼10
   10▲ message format is implemented but not used, since data formatting is not ▼10
   10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. ▼11
   11▲ Both camera frames are published in TF.
-- # * Threaded script is necessary for simMoveToObject function, which is now ▼12
   12▲ unused, therefore the scripts can be changed to non-threaded child or ▼12
   12▲ implemented back into robot script
-- ###


threadFunction=function()

  while simGetSimulationState() ~= sim_simulation_advancing_abouttostop do
    -- Step for waypoint switch
    if isint(counter / 2) then
      -- Using simMoveToObject should work in both cases below, but it does leave ▼21
    21▲ the position at zero
      -- Probably the velocity and acceleration parameters should be changed
      -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[' ▼23
    23▲ waypoint0' .. waypointcounter], 3, 1, 1, nil)
      -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[ ▼24
    24▲ waypointcounter], 3, 1, 0, nil)

      -- Since the approach does not work properly, direct position and orientation ▼26
    26▲ assignment is used
      -- Get current waypoint position and orientation
      position = simGetObjectPosition(waypointsHandles['waypoint0' .. ▼28
    28▲ waypointcounter], -1)
      eulerAngles = simGetObjectOrientation(waypointsHandles['waypoint0' .. ▼29
    29▲ waypointcounter], -1)
      -- Set marker to current waypoint
      position = simSetObjectPosition(waypointHandle, -1, position)
      eulerAngles = simSetObjectOrientation(waypointHandle, -1, eulerAngles)

      -- Increase waypoint counter
      waypointcounter = waypointcounter + 1
      -- Reset waypoint counter condition
      if (waypointcounter == 10) then
        waypointcounter = 0
      end
    end
```

```
    position = simGetObjectPosition(waypointHandle, -1)
    eulerAngles = simGetObjectOrientation(waypointHandle, -1)
    -- Print verbose information
    -- print('Loop: ' .. counter .. ', waypoint0' .. waypointcounter .. ', pos = '
    .. position[1] .. ', ' .. position[2] .. ', ' .. position[3] .. ', or = '
    .. eulerAngles[1] .. ', ' .. eulerAngles[2] .. ', ' .. eulerAngles[3])

    counter = counter + 1
    simSwitchThread()
  end

end

-- Put some initialization code here:
simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching

-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- Frames
worldHandle = simGetObjectHandle('frame00')

-- Waypoints
waypointsHandlesNames = {}
waypointHandle = simGetObjectHandle('waypoint')

for i = 1, 10, 1 do
  waypointsHandlesNames[i] = 'waypoint0' .. i - 1
end

waypointsHandles = {}
-- Asign waypoint handles to array
for i, link in ipairs(waypointsHandlesNames) do
  waypointsHandles[link] = simGetObjectHandle(link)
end

-- waypointsPub = simExtROS_enablePublisher('/vrep/waypoints', 1,
    simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')
waypointsPub = simExtROS_enablePublisher('/vrep/waypoints', 1,
    simros_strmcmd_get_object_pose, waypointHandle, -1, 'frame00')

counter = 1

waypointcounter = 0

-- Here we execute the regular thread code:
res, err = xpcall(threadFunction, function(err) return debug.traceback(err) end)

if not res then
```

```
90    simAddStatusbarMessage('Lua runtime error: ' .. err)
end

-- Put some clean-up code here:
```

Script 16: Lua script file: ugv-uav-01–scripts.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 04 -- Scripts
-- # Waypoint pose threaded script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
   10▲ format. UGV is mobile using forward and backward movements. The raw ▼10
   10▲ message format is implemented but not used, since data formatting is not ▼10
   10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. ▼11
   11▲ Both camera frames are published in TF.
-- # * Threaded script is necessary for simMoveToObject function, which is now ▼12
   12▲ unused, therefore the scripts can be changed to non-threaded child or ▼12
   12▲ implemented back into robot script
-- ###


threadFunction=function()

  while simGetSimulationState() ~= sim_simulation_advancing_abouttostop do
    -- Step for waypoint switch
    if isint(counter / 2) then
      -- Using simMoveToObject should work in both cases below, but it does leave ▼21
    21▲ the position at zero
      -- Probably the velocity and acceleration parameters should be changed
      -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[' ▼23
    23▲ waypoint0' .. waypointcounter], 3, 1, 1, nil)
      -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[ ▼24
    24▲ waypointcounter], 3, 1, 0, nil)

      -- Since the approach does not work properly, direct position and orientation ▼26
    26▲ assignment is used
      -- Get current waypoint position and orientation
      position = simGetObjectPosition(waypointsHandles['waypoint0' .. ▼28
    28▲ waypointcounter], -1)
      eulerAngles = simGetObjectOrientation(waypointsHandles['waypoint0' .. ▼29
    29▲ waypointcounter], -1)
      -- Set marker to current waypoint
      position = simSetObjectPosition(waypointHandle, -1, position)
      eulerAngles = simSetObjectOrientation(waypointHandle, -1, eulerAngles)

      -- Increase waypoint counter
      waypointcounter = waypointcounter + 1
      -- Reset waypoint counter condition
      if (waypointcounter == 10) then
        waypointcounter = 0
      end
    end
```

```
    position = simGetObjectPosition(waypointHandle, -1)
    eulerAngles = simGetObjectOrientation(waypointHandle, -1)
    -- Print verbose information
    -- print('Loop: ' .. counter .. ', waypoint0' .. waypointcounter .. ', pos = '
    .. position[1] .. ', ' .. position[2] .. ', ' .. position[3] .. ', or = '
    .. eulerAngles[1] .. ', ' .. eulerAngles[2] .. ', ' .. eulerAngles[3])

    counter = counter + 1
    simSwitchThread()
  end

end

-- Put some initialization code here:
simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching

-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- Frames
worldHandle = simGetObjectHandle('frame00')

-- Waypoints
waypointsHandlesNames = {}
waypointHandle = simGetObjectHandle('waypoint')

for i = 1, 10, 1 do
  waypointsHandlesNames[i] = 'waypoint0' .. i - 1
end

waypointsHandles = {}
-- Asign waypoint handles to array
for i, link in ipairs(waypointsHandlesNames) do
  waypointsHandles[link] = simGetObjectHandle(link)
end

-- waypointsPub = simExtROS_enablePublisher('/vrep/waypoints', 1,
    simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')
waypointsPub = simExtROS_enablePublisher('/vrep/waypoints', 1,
    simros_strmcmd_get_object_pose, waypointHandle, -1, 'frame00')

counter = 1

waypointcounter = 0

-- Here we execute the regular thread code:
res, err = xpcall(threadFunction, function(err) return debug.traceback(err) end)

if not res then
```

```
90    simAddStatusbarMessage('Lua runtime error: ' .. err)
end

-- Put some clean-up code here:
```

Script 17: Lua script file: ugv-uav-01–robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 07 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
    10▲ format. UGV is mobile using forward and backward movements. The raw ▼10
    10▲ message format is implemented but not used, since data formatting is not ▼10
    10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. ▼11
    11▲ Both camera frames are published in TF.
-- # * Robot travels along several waypoints in a loop using a simple controller
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  stopSimulation = 0 -- 0 = off; >0 = on, number of full cycles type A; <0 = on, ▼29
    29▲ number of full cycles type B (not implemented yet);
  counterSimulation = 0 -- Simulation counter offset
  followWaypoints = 0 -- 1 = on; 0 = off;
  -- counterStep = 10 -- Internal script handling step
  counterStep = 4 -- Internal script handling step
  -- Counters
  counter = 0
  -- Prefixes
  vrepPrefix = '/vrep/'
  rosParamPrefix = vrepPrefix .. 'init/'
  posePrefix = vrepPrefix .. 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼47
```

```
    47▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
--    if (moduleName == 'RosInterface') then -- Use this for V-REP > 3.3.2
    if (moduleName == 'Ros') then -- This works with V-REP == 3.3.2
      pluginNotFound = false
    end
    index = index + 1
  end

  if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼59
    59▲ run properly', sim_dlgstyle_ok, false, nil, {0.8, 0, 0, 0, 0, 0}, {0.5, 0,  ▼59
    59▲ 0, 1, 1, 1})
  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    stereoVisionHandle = simGetObjectHandle('stereo_vision') -- Stereo vision  ▼65
    65▲ system
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    -- object01Handle = simGetObjectHandle('object01') -- Cylinder
    -- object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- --[[ #S#
    -- Waypoints
    waypointsHandlesNames = {}
    waypointHandle = simGetObjectHandle('waypoint')
    for i = 1, 10, 1 do
      waypointsHandlesNames[i] = 'waypoint0' .. i - 1
    end
    waypointsHandles = {}
    -- Asign waypoint handles to array
    for i, link in ipairs(waypointsHandlesNames) do
      waypointsHandles[link] = simGetObjectHandle(link)
    end
    -- #E# --]]

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼89
    89▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Initial and gain velocity
    velocity = 10

```

```
    -- Robot 01 + stereo vision
    -- Tranforms
    tfLinks = {}
    -- Publish with rotated camera frames
100 for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼100
    100▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼100
    100▲ camera01', 'camera02', 'camera01r', 'camera02r'} do
    -- Publish without rotated camera frames
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼102
    102▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼102
    102▲ camera01', 'camera02'} do
    -- Publish also the VPL16 sensor
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼104
    104▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼104
    104▲ camera01', 'camera02', 'vpl16'} do
105 -- Alternative to publishing poses
    -- for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼106
    106▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link', 'stereo_vision', ' ▼106
    106▲ camera01', 'camera02', 'waypoint00', 'waypoint01', 'waypoint02', ' ▼106
    106▲ waypoint03', 'waypoint04', 'waypoint05', 'waypoint06', 'waypoint07', ' ▼106
    106▲ waypoint08', 'waypoint09'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

110 -- Frames
    worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
115 camera01Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera01' ▼115
    115▲ , 1, simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    camera02Pub = simExtROS_enablePublisher(vrepPrefix .. 'stereo_vision/camera02' ▼116
    116▲ , 1, simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher(vrepPrefix .. 'joints/robot01/ ▼118
    118▲ state', 1, simros_strmcmd_get_joint_state, sim_handle_all, -1, '')
    -- Publish robot01 pose
120 -- Argument -1 makes values absolute, could be also frame00 as world frame ▼120
    120▲ reference
    -- robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼121
    121▲ simros_strmcmd_get_object_pose, robot01Handle, -1, '')
    robot01posePub = simExtROS_enablePublisher(posePrefix .. 'robot01', 1, ▼122
    122▲ simros_strmcmd_get_object_pose, robot01Handle, worldHandle, 'frame00')
    -- Publish waypoints poses
    -- waypointsPub = simExtROS_enablePublisher(vrepPrefix .. 'waypoints', 1, ▼124
    124▲ simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')
125


    -- Init subscribers
    robot01StatesSub = simExtROS_enableSubscriber(vrepPrefix .. 'joints/robot01/ ▼128
    128▲ control', 1, simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

130 -- Simulation time
```

```
    -- clockPub = simExtROS_enablePublisher('/clock', 1, 'rosgraph_msgs/Clock')
    clockPub = simExtRosInterface_advertise('/clock', 'rosgraph_msgs/Clock')

    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

    -- Waypoints and control initialization
    -- Waypoints 01,05,04,03,02
    -- waypointsloopA = {1, 5, 4, 3, 2}
    -- Waypoints 01,02,03,04,05
    -- waypointsloopA = {1, 2, 3, 4, 5}
    -- Waypoints 01,02,03,04,05,01
    waypointsloopA = {1, 2, 3, 4, 5, 1}
    -- Correct index -- Waypoint names start with 00, Lua has initial index at 1
    for i in ipairs(waypointsloopA) do
      waypointsloopA[i] = waypointsloopA[i] + 1
    end
    waypointcounter = 1
    -- Tolerance position
    tp = 0.3
    -- Tolerance heading
    to = 15 / 360
    -- Control gains
    -- Control gain position
    kp = 8.0
    -- Control gain orientation
    ko = 16.0
    -- Wheel base
    L = 0.1655
    l = 2 * L
    -- Wheel radius
    r = 0.975
    -- r = 0.0975 -- Documentation
    -- Initial velocities
    v = 0
    w = 0

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then
```

```
185   -- Publish simulation time
      -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})
      simExtRosInterface_publish(clockPub, {clock = simGetSystemTime()})

      -- Send transforms
190   tf.header.stamp = simGetSimulationTime()
      for link, handle in pairs(tfLinks) do
        -- Get parent
        if (link == 'robot01') then
          tf.header.frame_id = 'frame00'
195       -- Get pose relative to world frame
          p = simGetObjectPosition(handle, worldHandle)
          o = simGetObjectQuaternion(handle, worldHandle)
        -- --[[ -- Uncomment this block comment for publishing all stereo vision  ▼198
       198▲ system handles relative to robot
        elseif string.find(link, 'camera') then
200       tf.header.frame_id = 'stereo_vision'
          -- Get pose relative to stereo vision system
          p = simGetObjectPosition(handle, stereoVisionHandle)
          o = simGetObjectQuaternion(handle, stereoVisionHandle)
        --]] -- End of block comment
205     else
          tf.header.frame_id = 'robot01'
          -- Get pose relative to robot
          p = simGetObjectPosition(handle, robot01Handle)
          o = simGetObjectQuaternion(handle, robot01Handle)
210     end

        -- Update TF
        tf.child_frame_id = link
        tf.transform.translation.x = p[1]
215     tf.transform.translation.y = p[2]
        tf.transform.translation.z = p[3]
        tf.transform.rotation.x = o[1]
        tf.transform.rotation.y = o[2]
        tf.transform.rotation.z = o[3]
220     tf.transform.rotation.w = o[4]
        simExtRosInterface_sendTransform(tf)
      end

      if followWaypoints == 1 then
225   -- --[[
        -- Get robot position and orientation
        -- rp = simGetObjectPosition(robot01Handle, robot01Handle)
        -- ro = simGetObjectQuaternion(robot01Handle, robot01Handle)
        rp = simGetObjectPosition(robot01Handle, worldHandle)
230     ro = simGetObjectQuaternion(robot01Handle, worldHandle)

        -- Assign waypoint from loop
        waypointHandle = waypointsHandles[ waypointsHandlesNames[ waypointsloopA[ ▼233
       233▲ waypointcounter] ] ]
        -- print(waypointsloopA[waypointcounter])
```

```
235
    -- Get waypoint position and orientation
    wp = simGetObjectPosition(waypointHandle, robot01Handle)
    wo = simGetObjectQuaternion(waypointHandle, robot01Handle)
    -- wp = simGetObjectPosition(waypointHandle, worldHandle)
240 -- wo = simGetObjectQuaternion(waypointHandle, worldHandle)

    -- Pose 2D
    -- Error position
    ep = math.sqrt(wp[1]^2 + wp[2]^2)
245 -- Error heading
    eh = math.fmod(math.atan2(wp[2], wp[1]) + math.pi, 2 * math.pi) - math.pi
    -- Error final orientation
    eo = wo[3] - ro[3]

250 -- Control
    if math.abs(ep) <= tp then
    -- if math.abs(eo) <= to and math.abs(ep) <= tp then
      -- Waypoint reached
      waypointcounter = waypointcounter + 1
255   -- Reset counter if overflow
      if waypointcounter > #waypointsloopA then
        waypointcounter = 1
        simAddStatusbarMessage('Full cycle achieved')
        counterSimulation = counterSimulation + 1
260   end
      if isint(counter / counterStep) then
        simAddStatusbarMessage('Waypoint reached. Moving to next waypoint: ' .. ▼262
262▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
      end
    elseif math.abs(ep) <= 3 * tp then
265   -- Increased gains near target
      if isint(counter / counterStep) then
        simAddStatusbarMessage('Using increased gains towards: ' .. ▼267
267▲ waypointsHandlesNames[waypointsloopA[waypointcounter]])
      end
      v = kp * 18 * (wp[1] - tp)
270   w = ko * 40 * eh
      vl = v / r - w * L / r
      vr = v / r + w * L / r
    else
      -- Normal control mode
275   v = kp * (wp[1] - tp)
      w = ko * eh
      vl = v / r - w * L / r
      vr = v / r + w * L / r
      if isint(counter / counterStep) then
280     simAddStatusbarMessage('Reaching waypoint: ' .. waypointsHandlesNames[ ▼280
280▲ waypointsloopA[waypointcounter]])
      end
    end

    -- Saturation
```

```
    velocities = {}
    -- for i, j in ipairs {'vl', 'vr'} do
      if vl > velocity then
        vl = velocity
      elseif vl < -velocity then
        vl = -velocity
      end
      velocities[1] = vl
      if vr > velocity then
        vr = velocity
      elseif vr < -velocity then
        vr = -velocity
      end
      velocities[2] = vr
    -- end

    if isint(counter / counterStep) then
      simAddStatusbarMessage('### Loop: ' .. counter .. ' ###')
      simAddStatusbarMessage('Robot: Position: X: ' .. rp[1] .. ', Y: ' .. rp[2]  ▼303
    303▲ .. ', Z: ' .. rp[3])
      simAddStatusbarMessage('Robot: Orientation: A: ' .. ro[1] .. ', B: ' .. ro ▼304
    304▲ [2] .. ', G: ' .. ro[3])
      simAddStatusbarMessage('Waypoint: Position: X: ' .. wp[1] .. ', Y: ' .. wp ▼305
    305▲ [2] .. ', Z: ' .. wp[3])
      simAddStatusbarMessage('Waypoint: Orientation: A: ' .. wo[1] .. ', B: ' ..  ▼306
    306▲ wo[2] .. ', G: ' .. wo[3])
      simAddStatusbarMessage('Velocity: V: ' .. v .. ', W: ' .. w)
      simAddStatusbarMessage('Velocity: L: ' .. vl .. ', R: ' .. vr)
      simAddStatusbarMessage('Errors: H: ' .. eh .. ', P: ' .. ep .. ' (X: ' .. wp ▼309
    309▲ [1] .. ', Y: ' .. wp[2] .. '), O: ' .. eo)
      simAddStatusbarMessage('Tolerance: H: ' .. to .. ', P: ' .. tp .. ', O: ' .. ▼310
    310▲  to)
    end

    -- Get joint handles
    jointHandles = {}
    -- Pioneer_p3dx_leftMotor
    for i, m in ipairs {'left', 'right'} do
      -- Motor velocity for fixed wheels
      joint = 'Pioneer_p3dx_' .. m .. 'Motor'
      jointHandles[joint] = simGetObjectHandle(joint)
      simSetJointTargetVelocity(jointHandles[joint], velocities[i])
      -- simAddStatusbarMessage('Velocity' .. i .. ': ' .. velocities[i])
    end
  --]]
  end

-- --[[
  -- Stop simulation condition
  if stopSimulation > 0 and counterSimulation >= stopSimulation then
      simAddStatusbarMessage('Stop simulation')
      simStopSimulation()
  end
```

```
--]]

  -- Update counter
  counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 18: Text file: ugv-04.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Text
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP scene - UGV 04 scenario

# General description
The UGV collects readings from VLP16 and publishes it in PointCloud2 message format ▼9
    9▲ . UGV is mobile using forward and backward movements, loop cycles number can  ▼9
    9▲ be set, default is 1. The raw message format is implemented but not used,  ▼9
    9▲ since data formatting is not working. An additional object was added to test  ▼9
    9▲ the filter.

# Stereo vision parameters
Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. Both  ▼12
    12▲ camera frames are published in TF.
```

Script 19: Text file: ugv-02.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Text
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP scene - UGV 02 scenario

# General description
The UGV collects readings from VLP16 and publishes it in PointCloud2 message format ▼9
    9▲ . UGV is mobile using forward and backward movements. The raw message format  ▼9
    9▲ is implemented but not used, since data formatting is not working.
```

Script 20: Lua script file: ugv-02–robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 02 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
    10▲  format. UGV is mobile using forward and backward movements. The raw  ▼10
    10▲ message format is implemented but not used, since data formatting is not  ▼10
    10▲ working.
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  -- Counters
  counter = 0
  -- Prefixes
  rosParamPrefix = '/init/'
  posePrefix = 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼39
    39▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
-- TODO which one is it? Ros or RosInterface ? #D#
--    if (moduleName == 'RosInterface') then
    if (moduleName == 'Ros') then
      pluginNotFound = false
    end
    index = index + 1
  end
```

```
50  if (pluginNotFound) then
      -- Display an error message if the plugin was not found:
      simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼52
     52▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼52
     52▲ {0.5,0,0,1,1,1})
    else

55    -- Handles
      -- Objects
      robot01Handle = simGetObjectHandle('robot01')
      camera01Handle = simGetObjectHandle('camera01') -- Left
      camera02Handle = simGetObjectHandle('camera02') -- Right
60    object01Handle = simGetObjectHandle('object01') -- Cylinder
      object02Handle = simGetObjectHandle('object02') -- Cuboid

      -- Robot specific
      -- Robot 01
65    -- Wheels
      robot01JointHandles = {}
      for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼67
     67▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
        robot01JointHandles[link] = simGetObjectHandle(link)
      end
70
      -- Robot 01
      -- Tranforms
      tfLinks = {}
      for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼74
     74▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link'} do
75      tfLinks[link] = simGetObjectHandle(link)
      end

      -- Frames
      worldHandle = simGetObjectHandle('frame00')
80
      -- Init publishers and subscribers
      -- Init publishers
      camera01Pub = simExtROS_enablePublisher('image01', 1, ▼83
     83▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
      camera02Pub = simExtROS_enablePublisher('image02', 1, ▼84
     84▲ simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
85    -- Robot 01
      robot01jointsPub = simExtROS_enablePublisher('/joints/robot01/state', 1, ▼86
     86▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')

      -- Init subscribers
      -- robot01StatesSub = simExtROS_enableSubscriber('/joints/robot01/control', 1, ▼89
     89▲ simros_strmcmd_set_joint_state, sim_handle_all, -1, '')
90
      -- Base TF
      tf = {
        header = {
```

```
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

  if isint(counter / 100) then
    simAddStatusbarMessage('Counter: ' .. counter)
    -- if ((counter + 100) % 200 == 0) then
    if (counter % 200 == 0) then
      velocity = 2.0
```

```
    else
      velocity = -2.0
    end
    -- Get joint handles
    jointHandles = {}
    -- Pioneer_p3dx_leftMotor
    for i, m in ipairs{'left', 'right'} do
      -- Motor velocity for fixed wheels
      joint = 'Pioneer_p3dx_' .. m .. 'Motor'
      jointHandles[joint] = simGetObjectHandle(joint)
      simSetJointTargetVelocity(jointHandles[joint], velocity)
    end
    simAddStatusbarMessage('Velocity: ' .. velocity)
  end

  -- if isint(counter / 10) then
    -- simAddStatusbarMessage('Point cloud: ' .. ptCloud)
  -- end

  -- Update counter
  counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼173
    173▲ down at simulation end):
    -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
  end
end
```

Script 21: Lua script file: ugv-uav-01–vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 04 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- # @todo Fix VelodyneScan publishing -- The 'packets' field is not assigned ▼12
    12▲ properly despite proper assignment when VelodynePacket is used directly, ▼12
    12▲ probably needs conversion from Lua floats to uint8
-- # @done Implement switch for publishing pointcloud2 and raw messages
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼24
    24▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 ▼42
    42▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 = ▼43
    43▲ pointCloud2 messages;
  -- Publisher switch
  publishSwitch = 1 -- 0 = off; +1 = PointCloud2 (not implemented, on by default); ▼45
    45▲ +2 = raw messages scan (not working properly because of incorrect packets ▼45
    45▲ assignment); +4 = raw messages packet;
```

```
-- Counters
counter = 0

-- Handles
-- Frames
worldHandle = simGetObjectHandle('frame00')
robot01Handle = simGetObjectHandle('robot01')
-- vlp16Handle = simGetObjectHandle('vlp16')
vlp16Handle = simGetObjectHandle('vpl16')

-- Tranform (TF)
tfLinks = {}
-- for i,link in ipairs {'vlp16'} do
for i, link in ipairs {'vpl16'} do
  tfLinks[link] = simGetObjectHandle(link)
end

-- Base TF
tf = {
  header = {
    stamp = 0,
    frame_id = 'frame00'
  },
  child_frame_id = '',
  transform = {
    -- ROS has definition x = front y = side z = up
    translation = {x = 0, y = 0, z = 0}, -- V-rep
    rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
  }
}


-- VLP16
pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
visionSensorHandles = {}
for i = 1, 4, 1 do
  visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
end

-- Setting VLP16 parameters
frequency = 10 -- Default model 5 Hz
-- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only ▼87
 87▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼87
 87▲  (8)=displayed points are emissive
options = 1 -- No display
pointSize = 2
coloringCloseAndFarDistance = {1, 4}
displayScaling = 0.999 -- so that points do not appear to disappear in objects

-- Reading VLP16 sensor data
vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles, ▼94
 94▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼94
 94▲  pointCloudHandle)
```

```
95
  -- Init publishers and subscribers
  -- Init publishers
-- #S#
100   -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1,  ▼100
        100▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
      -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
      -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼102
        102▲ PointCloud2')
      -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼103
        103▲ VelodyneScan')
      -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼104
        104▲ VelodynePacket')
105   -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼105
        105▲ VelodyneScan')
-- #E#
  if bitand(publishSwitch, 1) == 1 then
    vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2 ▼108
    108▲ ')
    simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
110 end
--  --[[ #DS#
  if bitand(publishSwitch, 2) == 2 then
    vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼113
    113▲ VelodyneScan')
    -- simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
115 elseif bitand(publishSwitch, 4) == 4 then
    vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼116
    116▲ VelodynePacket')
    simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
  end
-- #DE# --]]
120 end

-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
  -- Retrieve sensor data
125 data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle,  ▼125
    125▲ simGetSimulationTimeStep())
  -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

  -- Debug options
130 -- print('----------------')
  -- print(bitand(debugOutput, 1))
  -- print(bitand(debugOutput, 2))
  -- print(bitand(debugOutput, 4))
  -- print('----------------')
135
  -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
```

```
    simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
  else
    pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
  end
end

-- Construct the point cloud from raw data: x, y and z coordinates for every  ▼145
 145▲ point
-- Reference: From the original VPL_16 model script
local pointCloudData = {}
for i = 0, #data / 3 - 1, 1 do
  -- Temporary assignment to a vector d
  d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
  -- Transformation to sensor frame
  -- No transformation results in point cloud rotated by 90 deg around z
  d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
  data[3 * i + 1] = d[1]
  data[3 * i + 2] = d[2]
  data[3 * i + 3] = d[3]
  table.insert(pointCloudData, d)
  -- print(d)
end

-- Display the detected points
if displayPointCloud == 1 then
  simInsertPointsIntoPointCloud(pointCloud, 0, data)
end

if bitand(publishSwitch, 2) == 2 or bitand(publishSwitch, 4) == 4 then
-- --[[ #DS# -- Comment this line for raw message handling, currently only a  ▼167
 167▲ dummy message publishing implemented
  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402,  ▼176
 176▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
```

```
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
-- #S#
      -- table.insert(tpcl, {1, 1, 1})
      -- table.insert(tpcl, {1.0, 1.0, 1.0})
      -- table.insert(tpcl, {2, 2, 2})
-- #E#
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    if bitand(publishSwitch, 2) == 2 then
      pointCloudRawData['header'] = {seq = 0, stamp = simGetSimulationTimeStep() ▼202
 202▲ , frame_id = "frame00"}
      -- Apparently neither indirect nor direct assignment of packets object  ▼203
 203▲ works
      pointCloudRawData['packets']  = {stamp = simGetSimulationTimeStep(), data  ▼204
 204▲ = simPackFloats(tpcl)}
      -- Lua runtime error: [string "[embScript_51284011.lua] SCRIPT  ▼205
 205▲ velodyneVPL_16"]:253: read__velodyne_msgs__VelodyneScan: field packets:  ▼205
 205▲ expected array (simExtRosInterface_publish @ 'RosInterface' plugin)
-- #S#
      -- pointCloudRawData['packets']  = {stamp = simGetSimulationTimeStep(),  ▼207
 207▲ data = simPackUInts(tpcl)}
      -- local pointCloudRawDataPackets = {}
      -- pointCloudRawDataPackets = {stamp = simGetSimulationTimeStep(), data =  ▼209
 209▲ simPackFloats(tpcl)}
      -- pointCloudRawData['packets'] = pointCloudRawDataPackets
      -- Equivalent to above
      -- pointCloudRawData = { packets  = {
        -- stamp = simGetSimulationTimeStep(), data = simPackFloats(tpcl)
        -- }
      -- }
-- #E#
    elseif bitand(publishSwitch, 4) == 4 then
      pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼218
 218▲ simPackFloats(tpcl)}
    end
-- #S#
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼221
 221▲ simPackUInts(tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼222
 222▲ simPackUInts(data)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼223
 223▲ simPackUInts(tdata)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼224
 224▲ simPackUInts(pointCloudData)} -- CRASH
-- #E#

    if bitand(debugOutput, 2) == 2 then
      print('-*- VLP16 Raw START')
      print('tpcl')
```

```
        print(#tpcl)
        print('tdata')
        print(#tdata)
    -- #S#
        -- print('h')
        -- print(type(h))
        -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua]  ▼236
    236▲ SCRIPT velodyneVPL...":140: attempt to get length of global 'h' (a  ▼236
    236▲ number value)
        -- print(type(tpcl))
        -- print(type(simPackUInts(tpcl)))
        -- print(type(simPackFloats(tpcl)))
        -- print(#simPackUInts(tpcl))
        -- print(type(tdata))
        -- print(tdata)
        -- for i, j in pairs (tdata) do
          -- print(i, j)
        -- end
        -- for i, j in pairs (data) do
          -- print(i, j)
        -- end
        -- print(#simPackUInts(tdata)) -- nil
        -- print('raw')
        -- print(type(pointCloudRawData.data))
        -- print(#pointCloudRawData.data)
        -- print('-*- VLP16 Raw END')
    -- #E#
      end
      simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
      rmcounter = rmcounter + 1
    end
    if bitand(debugOutput, 2) == 2 then
      print('rmcounter')
      print(rmcounter)
      print('-*- VLP16 Raw END')
    end
    -- #S#
    -- #E#
  end
  -- #DE# --]]

  if bitand(publishSwitch, 1) == 1 then
    -- Publish point cloud

    -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
    -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼273
    273▲ Software/V-REP/robot.lua
    -- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
    local pointCloudData = {}
    --[[
    -- The frame of VLP16 shifts all the points to that frame
    -- If frame_id set to sensor frame in current configuration results in point  ▼278
    278▲ cloud shifted to the sensor position
```

```
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),
    frame_id = "vlp16"}
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),
    frame_id = "vpl16"}
    --]]
    pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),
    frame_id = "frame00"}
    -- Describes the channels and their layout in the binary data blob
    local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
    local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
    local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
    pointCloudData['fields'] = {field_x, field_y, field_z}
    pointCloudData['is_bigendian'] = false
    pointCloudData['is_dense'] = true
    pointCloudData['data'] = simPackFloats(data) --  Works
    -- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the
    data field being of type uint8
    -- VLP16 has 16 row scans, however the point cloud data is not always ordered
    and it differs in number of received points
    --[[
    -- Theoretically this should work, but the bytes offset still drop the
    messages
    pointCloudData['height'] = 16 -- Ordered, 16 scan lines
    pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData
    .height)
    pointCloudData['point_step'] = math.floor(#pointCloudData.data /
    pointCloudData.width / pointCloudData.height)
    --]]
    -- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
    pointCloudData['height'] = 1 -- Unordered
    -- Effectively number of cloud points, each having 3 coordinates
    pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
    -- Length of a point in bytes
    pointCloudData['point_step'] = math.floor(#pointCloudData.data /
    pointCloudData.width)
    -- Length of a row in bytes
    pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData['
    point_step']

    if bitand(debugOutput, 4) == 1 then
      print('-- VLP16 PointCloud2 START')
      print(#pointCloudData.data)
      print(pointCloudData.width * pointCloudData.point_step)
      print(pointCloudData.point_step)
      print(#pointCloudData.data - math.floor(#pointCloudData.data /
    pointCloudData.width) * pointCloudData.width)
      print(#pointCloudData.data - pointCloudData.width * pointCloudData.
    point_step)
      print(pointCloudData.width)
  -- #S#
      -- print(type(pointCloudData))
      -- print(type(tdata))
  -- #E#
```

```
      print('-- VLP16 PointCloud2 END')
    end

    simExtRosInterface_publish(vlp16Pub, pointCloudData)
  end

  -- --[[
  -- Send transforms
  tf.header.stamp = simGetSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00' -- Absolute frame reference
      -- tf.header.frame_id = 'robot01' -- Relative to robot frame reference
      -- Get pose relative to worldframe
      position = simGetObjectPosition(handle, worldHandle) -- Absolute frame ▼336
    336▲ reference
      orientation = simGetObjectQuaternion(handle, worldHandle) -- Absolute frame ▼337
    337▲ reference
      -- position = simGetObjectPosition(handle, robot01Handle) -- Relative to ▼338
    338▲ robot frame reference
      -- orientation = simGetObjectQuaternion(handle, robot01Handle) -- Relative ▼339
    339▲ to robot frame reference
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = simGetObjectPosition(handle, vlp16Handle)
      orientation = simGetObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end
  --]]

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    simAddStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
```

```
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 22: Lua script file: ugv-04–scripts–threaded-move-waypoint.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 04 -- Scripts
-- # Waypoint pose threaded script
-- # Notes:
-- # * The UGV collects readings from VLP16 and publishes it in PointCloud2 message ▼10
   10▲ format. UGV is mobile using forward and backward movements. The raw ▼10
   10▲ message format is implemented but not used, since data formatting is not ▼10
   10▲ working. An additional object was added to test the filter.
-- # * Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis. ▼11
   11▲ Both camera frames are published in TF.
-- # * Threaded script is necessary for simMoveToObject function, which is now ▼12
   12▲ unused, therefore the scripts can be changed to non-threaded child or ▼12
   12▲ implemented back into robot script
-- ###


threadFunction=function()

  while simGetSimulationState() ~= sim_simulation_advancing_abouttostop do
    -- Step for waypoint switch
    if isint(counter / 2) then
      -- Using simMoveToObject should work in both cases below, but it does leave ▼21
    21▲ the position at zero
      -- Probably the velocity and acceleration parameters should be changed
      -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[' ▼23
    23▲ waypoint0' .. waypointcounter], 3, 1, 1, nil)
      -- deltaTimeLeft = simMoveToObject(waypointHandle, waypointsHandles[ ▼24
    24▲ waypointcounter], 3, 1, 0, nil)

      -- Since the approach does not work properly, direct position and orientation ▼26
    26▲ assignment is used
      -- Get current waypoint position and orientation
      position = simGetObjectPosition(waypointsHandles['waypoint0' .. ▼28
    28▲ waypointcounter], -1)
      eulerAngles = simGetObjectOrientation(waypointsHandles['waypoint0' .. ▼29
    29▲ waypointcounter], -1)
      -- Set marker to current waypoint
      position = simSetObjectPosition(waypointHandle, -1, position)
      eulerAngles = simSetObjectOrientation(waypointHandle, -1, eulerAngles)

      -- Increase waypoint counter
      waypointcounter = waypointcounter + 1
      -- Reset waypoint counter condition
      if (waypointcounter == 10) then
        waypointcounter = 0
      end
    end
```

```
    position = simGetObjectPosition(waypointHandle, -1)
    eulerAngles = simGetObjectOrientation(waypointHandle, -1)
    -- Print verbose information
    -- print('Loop: ' .. counter .. ', waypoint0' .. waypointcounter .. ', pos = '
    .. position[1] .. ', ' .. position[2] .. ', ' .. position[3] .. ', or = '
    .. eulerAngles[1] .. ', ' .. eulerAngles[2] .. ', ' .. eulerAngles[3])

    counter = counter + 1
    simSwitchThread()
  end

end

-- Put some initialization code here:
simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching

-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- Frames
worldHandle = simGetObjectHandle('frame00')

-- Waypoints
waypointsHandlesNames = {}
waypointHandle = simGetObjectHandle('waypoint')

for i = 1, 10, 1 do
  waypointsHandlesNames[i] = 'waypoint0' .. i - 1
end

waypointsHandles = {}
-- Asign waypoint handles to array
for i, link in ipairs(waypointsHandlesNames) do
  waypointsHandles[link] = simGetObjectHandle(link)
end

-- waypointsPub = simExtROS_enablePublisher('/vrep/waypoints', 1,
    simros_strmcmd_get_object_pose, waypointHandle, worldHandle, 'frame00')
waypointsPub = simExtROS_enablePublisher('/vrep/waypoints', 1,
    simros_strmcmd_get_object_pose, waypointHandle, -1, 'frame00')

counter = 1

waypointcounter = 0

-- Here we execute the regular thread code:
res, err = xpcall(threadFunction, function(err) return debug.traceback(err) end)

if not res then
```

```
90    simAddStatusbarMessage('Lua runtime error: ' .. err)
end

-- Put some clean-up code here:
```

Script 23: Text file: ugv-03.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Text
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP scene - UGV 03 scenario

# General description
The UGV collects readings from VLP16 and publishes it in PointCloud2 message format ▼9
    9▲ . UGV is mobile using forward and backward movements. The raw message format  ▼9
    9▲ is implemented but not used, since data formatting is not working. An  ▼9
    9▲ additional object was added to test the filter.

# Stereo vision parameters
Camera 01 is translated with respect to camera 02 by 0.2 m along the Y axis.
```

Script 24: XML configuration file: package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>coslam_vrep</name>
  <version>0.5.0</version>
  <description>CoSLAM V-REP</description>
  <maintainer email="edskrzypcz2016@eleves.ec-natnes.fr">Ernest Skrzypczyk</ ▼6
    6▲ maintainer>
   <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>GPLv3</license>
   <!-- <url type="website">http://wiki.ros.org/coslam-vrep</url> -->
  <author email="emeres.code@onet.eu">Ernest Skrzypczyk</author>

   <!-- The *_depend tags are used to specify dependencies -->
   <!-- Dependencies can be catkin packages or system dependencies -->
   <!-- Examples: -->
   <!-- Use build_depend for packages you need at compile time: -->
   <!--   <build_depend>message_generation</build_depend> -->
   <!-- Use buildtool_depend for build tool packages: -->
   <!--   <buildtool_depend>catkin</buildtool_depend> -->
   <!-- Use run_depend for packages you need at runtime: -->
   <!--   <run_depend>message_runtime</run_depend> -->
   <!-- Use test_depend for packages you need only for testing: -->
   <!--   <test_depend>gtest</test_depend> -->

  <buildtool_depend>catkin</buildtool_depend>
  <buildtool_depend>velodyne_msgs</buildtool_depend>
  <build_depend>libpcl-all-dev</build_depend>
<!--
  <run_depend>libpcl-all</run_depend>
-->
  <depend>cv_bridge</depend>
  <depend>geometry_msgs</depend>
  <depend>image_transport</depend>
  <depend>python-numpy</depend>
  <depend>roscpp</depend>
  <depend>rospy</depend>
  <depend>sensor_msgs</depend>
  <depend>velodyne_msgs</depend>
  <depend>vrep_common</depend>

   <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>

</package>
```

Script 25: C++ header file: coslam_vrep.hpp

```
/**
 * @file
 * @author Ernest Skrzypczyk
 *
 * @date 28.04.17
 *
 * @brief Header file holding V-REP specific parameters and global references for the ROS nodes
 *
 * Project github repository
 *
 * @see https://github.com/em-er-es/coslam/coslam_vrep
 *
 */

// //! \section secmacros "Macros"
//! Concatenate helper
#define CC2(arg1, arg2) arg1 ## arg2
//! Concatenate macro
#define CC(arg1, arg2) CC2(arg1, arg2)

// //! \section seccoslamvrep "CoSLAM V-REP"
// //! \subsection subsecros "ROS"
//! ROS full package name
#define PKGNAME "CoSLAM V-REP"
//! ROS package name
#define PKG "coslam_vrep"
//! ROS package version
#define VERSION 0.5

//! @section Output control
//! Verbose switch
#ifndef VERBOSE
  #define VERBOSE 1
#endif
//! Debug switch
#ifndef DEBUG
  #define DEBUG 0
#endif
//! Debug loop counter filter
#ifndef DEBUG_LC
  #define DEBUG_LC 100000
#endif

//! @section Node names
// #define NN_XX5XX "XX5XX"
// #define NN_XX4XX "XX4XX"
// #define NN_XX3XX "XX3XX"
//! Velodyne Puck VLP16
#define NN_VLP16 "vlp16"
//! Robot control
#define NN_RBCTL "rbctl"
```

```
//! Stereo vision
#define NN_SVDPE "svdpe"
//! Stereo image preprocessing
#define NN_SIPPR "sippr"
//! Local ICP
#define NN_LICPR "licpr"


// ## Mathematical constants
//! Pi
#define PI 3.1415926535


// ## GNU/Linux terminal color codes
//! Reset
#define CR "\033[0m"


// ## Colors
//! Color output switch
#define COLORCODES 1
//! Color 1 - #5F5FFE
#define C1 "\033[38;5;63m"
//! Color 2 - #FFD700
#define C2 "\033[38;5;220m"
//! Color 3 - #5FFFFF
#define C3 "\033[38;5;87m"
//! Color 4 - #5FFF86
#define C4 "\033[38;5;84m"
//! Color 5 - #790303
#define C5 "\033[38;5;160m"
//! Color 6 - #D7005F
#define C6 "\033[38;5;161m"
//! Color 7 - #D70080
#define C7 "\033[38;5;162m"
//! Color 8 - #005F00
#define C8 "\033[38;5;22m"

//! Error
#define CEE "\033[38;5;124m" /* Error */
//! Success
#define CSS "\033[38;5;154m" /* Success */
//! Warning
#define CWW "\033[38;5;202m" /* Warning */
```

Script 26: C++ header file: licp.hpp

```
/**
 * @file
 * @author Ernest Skrzypczyk
 *
 * @date 12.07.17
 *
 * @brief Header file holding V-REP specific parameters and global references for the ROS nodes
 *
 * Project github repository
 *
 * @see https://github.com/em-er-es/coslam/coslam_vrep
 *
 */

// ## Topic descriptions
// #define TOPIC_NODE_NAME "/PREFIX/DESTINATION"
#define TP_PREFIX "/vrep/vlp16/"
//! Topic holding filtered VLP16 point cloud data in PointCloud2 format
#define TP_LICP_I_1 "/vlp16/pcl2filtered"
//! Topic holding stereo vision estimated point cloud data in PointCloud2 format
#define TP_LICP_I_2 "/stereo_vision/output/pcl"
//! Topic holding estimated pose of main camera of the stereo vision system
#define TP_LICP_O_1 "/licp/output/pose"
```

Script 27: C++ header file: control_robot.hpp

```
/**
 * @file
 * @author Ernest Skrzypczyk
 *
 * @date 20.05.17
 *
 * @brief Header file holding V-REP specific parameters and global references for the ROS nodes
 *
 * Project github repository
 *
 * @see https://github.com/em-er-es/coslam/coslam_vrep
 *
 */
```

Script 28: C++ header file: vlp16.hpp

```
/**
 * @file
 * @author Ernest Skrzypczyk
 *
 * @date 28.04.17
 *
 * @brief Header file holding VLP16 specific parameters
 *
 * Project github repository
 *
 * @see https://github.com/em-er-es/coslam/coslam_vrep
 *
 */

// ## Topic descriptions
// #define TOPIC_NODE_NAME "/PREFIX/DESTINATION"
#define TP_PREFIX "/vrep/vlp16/"
//! Topic holding VLP16 raw scan data
// #define TP_VLP16_RAW CC(TP_PREFIX, "raw")
#define TP_VLP16_I_RAW "/vrep/vlp16/raw"
//! Topic holding VLP16 point cloud data in PointCloud2 format
// #define TP_VLP16_PCL2 CC(TP_PREFIX, "pcl2")
#define TP_VLP16_I_PCL "/vrep/vlp16/pcl2"
//! Topic holding filtered VLP16 point cloud data
// #define TP_VLP16_PCL CC(TP_PREFIX, "pcl2filtered")
#define TP_VLP16_O_PCL "/vlp16/pcl2filtered"


// ## Parameters
// ### VLP16
//! VLP16 Position -- x
#define VLP16_POS_X 0.0000
//! VLP16 Position -- y
#define VLP16_POS_Y 0.0000
//! VLP16 Position -- z
#define VLP16_POS_Z 0.0000
//! VLP16 Orientation -- around x
#define VLP16_ROT_X 0.0000
//! VLP16 Orientation -- around y
#define VLP16_ROT_Y 0.0000
//! VLP16 Orientation -- around z
#define VLP16_ROT_Z 0.0000
//! Robot01 wheel left radius
#define ROBOT01_WHEEL_RADIUS_L 0.0076
//! Robot01 wheel right radius
#define ROBOT01_WHEEL_RADIUS_R 0.0076
```

Script 29: Python script file: ___init___.py

Script 30: Text file: vrep-170623-robot01-ugv_04–vlp16-01.txt

```
# ./record-vrep.sh -- 230617 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170623-robot01-ugv_04--vlp16-01.bag
path:        vrep-170623-robot01-ugv_04--vlp16-01.bag
version:     2.0
duration:    1:29s (89s)
start:       Jun 23 2017 19:54:27.24 (1498240467.24)
end:         Jun 23 2017 19:55:56.93 (1498240556.93)
size:        42.2 MB
messages:    5332
compression: none [54/54 chunks]
types:       sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage      [94810edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo    [66334ab2212d3c89226a89b7a37b2f95]
topics:      /tf                  4744 msgs    : tf2_msgs/TFMessage
             /vrep/info            308 msgs    : vrep_common/VrepInfo
             /vrep/vlp16/pcl2      280 msgs    : sensor_msgs/PointCloud2
```

Script 31: Text file: vrep-170630-robot01-ugv_04–vlp16-02.txt

```
# ./record-vrep.sh -- 300617 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170630-robot01-ugv_04--vlp16-02.bag
path:        vrep-170630-robot01-ugv_04--vlp16-02.bag
version:     2.0
duration:    2:36s (156s)
start:       Jun 30 2017 12:42:26.21 (1498819346.21)
end:         Jun 30 2017 12:45:03.09 (1498819503.09)
size:        67.4 MB
messages:    9289
compression: none [78/78 chunks]
types:       dynamic_reconfigure/ConfigDescription [757 ▼10
    10▲ ce9d44ba8ddd801bb30bc456f946f]
             geometry_msgs/PoseStamped             [ ▼11
    11▲ d3812c3cbc69362b77dc0b19b345f8f5]
             sensor_msgs/CompressedImage           [8 ▼12
    12▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/JointState                [3066 ▼13
    13▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2               [1158 ▼14
    14▲ d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage                    [94810 ▼15
    15▲ edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo                  [66334 ▼16
    16▲ ab2212d3c89226a89b7a37b2f95]
topics:      /tf                                                                  6579  ▼17
    17▲ msgs    : tf2_msgs/TFMessage
             /vrep/info                                                           387  ▼18
    18▲ msgs    : vrep_common/VrepInfo
             /vrep/joints/robot01/state                                           387  ▼19
    19▲ msgs    : sensor_msgs/JointState
             /vrep/pose/robot01                                                   387  ▼20
    20▲ msgs    : geometry_msgs/PoseStamped
             /vrep/stereo_vision/camera01/compressed                              387  ▼21
    21▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera01/compressed/parameter_descriptions         1  ▼22
    22▲ msg     : dynamic_reconfigure/ConfigDescription
             /vrep/stereo_vision/camera02/compressed                              387  ▼23
    23▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera02/compressed/parameter_descriptions         1  ▼24
    24▲ msg     : dynamic_reconfigure/ConfigDescription
             /vrep/vlp16/pcl2                                                     387  ▼25
    25▲ msgs    : sensor_msgs/PointCloud2
             /vrep/waypoints                                                      386  ▼26
    26▲ msgs    : geometry_msgs/PoseStamped
```

Script 32: Text file: vrep-170720-robot01-ugv_05–drive-full.txt

```
1 # record-vrep.sh -- 200717 -- localhost.localdomain@ -- vrep-170720-robot01-ugv_05 ▼1
    1▲ --drive-full.bag
  path:      vrep-170720-robot01-ugv_05--drive-full.bag
  version:   2.0
  size:      4.0 KB
```

Script 33: Text file: vrep-170704-robot01-ugv_04–vlp16-03.txt

```
# ./record-vrep.sh -- 040717 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170704-robot01-ugv_04--vlp16-03.bag
path:        vrep-170704-robot01-ugv_04--vlp16-03.bag
version:     2.0
duration:    1:24s (84s)
start:       Jul 04 2017 17:15:28.80 (1499181328.80)
end:         Jul 04 2017 17:16:53.31 (1499181413.31)
size:        34.9 MB
messages:    3038
compression: none [41/41 chunks]
types:       dynamic_reconfigure/ConfigDescription [757 ▼10
    10▲ ce9d44ba8ddd801bb30bc456f946f]
             geometry_msgs/PoseStamped             [ ▼11
    11▲ d3812c3cbc69362b77dc0b19b345f8f5]
             sensor_msgs/CompressedImage           [8 ▼12
    12▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/JointState                [3066 ▼13
    13▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2               [1158 ▼14
    14▲ d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage                    [94810 ▼15
    15▲ edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo                  [66334 ▼16
    16▲ ab2212d3c89226a89b7a37b2f95]
topics:      /tf                                                                1617   ▼17
    17▲ msgs    : tf2_msgs/TFMessage
             /vrep/info                                                          205   ▼18
    18▲ msgs    : vrep_common/VrepInfo
             /vrep/joints/robot01/state                                          202   ▼19
    19▲ msgs    : sensor_msgs/JointState
             /vrep/pose/robot01                                                  202   ▼20
    20▲ msgs    : geometry_msgs/PoseStamped
             /vrep/stereo_vision/camera01/compressed                             202   ▼21
    21▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera01/compressed/parameter_descriptions        2   ▼22
    22▲ msgs    : dynamic_reconfigure/ConfigDescription (2 connections)
             /vrep/stereo_vision/camera02/compressed                             202   ▼23
    23▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera02/compressed/parameter_descriptions        2   ▼24
    24▲ msgs    : dynamic_reconfigure/ConfigDescription (2 connections)
             /vrep/vlp16/pcl2                                                    203   ▼25
    25▲ msgs    : sensor_msgs/PointCloud2
             /vrep/waypoints                                                     201   ▼26
    26▲ msgs    : geometry_msgs/PoseStamped
```

Script 34: Text file: vrep-170615-robot01-ugv_03–vlp16-01.txt

```
# ./record-vrep.sh -- 150617 -- greybox@  -- vrep-170615-robot01-ugv_03--vlp16-01. ▼1
    1▲ bag
path:        vrep-170615-robot01-ugv_03--vlp16-01.bag
version:     2.0
duration:    2:42s (162s)
start:       Jun 15 2017 23:12:29.08 (1497561149.08)
end:         Jun 15 2017 23:15:11.91 (1497561311.91)
size:        39.1 MB
messages:    1022
compression: none [52/52 chunks]
types:       dynamic_reconfigure/ConfigDescription [757 ▼10
    10▲ ce9d44ba8ddd801bb30bc456f946f]
             sensor_msgs/CompressedImage           [8 ▼11
    11▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/JointState                [3066 ▼12
    12▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2               [1158 ▼13
    13▲ d486dd51d683ce2f1be655c3c181]
             vrep_common/VrepInfo                  [66334 ▼14
    14▲ ab2212d3c89226a89b7a37b2f95]
topics:      /joints/robot01/state                          204 msgs    : ▼15
    15▲ sensor_msgs/JointState
             /vrep/image01/compressed                       204 msgs    : ▼16
    16▲ sensor_msgs/CompressedImage
             /vrep/image01/compressed/parameter_descriptions  1 msg     : ▼17
    17▲ dynamic_reconfigure/ConfigDescription
             /vrep/image02/compressed                       204 msgs    : ▼18
    18▲ sensor_msgs/CompressedImage
             /vrep/image02/compressed/parameter_descriptions  1 msg     : ▼19
    19▲ dynamic_reconfigure/ConfigDescription
             /vrep/info                                     204 msgs    : ▼20
    20▲ vrep_common/VrepInfo
             /vrep/vlp16/pcl2                               204 msgs    : ▼21
    21▲ sensor_msgs/PointCloud2
```

Script 35: Text file: vrep-170623-robot01-ugv_04–vlp16-02.txt

```
# ./record-vrep.sh -- 230617 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170623-robot01-ugv_04--vlp16-02.bag
path:        vrep-170623-robot01-ugv_04--vlp16-02.bag
version:     2.0
duration:    55.0s
start:       Jun 23 2017 19:56:12.47 (1498240572.47)
end:         Jun 23 2017 19:57:07.47 (1498240627.47)
size:        30.8 MB
messages:    3876
compression: none [39/39 chunks]
types:       sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage      [94810edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo    [66334ab2212d3c89226a89b7a37b2f95]
topics:      /tf                  3468 msgs    : tf2_msgs/TFMessage
             /vrep/info            204 msgs    : vrep_common/VrepInfo
             /vrep/vlp16/pcl2      204 msgs    : sensor_msgs/PointCloud2
```

Script 36: Text file: vrep-170704-robot01-ugv_04–vlp16-04.txt

```
# ./record-vrep.sh -- 040717 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170704-robot01-ugv_04--vlp16-04.bag
path:        vrep-170704-robot01-ugv_04--vlp16-04.bag
version:     2.0
duration:    1:24s (84s)
start:       Jul 04 2017 19:55:25.01 (1499190925.01)
end:         Jul 04 2017 19:56:49.12 (1499191009.12)
size:        34.9 MB
messages:    3038
compression: none [41/41 chunks]
types:       dynamic_reconfigure/ConfigDescription [757 ▼10
    10▲ ce9d44ba8ddd801bb30bc456f946f]
             geometry_msgs/PoseStamped             [ ▼11
    11▲ d3812c3cbc69362b77dc0b19b345f8f5]
             sensor_msgs/CompressedImage           [8 ▼12
    12▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/JointState                [3066 ▼13
    13▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2               [1158 ▼14
    14▲ d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage                    [94810 ▼15
    15▲ edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo                  [66334 ▼16
    16▲ ab2212d3c89226a89b7a37b2f95]
topics:      /tf                                                                1617  ▼17
    17▲ msgs    : tf2_msgs/TFMessage
             /vrep/info                                                          205  ▼18
    18▲ msgs    : vrep_common/VrepInfo
             /vrep/joints/robot01/state                                          202  ▼19
    19▲ msgs    : sensor_msgs/JointState
             /vrep/pose/robot01                                                  202  ▼20
    20▲ msgs    : geometry_msgs/PoseStamped
             /vrep/stereo_vision/camera01/compressed                             202  ▼21
    21▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera01/compressed/parameter_descriptions        2  ▼22
    22▲ msgs    : dynamic_reconfigure/ConfigDescription (2 connections)
             /vrep/stereo_vision/camera02/compressed                             202  ▼23
    23▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera02/compressed/parameter_descriptions        2  ▼24
    24▲ msgs    : dynamic_reconfigure/ConfigDescription (2 connections)
             /vrep/vlp16/pcl2                                                    203  ▼25
    25▲ msgs    : sensor_msgs/PointCloud2
             /vrep/waypoints                                                     201  ▼26
    26▲ msgs    : geometry_msgs/PoseStamped
```

Script 37: Text file: vrep-170616-robot01-ugv_03–vlp16-01.txt

```
# ./record-vrep.sh -- 160617 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170616-robot01-ugv_03--vlp16-01.bag
path:        vrep-170616-robot01-ugv_03--vlp16-01.bag
version:     2.0
duration:    53.3s
start:       Jun 16 2017 18:09:22.14 (1497629362.14)
end:         Jun 16 2017 18:10:15.43 (1497629415.43)
size:        39.4 MB
messages:    2046
compression: none [52/52 chunks]
types:       dynamic_reconfigure/ConfigDescription [757 ▼10
    10▲ ce9d44ba8ddd801bb30bc456f946f]
             sensor_msgs/CompressedImage           [8 ▼11
    11▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/JointState                [3066 ▼12
    12▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2               [1158 ▼13
    13▲ d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage                    [94810 ▼14
    14▲ edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo                  [66334 ▼15
    15▲ ab2212d3c89226a89b7a37b2f95]
topics:      /joints/robot01/state                              204 msgs    : ▼16
    16▲ sensor_msgs/JointState
             /tf                                               1021 msgs    : ▼17
    17▲ tf2_msgs/TFMessage
             /vrep/image01/compressed                          205 msgs    : ▼18
    18▲ sensor_msgs/CompressedImage
             /vrep/image01/compressed/parameter_descriptions     1 msg     : ▼19
    19▲ dynamic_reconfigure/ConfigDescription
             /vrep/image02/compressed                          204 msgs    : ▼20
    20▲ sensor_msgs/CompressedImage
             /vrep/image02/compressed/parameter_descriptions     1 msg     : ▼21
    21▲ dynamic_reconfigure/ConfigDescription
             /vrep/info                                        205 msgs    : ▼22
    22▲ vrep_common/VrepInfo
             /vrep/vlp16/pcl2                                  205 msgs    : ▼23
    23▲ sensor_msgs/PointCloud2
```

Script 38: Text file: vrep-170720-robot01-ugv_05–wp01-wp02.txt

```
# /home/emeres/.ros/workspace/src/coslam_vrep/sessions/record-vrep.sh -- 200717 --  ▼1
    1▲ localhost.localdomain@ -- vrep-170720-robot01-ugv_05--wp01-wp02.bag
path:        vrep-170720-robot01-ugv_05--wp01-wp02.bag
version:     2.0
duration:    1:12s (72s)
start:       Jul 20 2017 23:08:40.18 (1500584920.18)
end:         Jul 20 2017 23:09:52.58 (1500584992.58)
size:        123.2 MB
messages:    1797
compression: none [125/125 chunks]
types:       dynamic_reconfigure/Config             [958 ▼10
    10▲ f16a05573709014982821e6822580]
             dynamic_reconfigure/ConfigDescription [757 ▼11
    11▲ ce9d44ba8ddd801bb30bc456f946f]
             geometry_msgs/PoseStamped              [ ▼12
    12▲ d3812c3cbc69362b77dc0b19b345f8f5]
             rosgraph_msgs/Clock                    [ ▼13
    13▲ a9c97c1d230cfc112e270351a944ee47]
             rosgraph_msgs/Log                      [ ▼14
    14▲ acffd30cd6b6de30f120938c17c593fb]
             sensor_msgs/CompressedImage            [8 ▼15
    15▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/Image                      [060021388200 ▼16
    16▲ f6f0f447d0fcd9c64743]
             sensor_msgs/JointState                 [3066 ▼17
    17▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2                [1158 ▼18
    18▲ d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage                     [94810 ▼19
    19▲ edda583a504dfda3829e70d7eec]
             theora_image_transport/Packet          [33 ▼20
    20▲ ac4e14a7cff32e7e0d65f18bb410f3]
             vrep_common/VrepInfo                   [66334 ▼21
    21▲ ab2212d3c89226a89b7a37b2f95]
topics:      /clock  ▼22
    22▲                                                          62 msgs      ▼22
    22▲ : rosgraph_msgs/Clock
             /rosout                                                          ▼23
    23▲ 137 msgs    : rosgraph_msgs/Log                (2 connections)
             /rosout_agg                                                      ▼24
    24▲ 148 msgs    : rosgraph_msgs/Log
             /stereo_vision/output/compressed  ▼25
    25▲                                  77 msgs    : sensor_msgs/ ▼25
    25▲ CompressedImage
             /stereo_vision/output/pcl2  ▼26
    26▲                                      37 msgs    : sensor_msgs/ ▼26
    26▲ PointCloud2
             /stereo_vision/output/pose  ▼27
    27▲                                      11 msgs    : geometry_msgs/ ▼27
    27▲ PoseStamped
             /tf                                                              ▼28
    28▲ 621 msgs    : tf2_msgs/TFMessage
```

```
          /vrep/info  ▼29
 29▲                                                          68 msgs     :  ▼29
 29▲ vrep_common/VrepInfo
30          /vrep/joints/robot01/state  ▼30
 30▲                                              62 msgs     : sensor_msgs/ ▼30
 30▲ JointState
          /vrep/pose/robot01  ▼31
 31▲                                                   62 msgs     :  ▼31
 31▲ geometry_msgs/PoseStamped
          /vrep/stereo_vision/camera01  ▼32
 32▲                                          62 msgs     : sensor_msgs/Image
          /vrep/stereo_vision/camera01/compressed  ▼33
 33▲                                  62 msgs     : sensor_msgs/CompressedImage
          /vrep/stereo_vision/camera01/compressed/parameter_descriptions  ▼34
 34▲          1 msg      : dynamic_reconfigure/ConfigDescription
35        /vrep/stereo_vision/camera01/compressed/parameter_updates  ▼35
 35▲              1 msg      : dynamic_reconfigure/Config
          /vrep/stereo_vision/camera01/compressedDepth/parameter_descriptions  ▼36
 36▲     1 msg      : dynamic_reconfigure/ConfigDescription
          /vrep/stereo_vision/camera01/compressedDepth/parameter_updates  ▼37
 37▲          1 msg      : dynamic_reconfigure/Config
          /vrep/stereo_vision/camera01/theora  ▼38
 38▲                                    64 msgs     : theora_image_transport/ ▼38
 38▲ Packet
          /vrep/stereo_vision/camera01/theora/parameter_descriptions  ▼39
 39▲              1 msg      : dynamic_reconfigure/ConfigDescription
40        /vrep/stereo_vision/camera01/theora/parameter_updates  ▼40
 40▲                  1 msg      : dynamic_reconfigure/Config
          /vrep/stereo_vision/camera02  ▼41
 41▲                                          62 msgs     : sensor_msgs/Image
          /vrep/stereo_vision/camera02/compressed  ▼42
 42▲                                  62 msgs     : sensor_msgs/CompressedImage
          /vrep/stereo_vision/camera02/compressed/parameter_descriptions  ▼43
 43▲          1 msg      : dynamic_reconfigure/ConfigDescription
          /vrep/stereo_vision/camera02/compressed/parameter_updates  ▼44
 44▲              1 msg      : dynamic_reconfigure/Config
45        /vrep/stereo_vision/camera02/compressedDepth/parameter_descriptions  ▼45
 45▲     1 msg      : dynamic_reconfigure/ConfigDescription
          /vrep/stereo_vision/camera02/compressedDepth/parameter_updates  ▼46
 46▲          1 msg      : dynamic_reconfigure/Config
          /vrep/stereo_vision/camera02/theora  ▼47
 47▲                                    65 msgs     : theora_image_transport/ ▼47
 47▲ Packet
          /vrep/stereo_vision/camera02/theora/parameter_descriptions  ▼48
 48▲              1 msg      : dynamic_reconfigure/ConfigDescription
          /vrep/stereo_vision/camera02/theora/parameter_updates  ▼49
 49▲                  1 msg      : dynamic_reconfigure/Config
50        /vrep/vlp16/pcl2  ▼50
 50▲                                                    62 msgs     :  ▼50
 50▲ sensor_msgs/PointCloud2
          /vrep/waypoints  ▼51
 51▲                                                     61 msgs     :  ▼51
 51▲ geometry_msgs/PoseStamped
```

Script 39: Bash compatible shell script file: record-vrep-ugv04.sh

```
1 #!/bin/bash
#./record-vrep.sh vrep-date-robot01-ugv_04--vlp16-01.bag /vrep/joints/robot01/state ▼2
    2▲  /vrep/pose/robot01 /vrep/stereo_vision/camera01 /vrep/stereo_vision/camera02 ▼2
    2▲  /vrep/info /vrep/vlp16/pcl2 /tf /vrep/waypoints
./record-vrep.sh vrep-date-robot01-ugv_04--vlp16-01.bag /vrep/joints/robot01/state  ▼3
    3▲ /vrep/pose/robot01 /vrep/stereo_vision/camera01/compressed /vrep/ ▼3
    3▲ stereo_vision/camera02/compressed /vrep/stereo_vision/camera01/compressed/ ▼3
    3▲ parameter_descriptions /vrep/stereo_vision/camera02/compressed/ ▼3
    3▲ parameter_descriptions /vrep/info /vrep/vlp16/pcl2 /tf /vrep/waypoints
```

Script 40: Text file: vrep-170630-robot01-ugv_04–vlp16-01.txt

```
# ./record-vrep.sh -- 300617 -- PC-PROBO-12.ec-nantes.fr@130.66.113.92 -- vrep ▼1
    1▲ -170630-robot01-ugv_04--vlp16-01.bag
path:        vrep-170630-robot01-ugv_04--vlp16-01.bag
version:     2.0
duration:    1:26s (86s)
start:       Jun 30 2017 17:53:23.23 (1498838003.23)
end:         Jun 30 2017 17:54:49.34 (1498838089.34)
size:        35.2 MB
messages:    4853
compression: none [41/41 chunks]
types:       dynamic_reconfigure/ConfigDescription [757 ▼10
    10▲ ce9d44ba8ddd801bb30bc456f946f]
             geometry_msgs/PoseStamped             [ ▼11
    11▲ d3812c3cbc69362b77dc0b19b345f8f5]
             sensor_msgs/CompressedImage           [8 ▼12
    12▲ f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/JointState                [3066 ▼13
    13▲ dcd76a6cfaef579bd0f34173e9fd]
             sensor_msgs/PointCloud2               [1158 ▼14
    14▲ d486dd51d683ce2f1be655c3c181]
             tf2_msgs/TFMessage                    [94810 ▼15
    15▲ edda583a504dfda3829e70d7eec]
             vrep_common/VrepInfo                  [66334 ▼16
    16▲ ab2212d3c89226a89b7a37b2f95]
topics:      /tf                                                                3435  ▼17
    17▲ msgs    : tf2_msgs/TFMessage
             /vrep/info                                                          204  ▼18
    18▲ msgs    : vrep_common/VrepInfo
             /vrep/joints/robot01/state                                          202  ▼19
    19▲ msgs    : sensor_msgs/JointState
             /vrep/pose/robot01                                                  202  ▼20
    20▲ msgs    : geometry_msgs/PoseStamped
             /vrep/stereo_vision/camera01/compressed                             202  ▼21
    21▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera01/compressed/parameter_descriptions        1  ▼22
    22▲ msg     : dynamic_reconfigure/ConfigDescription
             /vrep/stereo_vision/camera02/compressed                             202  ▼23
    23▲ msgs    : sensor_msgs/CompressedImage
             /vrep/stereo_vision/camera02/compressed/parameter_descriptions        1  ▼24
    24▲ msg     : dynamic_reconfigure/ConfigDescription
             /vrep/vlp16/pcl2                                                    203  ▼25
    25▲ msgs    : sensor_msgs/PointCloud2
             /vrep/waypoints                                                     201  ▼26
    26▲ msgs    : geometry_msgs/PoseStamped
```

Script 41: Bash compatible shell script file: generate-coordinate-system.sh

```
#!/bin/bash
OUTPUT="2d-frame.png"
arrow_head="path 'M 0,0  l -15,-5  +5,+5  -5,+5  +15,-5 z'"
convert -size 256x256 xc: \
  -draw 'line 128,0 128,256' \
  -draw 'line 0,128 256,128' \
  -draw "fill black translate 128,0 rotate -90 scale 2,1 $arrow_head" \
  -draw "fill black translate 256,128 rotate 0 scale 2,1 $arrow_head" \
  "$OUTPUT"
```

Script 42: Bash compatible shell script file: generate-chessboard.sh

```
#!/bin/bash
TILESIZE=0;
SIZEX=512;
SIZEY=512;
TILEX=2;
TILEY=2;
COLORA='#ffffff';
COLORB='#000000';
OUTPUT="chessboard-${TILEX}x${TILEY}.png";
if ! (command -v convert &>/dev/null); then echo ImageMagick dependency missing;  ▼10
    10▲ exit 1; fi

if [ $TILESIZE -gt 0 ]; then
convert -size ${SIZEX}x${SIZEY} xc:${COLORA} tmp1.png
convert -size ${SIZEX}x${SIZEY} xc:${COLORB} tmp2.png
else
convert -size $((SIZEX/TILEX))x$((SIZEY/TILEY)) xc:${COLORA} tmp1.png
convert -size $((SIZEX/TILEX))x$((SIZEY/TILEY)) xc:${COLORB} tmp2.png
fi

#for i in $(seq $((TILEX * TILEY / 2))); do ls -1 tmp*.png; done | $(xargs echo  ▼20
    20▲ montage -border 0 -tile ${TILEX}x${TILEY} -geometry ${SIZEX}x${SIZEY}+0+0)  ▼20
    20▲ "${OUTPUT}"
c=1; R="";
for i in $(seq $((TILEX * TILEY / 2))); do if [ $c -eq ${TILEX} ]; then c=1; if [ - ▼22
    22▲ z ${R} ]; then R="-r"; else R=""; fi; else c=$((c+1)); fi; ls -1 ${R} tmp*. ▼22
    22▲ png; done | $(xargs echo montage -border 0 -tile ${TILEX}x${TILEY} - ▼22
    22▲ geometry ${SIZEX}x${SIZEY}+0+0) "${OUTPUT}"

rm tmp*
```

Script 43: ROS launch file: camera_stereo_vision.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Launch stereo vision system view  -->
  <node
  pkg="image_view"
  type="image_view"
  name="stereo_vision_output"
  args="image:=/stereo_vision/output compressed"
  output="screen"
  respawn="false"
  >
  </node>
</launch>
```

Script 44: ROS launch file: vrep_session_latest.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Use simulation time -->
  <param name="/use_sime_time" value="true"/>
  <!-- Transform V-REP to simulation time -->
<!--
  <param name="/use_sime_time" value="false"/>
  <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info /clock ▼8
    8▲ rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -'-import ▼8
    8▲ rospy" />
-->

  <!-- Launch V-REP playback of latest session with time from bag -->
  <node pkg="rosbag" type="play" name="player" output="screen" args="-l --clock $( ▼12
    12▲ find coslam_vrep)/sessions/latest-vrep.bag"/>
  <!-- Launch V-REP playback of latest session with system time -->
<!--
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼15
    15▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
  <node pkg="rosbag" type="play" name="player" output="screen" args="-l $(find ▼16
    16▲ coslam_vrep)/sessions/latest-vrep.bag"/>
-->

  <!-- Launch V-REP with latest scene -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/latest.ttt" />
<!--
    <arg name="topic" value="/vrep/joint_setstates" />
    <arg name="topic" value="/vrep/joint_setpoint" />
 -->
  </include>

  <!-- Launch preprocessor node -->
  <!-- Filter noise with floor visible -->
  <node pkg="coslam_vrep" type="vlp16" name="vlp16" args="_filter/noise/sw:=true ▼30
    30▲ _filter/noise/threshold:=0.001 _filter/voxelgrid/sw:=false"/>
  <!-- Filter noise with floor invisible -->
<!--
  <node pkg="coslam_vrep" type="vlp16" name="vlp16" args="_filter/noise/sw:=true ▼33
    33▲ _filter/noise/threshold:=0.02 _filter/voxelgrid/sw:=false"/>
-->

  <!-- Launch stereo vision node -->
<!--
  <node pkg="coslam_vrep" type="stereo_vision.py" name="stereo_vision" args=""/>
-->

  <!-- Launch RViz -->
  <node pkg="rviz" type="rviz" name="rviz_vrep" args="-d $(find coslam_vrep)/ ▼42
    42▲ configuration/vrep_session.rviz"/>

  <!-- Launch main camera views -->
```

```
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
  <include file="$(find coslam_vrep)/launch/stereo_vision.launch" />
 -->

</launch>
```

Script 45: ROS launch file: camera02.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Launch secondary/right camera view  -->
  <node
  pkg="image_view"
  type="image_view"
  name="camera02"
  args="image:=/vrep/stereo_vision/camera02 compressed"
  output="screen"
  respawn="true"
  >
  </node>
</launch>
```

Script 46: ROS launch file: vlp16.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Launch VLP16 node with default parameters -->
  <!-- Launch preprocessor node -->
  <!-- Filter noise with floor visible -->
  <node pkg="coslam_vrep" type="vlp16" name="vlp16" args="_filter/noise/sw:=true
    _filter/noise/threshold:=0.001 _filter/voxelgrid/sw:=false"/>
  <!-- Filter noise with floor invisible -->
<!--
  <node pkg="coslam_vrep" type="vlp16" name="vlp16" args="_filter/noise/sw:=true
    _filter/noise/threshold:=0.02 _filter/voxelgrid/sw:=false"/>
-->
</launch>
```

Script 47: ROS launch file: vrep_ugv_03.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
   8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
   8▲ import rospy$
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼9
   9▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
-->
  <!-- Launch V-REP with UGV -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/ugv-03.ttt" />
<!--
    <arg name="topic" value="/vrep/joint_setstates" />
    <arg name="topic" value="/vrep/joint_setpoint" />
 -->
  </include>

  <!-- Launch main camera views -->
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
 -->

  <!-- Load parameters -->
  <rosparam file="$(find coslam_vrep)/launch/ugv_01a.yaml" command="load" ns="init ▼27
   27▲ "/>
</launch>
```

Script 48: YAML configuration file: ugv_01a.yaml

```
1 robot01: {x: -2.000, y: 2.000, z: 0.13879, alpha: 0.000, beta: 0.000, gamma: 0.000}
```

Script 49: ROS launch file: vrep_ugv_05.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
    8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
    8▲ import rospy$
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼9
    9▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
-->
  <!-- Launch V-REP with UGV -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/ugv-05.ttt" />
<!--
    <arg name="topic" value="/vrep/joint_setstates" />
    <arg name="topic" value="/vrep/joint_setpoint" />
 -->
  </include>

  <!-- Launch main camera views -->
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
 -->

  <!-- Load parameters -->
  <rosparam file="$(find coslam_vrep)/launch/ugv_01a.yaml" command="load" ns="init ▼27
    27▲ "/>
</launch>
```

Script 50: ROS launch file: vrep_session.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
    8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
    8▲ import rospy$
-->

  <!-- Launch V-REP playback of latest session with time from bag -->
  <node pkg="rosbag" type="play" name="player" output="screen" args="-l --clock $( ▼12
    12▲ find coslam_vrep)/sessions/latest-vrep.bag"/>
  <!-- Launch V-REP playback of latest session with system time -->
<!--
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼15
    15▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
  <node pkg="rosbag" type="play" name="player" output="screen" args="-l $(find  ▼16
    16▲ coslam_vrep)/sessions/latest-vrep.bag"/>
-->

  <!-- Launch main camera views -->
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
 -->

  <!-- Launch main camera views -->
  <node pkg="rviz" type="rviz" name="rviz_vrep" args="-d $(find coslam_vrep)/ ▼26
    26▲ configuration/vrep_session.rviz"/>
</launch>
```

Script 51: ROS launch file: vrep_ugv_02.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
    8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
    8▲ import rospy$
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼9
    9▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
-->
  <!-- Launch V-REP with UGV -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/ugv-02.ttt" />
<!--
    <arg name="topic" value="/vrep/joint_setstates" />
    <arg name="topic" value="/vrep/joint_setpoint" />
 -->
  </include>

  <!-- Launch main camera views -->
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
 -->

  <!-- Load parameters -->
  <rosparam file="$(find coslam_vrep)/launch/ugv_01a.yaml" command="load" ns="init ▼27
    27▲ "/>
</launch>
```

Script 52: Python script file: yaml-load.py

```
#!/bin/env python2
# -*- coding: utf-8 -*-
""" @author Ernest Skrzypczyk
  @date 27.05.2017
  Camera calibration with Python using OpenCV
"""

#from yaml import load, dump as yload, ydump
from yaml import load as yload
from yaml import dump as ydump

Filename = "ugv_01a.yaml"

with open(Filename) as File:
  ConfigurationDataRaw = yload(File)

print(ConfigurationDataRaw)

globals().update(ConfigurationDataRaw)

print(robot01)
```

Script 53: ROS launch file: vrep_ugv_04.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
    8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
    8▲ import rospy$
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼9
    9▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
-->
  <!-- Launch V-REP with UGV -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/ugv-04.ttt" />
<!--
    <arg name="topic" value="/vrep/joint_setstates" />
    <arg name="topic" value="/vrep/joint_setpoint" />
 -->
  </include>

  <!-- Launch main camera views -->
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
 -->

  <!-- Load parameters -->
  <rosparam file="$(find coslam_vrep)/launch/ugv_01a.yaml" command="load" ns="init ▼27
    27▲ "/>
</launch>
```

Script 54: ROS launch file: vrep_ugv_01.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
    8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
    8▲ import rospy$
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼9
    9▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
-->

  <!-- Launch V-REP with UGV -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/ugv-01.ttt" />
  </include>

  <!-- Launch main camera view  -->
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
</launch>
```

Script 55: ROS launch file: camera01.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Launch primary/left camera view  -->
  <node
  pkg="image_view"
  type="image_view"
  name="camera01"
  args="image:=/vrep/stereo_vision/camera01 compressed"
  output="screen"
  respawn="true"
  >
  </node>
</launch>
```

Script 56: ROS launch file: vrep_ugv_uav_01.launch

```
<?xml version="1.0"?>
<launch>
        <!-- Use simulation time -->
        <param name="/use_sime_time" value="true"/>
        <!-- Transform V-REP to simulation time -->
<!--
        <param name="/use_sime_time" value="false"/>
        <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info / ▼8
    8▲ clock rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\- ▼8
    8▲ import rospy$
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼9
    9▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
-->
  <!-- Launch V-REP with UGV and UAV -->
  <include file="$(find vrep_plugin)/vrep.launch">
    <arg name="scene" value="$(find coslam_vrep)/scenes/ugv-uav-01.ttt" />
<!--
    Add appropriate prefixes for different robots
    <arg name="topic" value="/vrep/joint_setstates" />
    <arg name="topic" value="/vrep/joint_setpoint" />
 -->
  </include>

  <!-- Launch main camera views -->
<!--
  <include file="$(find coslam_vrep)/launch/camera01.launch" />
  <include file="$(find coslam_vrep)/launch/camera02.launch" />
 -->

  <!-- Load parameters -->
  <rosparam file="$(find coslam_vrep)/launch/ugv_01a.yaml" command="load" ns="init ▼28
    28▲ "/>
<!--
  <rosparam file="$(find coslam_vrep)/launch/uav_01a.yaml" command="load" ns="init ▼30
    30▲ "/>
 -->
</launch>
```

Script 57: ROS launch file: stereo_vision.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Launch stereo vision node -->
  <node pkg="coslam_vrep" type="stereo_vision.py" name="stereo_vision" args="" output="screen" respawn="true" />
<!--
  <node pkg="coslam_vrep" type="stereo_vision.py" name="stereo_vision" args="" />
  <node pkg="coslam_vrep" type="$(find stereo_vision.py)" name="stereo_vision" args =""/>
-->
</launch>
```

Script 58: ROS launch file: vrep_playback.launch

```
<?xml version="1.0"?>
<launch>
  <!-- Use simulation time -->
  <param name="/use_sime_time" value="true"/>
  <!-- Transform V-REP to simulation time -->
<!--
  <param name="/use_sime_time" value="false"/>
  <node name="sim_time" pkg="topic_tools" type="transform" args="/vrep/info /clock ▼8
    8▲ rosgraph_msgs/Clock 'rospy.Time.from_sec(m.simulationTime.data)' -\-import ▼8
    8▲ rospy" />
-->
  <!-- Launch V-REP playback of latest session with time from bag -->
  <node pkg="rosbag" type="play" name="player" output="screen" args="-l --clock $( ▼11
    11▲ find coslam_vrep)/sessions/latest-vrep.bag"/>
  <!-- Launch V-REP playback of latest session with system time -->
<!--
  <node pkg="rostopic" type="rostopic" name="clock" output="screen" args="pub -r 10 ▼14
    14▲  -v -s /clock rosgraph_msgs/Clock '{clock: now}'"/>
  <node pkg="rosbag" type="play" name="player" output="screen" args="-l $(find ▼15
    15▲ coslam_vrep)/sessions/latest-vrep.bag"/>
-->
</launch>
```

Script 59: Licence file

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
```

protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2)

```
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
```

```
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
```

terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no

more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

```
    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
```

violation by some reasonable means, this is the first time you have
425 received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

Termination of your rights under this section does not terminate the
430 licenses of parties who have received copies or rights from you under
this License. If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

435 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or
run a copy of the Program. Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
440 to receive a copy likewise does not require acceptance. However,
nothing other than this License grants you permission to propagate or
modify any covered work. These actions infringe copyright if you do
not accept this License. Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.
445

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
450 propagate that work, subject to this License. You are not responsible
for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
455 organization, or merging organizations. If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
460 Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License. For example, you may
465 not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.
470

11. Patents.

A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based. The
475 work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory

patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

```
  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
```

```
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

Script 60: Bash compatible shell script file: update.sh

```
#!/bin/bash
## @file
## @author Ernest Skrzypczyk
## @brief Runs scripts with executable flag
## @details This shell script can be further extended to include custom procedures.
## @details Add or modify scripts in ${SCRIPTSDIR} for additional processing.
## @note Pay attention to file names, since the filename of this script or symlinks ▼7
    7▲  is skipped
## @date 01.05.2017
## @licence GPLv3

## Package path variable
export PKGNAME="coslam_vrep";
## Base directory of the package
export DIRBASE="$HOME/.ros/workspace/src/${PKGNAME}";
export SCRIPTSDIR="$DIRBASE/scripts/shell";
export CDATE="$(date '+%d%m%y.%H%M%S')";
export LOGSDIR="${DIRBASE}/logs/scripts";
export LOG="${LOGSDIR}/${0%.sh}.$CDATE.log";
export BACKUPDIR="$DIRBASE/.backup";

## @section Script parameters
## Supress script output
QUIET="&>/dev/null";
#QUIET="";

cd "$DIRBASE";
echo "[Working directory ${PWD}]" | tee "$LOG";
echo "[Starting ${0}]" | tee -a "$LOG";
# Find all executable scripts
#find "$SCRIPTSDIR" -type f -iname '*.sh' \! -iname "${0}" -executable -print0 |  ▼30
    30▲ while IFS= read -r -d '' SCRIPT; do
find "$SCRIPTSDIR" -type f -iname '*.sh' \! -iname "${0}" -print0 | while read -r - ▼31
    31▲ d $'\0' SCRIPT; do
# These checks allow for more elaborate logging, otherwise override with find and - ▼32
    32▲ executable option
  # Check if it is a bash script and not a sed processing script
# if [ "$(head -1 "$SCRIPT")" != "#!"*"$(basename $(which bash))" ]; then
#   echo "[Incorrect shebang][Skipping $SCRIPT]" | tee -a "$LOG";
#   continue;
# fi
  if [ -x "$SCRIPT" ]; then
    SKIP=0;
    if [ -f "$SCRIPTSDIR/${0/.sh/.blacklist.ls}" ]; then
      while read -r F; do
        if [ "${SCRIPT##*/}" = "$F" ]; then
          SKIP=1; break;
        fi;
      done < "$SCRIPTSDIR/${0/.sh/.blacklist.ls}";
    fi
    if [ $SKIP -eq 1 ]; then
      echo "[Skipping $SCRIPT]" | tee -a "$LOG";
```

```
      continue;
    else
      echo -n "[Executing $SCRIPT]" | tee -a "$LOG";
      eval "$SCRIPT" $QUIET;
      echo "[Execution complete]" | tee -a "$LOG";
    fi
  fi
done
echo "[Complete ${0}]" | tee -a "$LOG";
```

Script 61: Bash compatible shell script file: record-vrep.sh

```
#!/bin/bash
## @file
##
## @institute École centrale de Nantes
## @institute Università degli studi di Genova
##
## @programme EMARO+ - 2016--2017
## @thesis Master thesis - Source code -- Script -- Bash
## @title UGV and UAV collaboration in an autonomous infrastructure scenario
##
## @author Ernest Skrzypczyk
##
## @date 28.04.17
##
## @subtitle General title -- subtitle convention
##
###
## @brief Preprocessor for Velodyne Puck (VLP16) measurements
##
## @detail Filter the simuated or raw data from Velodyne Puck (VLP16) system and publish it.
##
## @shell{$, record-vrep.sh FILENAME TOPIC1 [TOPIC2] [...]}
##
## @param bool filterNoise: Use noise filtering <!1>
## @param float filternoisethreshold: Threshold for noise filter <!1 [int]>
## @param filtervoxelgrid: Use VoxelGrid filtering <!1 [bool]>
## @param rate: Sampling frequency of the node <!25 [Hz]>
## @param samplesize: Number of elements that are averaged/subsampled <!4 [1]>
## @param sampling: Selects if the raw data should be subsampled after a certain delay or averaged over a certain period <!0 [1]>
##    - sampling 0 sets subsampling
##    - sampling !0 sets averaging with given number of samples
##
## Project github repository
##
## @see https://github.com/em-er-es/coslam/coslam_vrep
##
## @repository https://github.com/em-er-es/coslam/coslam_vrep
##
## @todo FIX DOXYGEN documentation
##
## @todo Add check for presence of V-REP and ask to restart if detected (pidof vrep )
##

# Settings
## Latest symlink filename
LATEST="latest-vrep.bag";
## Restart V-REP switch
RESTARTVREP=1;
```

```
vrep-restart-simulation() {
  if [[ "${ROS_MASTER_URI-stest}" == "stest" ]]; then echo "ROS environment not set ▼51
    51▲ "; exit 1; fi
  if [ $# -eq 0 ]; then SLEEP=3; else SLEEP=$1; fi
  rosservice call /vrep/simRosStopSimulation;
  sleep $SLEEP;
  rosservice call /vrep/simRosStartSimulation;
}

CDATE="$(date '+%d%m%y')";
ICDATE="$(date '+%y%m%d')";

if [ $# -lt 2 ]; then
echo -e "Usage:\t${0} FILENAME TOPIC1 [TOPIC2] [...]";
cat << EOF
# Syntax elements
<robot> - vehicle, robot, platform
<location> - place, area
<session> - session run ID
<run> - variations of session
<date> - preferably in "%Y%m%d" format for easier sorting, string date will be  ▼69
    69▲ automatically overridden to preferred format

# Additional syntax
|delimiters| - whitespace discouraged, because of escape characters in shell,  ▼72
    72▲ preferably single hyphen for linking elements and double hyphen for linking ▼72
    72▲  element sub-information

# Examples
Example command naming convention:
rosbag play <location>-<date>-<robot>-<session>-<run>.bag

Example 1:
rosbag play ecn-20170214-turtlebot-loop-forward.bag

Example 2:
rosbag play ecn--parkinglot-20170214-turtlebot--a-loop--01-forward--a.bag

Format of example 2 can be translated into format from example 1 using sed:
\$ echo rosbag play ecn--parkinglot-20170214-turtlebot--a-loop--01-forward--a.bag | ▼85
    85▲  sed 's/\(--[^-]*-\)/-/g;s/\(--[^-]*\.\)/./' rosbag play ecn-20170214- ▼85
    85▲ turtlebot-loop-forward.bag
EOF

else
  # Set first argument to filename
  FILE="${1/date/${ICDATE}}";
  shift;

  # Check filename extension
  if [[ "${FILE,,}" != *"bag" ]]; then FILE="${FILE%.}.bag"; fi

  # Set topics from remaining arguments
```

```
  TOPICS="${@}";

  if [ -f "$FILE" ]; then INPUTFLAG=1;
100 while [ $INPUTFLAG -eq 1 ]; do read -n 1 -p "Overwrite? (yes/No/modify)" ▼100
    100▲ KEYINPUT;
  case "$KEYINPUT" in
    y|Y)
      INPUTFLAG=0;
      continue;
105 ;;
    n|N)
      echo "Exiting";
      exit 1;
    ;;
110 m)
      KEYINPUT="$FILE";
      while [[ "$KEYINPUT" == "$FILE" ]] || [[ -z "$KEYINPUT" ]]; do
        echo; read -p "Provide new filename for $FILE:" KEYINPUT;
      done
115   FILE="${KEYINPUT/date/${ICDATE}}";

      # Check filename extension
      if [[ "${FILE,,}" != *"bag" ]]; then FILE="${FILE%.}.bag"; fi
      INPUTFLAG=0;
120 ;;
    *)
      echo "Input not recognized";
  esac
  unset KEYINPUT;
125 done; fi

  echo rosbag record -O "$FILE" "${TOPICS[@]}";
  read -n 1 -p "Press key to restart V-REP" KEYINPUT;
  { if [ $RESTARTVREP -eq 1 ]; then vrep-restart-simulation $SLEEP; fi; } \
130 && rosbag record -O "$FILE" ${TOPICS[@]};

  echo "# ${0} -- $CDATE -- $(hostname --long)@$(hostname -I) -- $FILE" > "${FILE ▼132
    132▲ /.bag/.txt}";
  rosbag info "$FILE" >> "${FILE/.bag/.txt}";

135 ## Update symlink
  if [ -f "$FILE" ]; then ln -sfv "$FILE" "$LATEST"; fi
fi
```

Script 62: SED script file: filter-stashed.sed

```
#!/usr/bin/sed -nf
## @file
## @author Ernest Skrzypczyk
## @brief Cleaning stream input using tag marks
## @date 01.05.2017
## @licence GPLv3

/>>>>>>> /,/<<<<<<< /{/<<<<<<< /d; />>>>>>> /d; p}
```

Script 63: SED script file: filter-upstream.sed

```
#!/usr/bin/sed -nf
## @file
## @author Ernest Skrzypczyk
## @brief Cleaning stream input using tag marks
## @date 01.05.2017
## @licence GPLv3

/>>>>>>>/,/<<<<<<<</{/<<<<<<<</d; />>>>>>>/d; p}
```

Script 64: Bash compatible shell script file: publish-posestamped.sh

```
1 #!/bin/bash
rostopic pub -r 4 -v -l -s pose geometry_msgs/PoseStamped '{header: {seq: auto,  ▼2
    2▲ stamp: now, frame_id: frame00}, pose: {position: {x: 0.5, y: 0.5, z: 0.5},  ▼2
    2▲ orientation: {x: 0, y: 0, z: 0, w: 1}}}'
```

Script 65: Bash compatible shell script file: vlp16-set-leafsize.sh

```
#!/bin/bash
if [ $# -eq 0 ]; then
  LEAFSIZE=0.01;
elif [ $# -eq 1 ]; then
  LEAFSIZE="$@";
elif [ $# -eq 3 ]; then
  LEAFSIZEX="$1";
  LEAFSIZEY="$2";
  LEAFSIZEZ="$3";
else
  echo "Usage: ${0} [Symmetrical leafsize|Leaf size x] [[Leaf size y] [Leaf size z ▼11
    11▲ ]]";
fi
PARAMPATH="/vlp16/filter/voxelgrid/leafsize";
LEAFS=("x" "y" "z");
if [ -n "$LEAFSIZE" ]; then
  for LEAF in "${LEAFS[@]}"; do rosparam set "$PARAMPATH/$LEAF" "$LEAFSIZE"; echo - ▼16
    16▲ n "$LEAF: "; rosparam get "$PARAMPATH/$LEAF"; done;
elif [ -n "$LEAFSIZEZ" ]; then
  LEAFSIZES=("$LEAFSIZEX" "$LEAFSIZEY" "$LEAFSIZEZ");
  i=0;
  for LEAF in "${LEAFS[@]}"; do rosparam set "$PARAMPATH/$LEAF" "${LEAFSIZES[$i]}"; ▼20
    20▲  echo -n "$LEAF: "; rosparam get "$PARAMPATH/$LEAF"; i=$((i+1)); done;
fi
```

Script 66: Bash compatible shell script file: clean-files.sh

```
#!/bin/bash
## @file
## @author Ernest Skrzypczyk
## @brief Cleans script files from tag marked lines
## @date 01.05.2017
## @licence GPLv3

if [ -n "$PKGNAME" ]; then
  PKGNAME="coslam_vrep";
  DIRBASE="$HOME/.ros/workspace/src/${PKGNAME}";
  SCRIPTSDIR="$DIRBASE/scripts/shell";
  CDATE="$(date '+%d%m%y.%H%M%S')";
  LOGSDIR="${DIRBASE}/logs/scripts";
  LOG="${LOGSDIR}/${0%.sh}.$CDATE.log";
  BACKUPDIR="$DIRBASE/.backup";
fi

FPS=(".c" ".cpp" ".h" ".hpp" ".lua");
unset FPO;
for FP in ${FPS[@]}; do FPO="-iname *$FP ${FPO+-o} $FPO"; done

cd "$DIRBASE";
if [ ! -d "$BACKUPDIR" ]; then mkdir "$BACKUPDIR"; fi
find "$SCRIPTSDIR" -type f $FPO \! -iname '*.stripped.*' -print0 | while IFS= read -r -d '' SCRIPT; do
  echo "[Processing $SCRIPT]" | tee -a "$LOG";
  cp -nv "$SCRIPT" "$BACKUPDIR/$SCRIPT.$CDATE";
  EXTENSION="${SCRIPT##*.}";
  sed '/#S#/,/#E#/d;/#D#/d' "$SCRIPT" > "${SCRIPT/.$EXTENSION/.stripped.$EXTENSION
  }";
done
```

Script 67: Bash compatible shell script file: publish-jointstate.sh

```
#!/bin/bash
rostopic pub -r 1 -v -l -s /vrep/joints/robot01/control sensor_msgs/JointState '{ ▼2
    2▲ header: {seq: auto, stamp: now, frame_id: Pioneer_p3dx_leftWheel}, name: [ ▼2
    2▲ Pioneer_p3dx_leftWheel, Pioneer_p3dx_rightWheel] , position: [0, 0], velocity ▼2
    2▲ : [1.0, 0], effort: [0, 0]}'
```

Script 68: Bash compatible shell script file: colors.sh

```
#!/bin/bash
COLORCODES=('AFFF00' '00BFFF' 'F6EB13' 'BF2900' 'EA7000' '878700');

for i in $(seq -w 0 0$((${#COLORCODES[@]}-1))); do
  eval export "COLOR$i=#${COLORCODES[$i]}";
done
```

Script 69: Bash compatible shell script file: documentation.sh

```
#!/bin/bash
## @file
## @author Ernest Skrzypczyk
## @brief Generates documentation based on defined setup in file link
## Generates documentation based on defined setup in 1[file link](file:///home/ ▼5
    5▲ emeres/.ros/workspace-170228-jade/src/coslam_vrep/configuration/doxygen.cfg)
## Generates documentation based on defined setup in 2[file link](file:// ▼6
    6▲ configuration/doxygen.cfg)
## Generates documentation based on defined setup in 3[file link](file:///../ ▼7
    7▲ configuration/doxygen.cfg)
## Generates documentation based on defined setup in 4[file link](file://../ ▼8
    8▲ configuration/doxygen.cfg)
## Generates documentation based on defined setup in 5[file link](file://./../ ▼9
    9▲ configuration/doxygen.cfg)
## Generates documentation based on defined setup in 6
## Generates documentation based on defined setup in 6[file link](file:// ▼11
    11▲ configuration/doxygen.cfg)
## Generates documentation based on defined setup in 6[file link](file:/// ▼12
    12▲ configuration/doxygen.cfg)
## ates documentation based on defined setup in @link configuration/doxygen.cfg ▼13
    13▲ test @endlink
## cumentation based on defined setup in @link configuration/doxygen.cfg ▼14
    14▲ test@endlink
## ased on defined setup in @link $OUTPUT_DIRECTORY/configuration/doxygen.cfg test ▼15
    15▲ @endlink
## on based on defined setup in @link /home/emeres/.ros/workspace/src/coslam_vrep/ ▼16
    16▲ configuration/doxygen.cfg test @endlink
## tup in @link /configuration/doxygen.cfg test @endlink
##
## @date 01.05.2017
## @licence GPLv3

if [ -n "$PKGNAME" ]; then
  PKGNAME="coslam_vrep";
  DIRBASE="$HOME/.ros/workspace/src/${PKGNAME}";
  SCRIPTSDIR="$DIRBASE/scripts/shell";
  CDATE="$(date '+%d%m%y.%H%M%S')";
  LOGSDIR="${DIRBASE}/logs/scripts";
  LOG="${LOGSDIR}/${0%.sh}.$CDATE.log";
  BACKUPDIR="$DIRBASE/.backup";
fi

cd "$DIRBASE";
CFGFILES=("configuration/doxygen.cfg");
for CFGFILE in ${CFGFILES[@]}; do
  echo "[Generating documentation for $CFGFILE]"
  doxygen "$CFGFILE";
done
```

Script 70: List file: update.blacklist.ls

```
clean-files.sh
filter-stashed.sed
filter-upstream.sed
record-vrep.sh
record-vrep-interrupt.sh
source-code-filter.sed
vlp16-set-leafsize.sh
```

Script 71: Bash compatible shell script file: post-documentation-convert-images.sh

```
#!/bin/bash
## @file
## @author Ernest Skrzypczyk
## @brief Runs scripts with executable flag
## @details This shell script can be further extended to include custom procedures.
## @details Add or modify scripts in ${SCRIPTSDIR} for additional processing.
## @note Pay attention to file names, since the filename of this script or symlinks ▼7
    7▲ is skipped
## @date 01.05.2017
## @licence GPLv3

if [ -n "$PKGNAME" ]; then
  PKGNAME="coslam_vrep";
  DIRBASE="$HOME/.ros/workspace/src/${PKGNAME}";
  SCRIPTSDIR="$DIRBASE/scripts/shell";
  CDATE="$(date '+%d%m%y.%H%M%S')";
  LOGSDIR="${DIRBASE}/logs/scripts";
  LOG="${LOGSDIR}/${0%.sh}.$CDATE.log";
  BACKUPDIR="$DIRBASE/.backup";
fi

## @section Script parameters
## Supress script output
QUIET="&>/dev/null";
#QUIET="";

cd "$DIRBASE/documentation";
DIR="html";
TFILE="$DIR/nav_f.png";

## Convert files to grayscale negation if detected image values are high, ergo ▼30
    30▲ image is light
if [ $(identify -format "%[fx:int(100*mean)]\n" "$TFILE") -gt 50 ]; then
  for F in "$DIR/"*.png; do
    convert -negate -colorspace gray "$F" "$F";
  done
fi
```

Script 72: Bash compatible shell script file: publish-clock.sh

```
1 #!/bin/bash
if [ $# -eq 0 ]; then RATE=10; else RATE=$1; fi
rostopic pub -r $RATE -v -s /clock rosgraph_msgs/Clock '{clock: now}'
```

Script 73: SED script file: source-code-filter.sed

```
#!/bin/sed -f
## @file
## @author Ernest Skrzypczyk
## @brief Cleaning stream input using tag marks
## @date 01.05.2017
## @licence GPLv3

/#S#/,/#E#/d
/#D#/d
s@///\ @@g
```

Script 74: Bash compatible shell script file: vrep-restart-simulation.sh

```
#!/bin/bash
if [ $# -eq 0 ]; then SLEEP=3; else SLEEP=$1; fi

if (pidof vrep &>/dev/null); then
  rosservice call /vrep/simRosStopSimulation;
  sleep "$SLEEP";
  rosservice call /vrep/simRosStartSimulation;
else
  echo "V-REP not running";
fi
```

Script 75: Bash compatible shell script file: post-documentation-copy-stylesheets.sh

```
#!/bin/bash
## @file
## @author Ernest Skrzypczyk
## @brief Runs scripts with executable flag
## @details This shell script can be further extended to include custom procedures.
## @details Add or modify scripts in ${SCRIPTSDIR} for additional processing.
## @note Pay attention to file names, since the filename of this script or symlinks ▼7
    7▲ is skipped
## @date 01.05.2017
## @licence GPLv3

if [ -n "$PKGNAME" ]; then
  PKGNAME="coslam_vrep";
  DIRBASE="$HOME/.ros/workspace/src/${PKGNAME}";
  SCRIPTSDIR="$DIRBASE/scripts/shell";
  CDATE="$(date '+%d%m%y.%H%M%S')";
  LOGSDIR="${DIRBASE}/logs/scripts";
  LOG="${LOGSDIR}/${0%.sh}.$CDATE.log";
  BACKUPDIR="$DIRBASE/.backup";
fi

## @section Script parameters
## Supress script output
QUIET="&>/dev/null";
#QUIET="";

cd "$DIRBASE/documentation";
DIR="html";
TFILE="$DIR/nav_f.png";

## Convert files to grayscale negation if detected image values are high, ergo ▼30
    30▲ image is light
if [ $(identify -format "%[fx:int(100*mean)]\n" "$TFILE") -gt 50 ]; then
  for F in "$DIR/"*.png; do
    convert -negate -colorspace gray "$F" "$F";
  done
fi
```

Script 76: Bash compatible shell script file: record-vrep-interrupt.sh

```
#!/bin/bash
## @file
##
## @institute École centrale de Nantes
## @institute Università degli studi di Genova
##
## @programme EMARO+ - 2016--2017
## @thesis Master thesis - Source code -- Script -- Bash
## @title UGV and UAV collaboration in an autonomous infrastructure scenario
##
## @author Ernest Skrzypczyk
##
## @date 28.04.17
##
## @subtitle General title -- subtitle convention
##
###
## @brief Preprocessor for Velodyne Puck (VLP16) measurements
##
## @detail Filter the simuated or raw data from Velodyne Puck (VLP16) system and ▼20
    20▲ publish it.
##
## @shell{$, record-vrep-interrupt.sh}
##
## Project github repository
##
## @see https://github.com/em-er-es/coslam/coslam_vrep
##
## @repository https://github.com/em-er-es/coslam/coslam_vrep
##
## @todo FIX DOXYGEN documentation
##
## @todo Add check for presence of V-REP and ask to restart if detected (pidof vrep ▼32
    32▲ )
##

# Settings

while :; do
  # Monitor V-REP simulation state, exit once simulation halt detected by gently ▼38
    38▲ interrupting rosbag record
  rostopic echo -n 1 "/vrep/info" | grep -A 1 simulatorState | tail -1 | grep -c 0 ▼39
    39▲ &>/dev/null && { echo stopped; kill -INT $(pidof /opt/ros/jade/lib/rosbag/ ▼39
    39▲ record); break; };
done
```

Script 77: Bash compatible shell script file: alias-vrep-restart.sh

```
#!/bin/bash
ALIAS="vrep-restart-simulation";
if [ $# -eq 0 ]; then SLEEP=3; else SLEEP=$1; fi

if (alias "$ALIAS" &>/dev/null); then
  echo "$ALIAS already defined";
else
  alias "$ALIAS"='rosservice call /vrep/simRosStopSimulation; sleep '"$SLEEP"'; ▼8
    8▲ rosservice call /vrep/simRosStartSimulation;';
fi
alias "$ALIAS";
```

Script 78: Lua script file: robot01.3.3.2.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # *
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  -- Counters
  counter = 0
  -- Prefixes
  rosParamPrefix = '/init/'
  posePrefix = 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼39
    39▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
-- TODO which one is it? Ros or RosInterface ? #D#
--    if (moduleName == 'RosInterface') then
    if (moduleName == 'Ros') then
      pluginNotFound = false
    end
    index = index + 1
  end

  if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
```

```
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼52
    52▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼52
    52▲ {0.5,0,0,1,1,1})
  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    object01Handle = simGetObjectHandle('object01') -- Cylinder
    object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼67
    67▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Robot 01
    -- Tranforms
    tfLinks = {}
    for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼74
    74▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

    -- Frames
    worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
    camera01Pub = simExtROS_enablePublisher('image01', 1, ▼83
    83▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    camera02Pub = simExtROS_enablePublisher('image02', 1, ▼84
    84▲ simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher('/joints/robot01/state', 1, ▼86
    86▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')

    -- Init subscribers
    -- robot01StatesSub = simExtROS_enableSubscriber('/joints/robot01/control', 1, ▼89
    89▲ simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
```

```
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

  if isint(counter / 100) then
    simAddStatusbarMessage('Counter: ' .. counter)
    -- if ((counter + 100) % 200 == 0) then
    if (counter % 200 == 0) then
      velocity = 2.0
    else
      velocity = -2.0
    end
```

```
150     -- Get joint handles
        jointHandles = {}
        -- Pioneer_p3dx_leftMotor
        for i, m in ipairs{'left', 'right'} do
          -- Motor velocity for fixed wheels
155       joint = 'Pioneer_p3dx_' .. m .. 'Motor'
          jointHandles[joint] = simGetObjectHandle(joint)
          simSetJointTargetVelocity(jointHandles[joint], velocity)
        end
        simAddStatusbarMessage('Velocity: ' .. velocity)
160   end

      -- if isint(counter / 10) then
        -- simAddStatusbarMessage('Point cloud: ' .. ptCloud)
      -- end
165
      -- Update counter
      counter = counter + 1
    end

170 -- # Clean-up
    if (sim_call_type==sim_childscriptcall_cleanup) then
      if not pluginNotFound then
        -- Following not really needed in a simulation script (i.e. automatically shut ▼173
        173▲ down at simulation end):
        -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
175   end
    end
```

Script 79: Lua script file: vlp16.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼22
    22▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4  ▼40
    40▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 =  ▼41
    41▲ pointCloud2 messages;
  -- Counters
  counter = 0

  -- Handles
  -- Frames
  worldHandle = simGetObjectHandle('frame00')
  -- robot01Handle = simGetObjectHandle('robot01')
  -- vlp16Handle = simGetObjectHandle('vlp16')
```

```
  vlp16Handle = simGetObjectHandle('vpl16')

  -- Tranform (TF)
  tfLinks = {}
  -- for i,link in ipairs {'vlp16'} do
  for i, link in ipairs {'vpl16'} do
    tfLinks[link] = simGetObjectHandle(link)
  end

  -- Base TF
  tf = {
    header = {
      stamp = 0,
      frame_id = 'frame00'
    },
    child_frame_id = '',
    transform = {
      -- ROS has definition x = front y = side z = up
      translation = {x = 0, y = 0, z = 0}, -- V-rep
      rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
    }
  }


  -- VLP16
  pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
  visionSensorHandles = {}
  for i = 1, 4, 1 do
    visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
  end

  -- Setting VLP16 parameters
  frequency = 20 -- Default model 5 Hz
  -- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only ▼83
    83▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼83
    83▲  (8)=displayed points are emissive
  options = 1 -- No display
  pointSize = 2
  coloringCloseAndFarDistance = {1, 4}
  displayScaling = 0.999 -- so that points do not appear to disappear in objects

  -- Reading VLP16 sensor data
  vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles, ▼90
    90▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼90
    90▲  pointCloudHandle)


  -- Init publishers and subscribers
  -- Init publishers
-- #S#
  -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1, ▼96
    96▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
  -- vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
```

```
  -- vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/ ▼98
    98▲ PointCloud2')
  -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼99
    99▲ VelodyneScan')
100 -- vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼100
    100▲ VelodynePacket')
  -- vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼101
    101▲ VelodyneScan')
-- #E#
  vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
105 --[[ #S#
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼106
    106▲ VelodynePacket')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
-- #E# --]]
end
110
-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
  -- Retrieve sensor data
  data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼114
    114▲ simGetSimulationTimeStep())
115 -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

  -- Debug options
  -- print('----------------')
120 -- print(bitand(debugOutput, 1))
  -- print(bitand(debugOutput, 2))
  -- print(bitand(debugOutput, 4))
  -- print('----------------')

125 -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
      simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
    else
130   pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
    end
  end

  -- Construct the point cloud from raw data: x, y and z coordinates for every ▼134
    134▲ point
135 -- Reference: From the original VPL_16 model script
  local pointCloudData = {}
  for i = 0, #data / 3 - 1, 1 do
    -- Temporary assignment to a vector d
    d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
140 -- Transformation to sensor frame
    -- No transformation results in point cloud rotated by 90 deg around z
    d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
    data[3 * i + 1] = d[1]
```

```
    data[3 * i + 2] = d[2]
    data[3 * i + 3] = d[3]
    table.insert(pointCloudData, d)
    -- print(d)
  end

  -- Display the detected points
  if displayPointCloud == 1 then
    simInsertPointsIntoPointCloud(pointCloud, 0, data)
  end

--[[ #S#
  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402, ▼164
    164▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
-- #S#
      -- table.insert(tpcl, {1, 1, 1})
      -- table.insert(tpcl, {1.0, 1.0, 1.0})
      -- table.insert(tpcl, {2, 2, 2})
-- #E#
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackFloats( ▼189
    189▲ tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼190
    190▲ simPackUInts(tpcl)}
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼191
    191▲ simPackUInts(data)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = ▼192
```

```
    192▲ simPackUInts(tdata)} -- CRASH
    -- pointCloudRawData = {stamp = simGetSimulationTimeStep(), data =  ▼193
    193▲ simPackUInts(pointCloudData)} -- CRASH
-- #S#
-- #E#

    if bitand(debugOutput, 2) == 2 then
      print('-*- VLP16 Raw START')
      print('tpcl')
      print(#tpcl)
      print('tdata')
      print(#tdata)
-- #S#
      -- print('h')
      -- print(type(h))
      -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua] SCRIPT  ▼206
    206▲ velodyneVPL...]:140: attempt to get length of global 'h' (a number value ▼206
    206▲ )
      -- print(type(tpcl))
      -- print(type(simPackUInts(tpcl)))
      -- print(type(simPackFloats(tpcl)))
      -- print(#simPackUInts(tpcl))
      -- print(type(tdata))
      -- print(tdata)
      -- for i, j in pairs(tdata) do
        -- print(i, j)
      -- end
      -- for i, j in pairs(data) do
        -- print(i, j)
      -- end
      -- print(#simPackUInts(tdata)) -- nil
      -- print('raw')
      -- print(type(pointCloudRawData.data))
      -- print(#pointCloudRawData.data)
      -- print('-*- VLP16 Raw END')
-- #E#
    end
    simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
    rmcounter = rmcounter + 1
  end
  if bitand(debugOutput, 2) == 2 then
    print('rmcounter')
    print(rmcounter)
    print('-*- VLP16 Raw END')
  end
-- #S#
-- #E#
-- #E# --]]

  -- Publish point cloud

  -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
  -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼241
```

```
  241▲ Software/V-REP/robot.lua
-- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
local pointCloudData = {}
--[[
-- The frame of VLP16 shifts all the points to that frame
-- If frame_id set to sensor frame in current configuration results in point  ▼246
  246▲ cloud shifted to the sensor position
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼247
  247▲ frame_id = "vlp16"}
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼248
  248▲ frame_id = "vpl16"}
--]]
pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(),  ▼250
  250▲ frame_id = "frame00"}
-- Describes the channels and their layout in the binary data blob
local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
pointCloudData['fields'] = {field_x, field_y, field_z}
pointCloudData['is_bigendian'] = false
pointCloudData['is_dense'] = true
pointCloudData['data'] = simPackFloats(data) --  Works
-- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the  ▼259
  259▲ data field being of type uint8
-- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼260
  260▲ and it differs in number of received points
--[[
-- Theoretically this should work, but the bytes offset still drop the messages
pointCloudData['height'] = 16 -- Ordered, 16 scan lines
pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData. ▼264
  264▲ height)
pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼265
  265▲ width / pointCloudData.height)
--]]
-- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
pointCloudData['height'] = 1 -- Unordered
-- Effectively number of cloud points, each having 3 coordinates
pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
-- Length of a point in bytes
pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼272
  272▲ width)
-- Length of a row in bytes
pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼274
  274▲ point_step']

if bitand(debugOutput, 4) == 1 then
  print('-- VLP16 PointCloud2 START')
  print(#pointCloudData.data)
  print(pointCloudData.width * pointCloudData.point_step)
  print(pointCloudData.point_step)
  print(#pointCloudData.data - math.floor(#pointCloudData.data / pointCloudData. ▼281
  281▲ width) * pointCloudData.width)
  print(#pointCloudData.data - pointCloudData.width * pointCloudData.point_step)
```

```
    print(pointCloudData.width)
-- #S#
    -- print(type(pointCloudData))
    -- print(type(tdata))
-- #E#
    print('-- VLP16 PointCloud2 END')
  end

  simExtRosInterface_publish(vlp16Pub, pointCloudData)

  -- Send transforms
  tf.header.stamp = simGetSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      position = simGetObjectPosition(handle, worldHandle)
      orientation = simGetObjectQuaternion(handle, worldHandle)
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = simGetObjectPosition(handle, vlp16Handle)
      orientation = simGetObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    simAddStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
```

```
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- simAddStatusbarMessage('Point cloud: ' .. data)
    -- simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 80: URL file: custom-messages-lua.url

```
1 http://www.forum.coppeliarobotics.com/viewtopic.php?f=9&t=6330
```

Script 81: Lua script file: vrep-publish.lua

```
-- Functions
-- Callback for desired velocity
function cmd_cb(msg)
  -- Reads desired linear and angular velocities
  if(table.getn(msg.name) == table.getn(msg.position)) then
    for i, joint in ipairs(msg.name) do
      simSetJointTargetVelocity(jointHandles[joint], msg.position[i])
    end
  end
end

-- Callback for current path computation
function path_cb(msg)
  current_path = pathHandles['Path' .. msg.data]
  current_path_pose = simGetObjectPosition(current_path, -1)
end

-- Callback for current pose to path projection
function pose_cb(pose)
dist = simGetClosestPositionOnPath(current_path, {pose.x-current_path_pose[1], pose.y-current_path_pose[2], 0}) + delta
  --simAddStatusbarMessage('delta: ' .. delta)
  if dist > 1 or dist < 0 then
    delta = -delta
  end
  xyz = simGetPositionOnPath (current_path, dist)
  rpy = simGetOrientationOnPath(current_path, dist)
  simExtRosInterface_publish(posePub, {x=xyz[1], y=xyz[2], theta=rpy[3]})
  simSetObjectPosition(proj, -1, xyz)
  --simAddStatusbarMessage('on path @ (' .. xyz[1] .. ', ' .. xyz[2] .. ', ' .. rpy[3] .. ')')

  -- Put the robot there
  simSetObjectPosition(req, -1, {pose.x, pose.y, 4})
end

-- Function for TF publications
function getTransformStamped(objHandle, name, relTo, relToName)
  t = simGetSystemTime()
  p = simGetObjectPosition(objHandle, relTo)
  o = simGetObjectQuaternion(objHandle, relTo)
  return {
    header={
      stamp = t,
      frame_id = relToName
    },
    child_frame_id = name,
    transform = {
      -- ROS has definition x=front y=side z=up
      translation = {x = p[1], y = p[2], z = p[3]}, --V-rep
      rotation = {x = o[1], y = o[2], z = o[3], w = o[4]} --V-rep
    }
```

```
  }
end

-- Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- Simulation
-- Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  counter = 0
    local moduleName = 0
    local moduleVersion = 0
    local index = 0
    local pluginNotFound = true
    poseprefix = 'pose/'
    -- poseprefix = ''
    -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼70
   70▲ dylib):
    while moduleName do
        moduleName,moduleVersion = simGetModuleName(index)
--        if (moduleName == 'RosInterface') then
        if (moduleName == 'Ros') then
            pluginNotFound = false
        end
        index = index + 1
    end

  -- Conversion deg / rad
  d2r = math.pi / 180.
  r2d = 180. / math.pi

  -- Paths
  delta = 0.01
  delta_sign = 1

    if (pluginNotFound) then

        -- Display an error message if the plugin was not found:
        simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will  ▼91
   91▲ not run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼91
   91▲ {0.5,0,0,1,1,1})

    else

        -- Handles
        -- Objects
        robot01Handle = simGetObjectHandle('Pioneer_p3dx')
        camera01Handle = simGetObjectHandle('camera01')
        object01Handle = simGetObjectHandle('object01') -- Cylinder
        object02Handle = simGetObjectHandle('object02') -- Cuboid
```

```
--        visionSensorHandle = simGetObjectHandle('camera01')
    -- Frames
    worldHandle = simGetObjectHandle('frame00')

        -- Init publishers and subscribers

        -- Init publishers
        camera01Pub = simExtROS_enablePublisher('image', 1,  ▼108
    108▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    -- Cylinder
        object01Pub = simExtROS_enablePublisher(poseprefix .. 'object01', 1,  ▼110
    110▲ simros_strmcmd_get_object_pose, object01Handle, robot01Handle, '')
        -- Cuboid
        object02Pub = simExtROS_enablePublisher(poseprefix .. 'object02', 1,  ▼112
    112▲ simros_strmcmd_get_object_pose, object02Handle, camera01Handle, '')
        JSPub = simExtROS_enablePublisher('joint_states', 512,  ▼113
    113▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')
        PCLPub = simExtROS_enablePublisher('pcl', 1,  ▼114
    114▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')

        -- Init subscribers
        robot01StatesSub = simExtROS_enableSubscriber('joint_setstates', 1,  ▼117
    117▲ simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

  -- PASTE

--[[
    req = simGetObjectHandle('pose_request')
    proj = simGetObjectHandle('pose_response')
    pathHandles = {}
    for index=1,1 do
      pathHandles['Path' .. index] = simGetObjectHandle('Path' .. index)
      simAddStatusbarMessage('Found path #' .. index)
    end

    -- Some handles for TF
    tfLinks = {}
    for i,link in ipairs {'base_link', 'laser_link', 'alfs_mesh', 'wheelFL', ' ▼132
    132▲ wheelFR', 'wheelRL', 'wheelRR'} do
      tfLinks[link] = simGetObjectHandle(link)
    end
--]]
    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x=front y=side z=up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
```

```
      }
    }

--[[
    -- Publish joint states
    -- simExtROS_enablePublisher("/joint_states", 4102,  ▼152
    152▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, "")

    -- Get joint handles
    jointHandles={}
    for j,lr in ipairs{'L','R'} do
      for i,fr in ipairs{'F','R'} do
        -- Steering
        joint = 'steering' .. fr .. lr
        jointHandles[joint] = simGetObjectHandle(joint)
        simSetJointTargetVelocity(jointHandles[joint],0)
      end
      -- Motor for F wheels
      joint = 'motorF' .. lr
      jointHandles[joint] = simGetObjectHandle(joint)
      simSetJointTargetVelocity(jointHandles[joint],0)
    end

    -- Joint subscriber
    velocitySub = simExtRosInterface_subscribe('/joint_setpoint', 'sensor_msgs/ ▼170
    170▲ JointState', 'cmd_cb')

    -- Advertise  pseudo-service = subscribe and publish topic
    current_path = pathHandles['Path1']
    current_path_pose = simGetObjectPosition(current_path, -1)
    pathSub = simExtRosInterface_subscribe('path_nb', 'std_msgs/Int32', 'path_cb')
    poseSub = simExtRosInterface_subscribe('pose_request', 'geometry_msgs/Pose2D', ▼176
    176▲  'pose_cb')
    posePub = simExtRosInterface_advertise('pose_response', 'geometry_msgs/Pose2D' ▼177
    177▲ )

--]]

    -- Publish robot twist
    twistPub = simExtRosInterface_advertise('twist', 'geometry_msgs/Twist')

    -- Simulation time
    clockPub = simExtRosInterface_advertise('/clock', 'rosgraph_msgs/Clock')

    end
end

-- Runtime
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Update counter
  counter = counter + 1
  if isint(counter / 100) then
```

```
    simAddStatusbarMessage('Counter: ' .. counter)
  end

  -- Publish simulation time
  simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})
--[[
  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    if (link == 'base_link') then
      tf.header.frame_id = 'odom'
      -- Get pose
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'base_link'
      -- Get pose
      p = simGetObjectPosition(handle, robotHandle)
      o = simGetObjectQuaternion(handle, robotHandle)
    end

    -- Update TF
    tf.child_frame_id=link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end
]]
  -- Publish twist
  -- o = simGetObjectOrientation(robotHandle, worldHandle)
  -- v, w = simGetObjectVelocity(robotHandle)
  -- c = math.cos(o[3])
  -- s = math.sin(o[3])
  -- simExtRosInterface_publish(twistPub, {linear =
                    -- {x = c * v[1] + s * v[2],
                     -- y = c * v[2] - s * v[1],
                     -- z = v[3]},
  -- angular={x = w[1], y = w[2], z = w[3]}})

end

-- Clean-up
if (sim_call_type==sim_childscriptcall_cleanup) then
  if not pluginNotFound then
    -- Following not really needed in a simulation script (i.e. automatically shut ▼246
    246▲ down at simulation end):
    simExtRosInterface_shutdownSubscriber(robot01StatesSub)
```

```
  end
end
```

Script 82: Lua script file: bitwise-operations.lua

```lua
--!-----------------------------
--! @file bitwise-operations.lua
--! @brief Bitwise operations
--!
--! Reference:
--! @see http://stackoverflow.com/questions/5977654/lua-bitwise-logical-operations
local function BitOR(a, b)
  local p, c = 1, 0
  while a + b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 0 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end

local function BitNOT(n)
  local p, c = 1, 0
  while n > 0 do
    local r = n % 2
    if r < 1 then c = c + p end
    n, p = (n - r) / 2, p * 2
  end
  return c
end

local function BitAND(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end

local function BitXOR(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra ~= rb then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  if a < b then a = b end
  while a > 0 do
    local ra = a % 2
    if ra > 0 then c = c + p end
    a, p = (a - ra) / 2, p * 2
  end
  return c
end
```

Script 83: Lua script file: robot01.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # *
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  -- Counters
  counter = 0
  -- Prefixes
  rosParamPrefix = '/init/'
  posePrefix = 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼39
    39▲ dylib):
  while moduleName do
    moduleName, moduleVersion = simGetModuleName(index)
-- TODO which one is it? Ros or RosInterface ? #D#
--    if (moduleName == 'RosInterface') then
    if (moduleName == 'Ros') then
      pluginNotFound = false
    end
    index = index + 1
  end

  if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
```

```
    simDisplayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not  ▼52
    52▲ run properly', sim_dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼52
    52▲ {0.5,0,0,1,1,1})
  else

    -- Handles
    -- Objects
    robot01Handle = simGetObjectHandle('robot01')
    camera01Handle = simGetObjectHandle('camera01') -- Left
    camera02Handle = simGetObjectHandle('camera02') -- Right
    object01Handle = simGetObjectHandle('object01') -- Cylinder
    object02Handle = simGetObjectHandle('object02') -- Cuboid

    -- Robot specific
    -- Robot 01
    -- Wheels
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼67
    67▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
      robot01JointHandles[link] = simGetObjectHandle(link)
    end

    -- Robot 01
    -- Tranforms
    tfLinks = {}
    for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼74
    74▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link'} do
      tfLinks[link] = simGetObjectHandle(link)
    end

    -- Frames
    worldHandle = simGetObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
    camera01Pub = simExtROS_enablePublisher('image01', 1, ▼83
    83▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    camera02Pub = simExtROS_enablePublisher('image02', 1, ▼84
    84▲ simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher('/joints/robot01/state', 1, ▼86
    86▲ simros_strmcmd_get_joint_state, sim_handle_all, -1, '')

    -- Init subscribers
    -- robot01StatesSub = simExtROS_enableSubscriber('/joints/robot01/control', 1, ▼89
    89▲ simros_strmcmd_set_joint_state, sim_handle_all, -1, '')

    -- Base TF
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
```

```
      child_frame_id = '',
      transform = {
        -- ROS has definition x = front y = side z = up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }

  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim_childscriptcall_actuation) then

  -- Publish simulation time
  -- simExtRosInterface_publish(clockPub, {clock = simGetSimulationTime()})

  -- Send transforms
  tf.header.stamp = simGetSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      p = simGetObjectPosition(handle, worldHandle)
      o = simGetObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'robot01'
      -- Get pose relative to robot
      p = simGetObjectPosition(handle, robot01Handle)
      o = simGetObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

  if isint(counter / 100) then
    simAddStatusbarMessage('Counter: ' .. counter)
    -- if ((counter + 100) % 200 == 0) then
    if (counter % 200 == 0) then
      velocity = 2.0
    else
      velocity = -2.0
    end
```

```
150     -- Get joint handles
        jointHandles = {}
        -- Pioneer_p3dx_leftMotor
        for i, m in ipairs{'left', 'right'} do
          -- Motor velocity for fixed wheels
155       joint = 'Pioneer_p3dx_' .. m .. 'Motor'
          jointHandles[joint] = simGetObjectHandle(joint)
          simSetJointTargetVelocity(jointHandles[joint], velocity)
        end
        simAddStatusbarMessage('Velocity: ' .. velocity)
160   end

      -- if isint(counter / 10) then
        -- simAddStatusbarMessage('Point cloud: ' .. ptCloud)
      -- end
165
      -- Update counter
      counter = counter + 1
    end

170 -- # Clean-up
    if (sim_call_type==sim_childscriptcall_cleanup) then
      if not pluginNotFound then
        -- Following not really needed in a simulation script (i.e. automatically shut ▼173
        173▲ down at simulation end):
        -- simExtRosInterface_shutdownSubscriber(robot01StatesSub)
175   end
    end
```

Script 84: URL file: bitwise-operations.url

```
1 http://stackoverflow.com/questions/5977654/lua-bitwise-logical-operations
```

Script 85: Lua script file: robot01.3.4.0.lua

```
-- ### Ã cole centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Robot01 -- Pioneer P3DX
-- # Robot script
-- # Notes:
-- # *
-- ###


-- ## Functions
-- # Callback for desired velocity
function controlCallback(msg)
  -- Reads desired linear and angular velocities
  if(table.getn(msg.name) == table.getn(msg.position)) then
    for i, joint in ipairs(msg.name) do
      sim.setJointTargetVelocity(jointHandles[joint], msg.position[i])
    end
  end
end

-- # Callback for current path computation
--[[
function pathCallback(msg)
  pathCurrent = pathHandles['Path' .. msg.data]
  pathCurrent_pose = sim.getObjectPosition(pathCurrent, -1)
end
--]]

-- # Callback for current pose to path projection
function poseCallback(pose)
dist = sim.getClosestPositionOnPath(pathCurrent, {pose.x-pathCurrent_pose[1], pose. ▼35
    35▲ y-pathCurrent_pose[2], 0}) + delta
  --sim.addStatusbarMessage('delta: ' .. delta)
  if dist > 1 or dist < 0 then
    delta = -delta
  end
  xyz = sim.getPositionOnPath (pathCurrent, dist)
  rpy = sim.getOrientationOnPath(pathCurrent, dist)
  simExtRosInterface_publish(posePub, {x = xyz[1], y = xyz[2], theta = rpy[3]})
  sim.setObjectPosition(proj, -1, xyz)
  --sim.addStatusbarMessage('on path @ (' .. xyz[1] .. ', ' .. xyz[2] .. ', ' ..  ▼44
    44▲ rpy[3] .. ')')

  -- Put the robot there
  sim.setObjectPosition(req, -1, {pose.x, pose.y, 4})
end

-- # Function for TF publications
```

```
function getTransformStamped(objHandle, name, relTo, relToName)
  t = sim.getSystemTime()
  p = sim.getObjectPosition(objHandle, relTo)
  o = sim.getObjectQuaternion(objHandle, relTo)
  return {
    header={
      stamp = t,
      frame_id = relToName
    },
    child_frame_id = name,
    transform = {
      -- ROS has definition x=front y=side z=up
      translation = {x = p[1], y = p[2], z = p[3]}, --V-rep
      rotation = {x = o[1], y = o[2], z = o[3], w = o[4]} --V-rep
    }
  }
end

-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end



-- ## Simulation
-- # Initialization
if (sim_call_type == sim.childscriptcall_initialization) then
  -- Parameters
  -- Settings
  debugOutput = 0 -- 0 = off; +1 = global messages; +2 =  messages; +4 =  messages;
  -- Counters
  counter = 0
  -- Prefixes
  rosParamPrefix = '/init/'
  posePrefix = 'pose/'
  -- poseprefix = ''

  -- Simulation and plugin specific parameters
  local moduleName = 0
  local moduleVersion = 0
  local index = 0
  local pluginNotFound = true
  -- Check if the required plugin is there (libv_repExtRos.so or libv_repExtRos. ▼93
   93▲ dylib):
  while moduleName do
    moduleName, moduleVersion = sim.getModuleName(index)
-- TODO which one is it? Ros or RosInterface ? #D#
--    if (moduleName == 'RosInterface') then
    if (moduleName == 'Ros') then
      pluginNotFound = false
    end
    index = index + 1
  end
```

```
if (pluginNotFound) then
    -- Display an error message if the plugin was not found:
    sim.displayDialog('Error', 'The RosPlugin was not found.&&nSimulation will not ▼106
 106▲  run properly', sim.dlgstyle_ok, false, nil, {0.8,0,0,0,0,0}, ▼106
 106▲ {0.5,0,0,1,1,1})
else

    -- Handles
    -- Objects
    robot01Handle = sim.getObjectHandle('robot01')
    camera01Handle = sim.getObjectHandle('camera01') -- Left
    camera02Handle = sim.getObjectHandle('camera02') -- Right
    object01Handle = sim.getObjectHandle('object01') -- Cylinder
    object02Handle = sim.getObjectHandle('object02') -- Cuboid

    -- Robot specific
    robot01JointHandles = {}
    for i, link in ipairs {'Pioneer_p3dx_leftWheel', 'Pioneer_p3dx_rightWheel', ' ▼119
 119▲ Pioneer_p3dx_caster_freeJoint1', 'Pioneer_p3dx_caster_freeJoint2'} do
        robot01JointHandles[link] = sim.getObjectHandle(link)
    end

    -- Frames
    worldHandle = sim.getObjectHandle('frame00')

    -- Init publishers and subscribers
    -- Init publishers
    camera01Pub = simExtROS_enablePublisher('image01', 1, ▼128
 128▲ simros_strmcmd_get_vision_sensor_image, camera01Handle, 0, '')
    camera02Pub = simExtROS_enablePublisher('image02', 1, ▼129
 129▲ simros_strmcmd_get_vision_sensor_image, camera02Handle, 0, '')
    -- Cylinder
    -- object01Pub = simExtROS_enablePublisher(poseprefix .. 'object01', 1, ▼131
 131▲ simros_strmcmd_get_object_pose, object01Handle, robot01Handle, '')
    -- Cuboid
    -- object02Pub = simExtROS_enablePublisher(poseprefix .. 'object02', 1, ▼133
 133▲ simros_strmcmd_get_object_pose, object02Handle, camera01Handle, '')
    -- Robot 01
    robot01jointsPub = simExtROS_enablePublisher('/joints/robot01/state', 1, ▼135
 135▲ simros_strmcmd_get_joint_state, sim.handle_all, -1, '')
    -- robot01jointsPub = simExtROS_enablePublisher('robot01joints', 1, ▼136
 136▲ simros_strmcmd_get_joint_state, sim.handle_all, -1, '')
    -- robot01jointsPub = simExtROS_enablePublisher('robot01joints', 1, ▼137
 137▲ simros_strmcmd_get_joint_state, robot01JointHandles, -1, '')
    -- PCLPub = simExtROS_enablePublisher('pcl', 1, simros_strmcmd_get_joint_state ▼138
 138▲ , sim.handle_all, -1, '')

    -- Init subscribers
    -- robot01StatesSub = simExtROS_enableSubscriber('robot01jointscontrol', 1, ▼141
 141▲ simros_strmcmd_set_joint_state, sim.handle_all, -1, '')
    robot01StatesSub = simExtROS_enableSubscriber('/joints/robot01/control', 1, ▼142
 142▲ simros_strmcmd_set_joint_state, sim.handle_all, -1, '')
```

```
  -- PASTE

--[[
    req = sim.getObjectHandle('pose_request')
    proj = sim.getObjectHandle('pose_response')
    pathHandles = {}
    for index=1,1 do
      pathHandles['Path' .. index] = sim.getObjectHandle('Path' .. index)
      sim.addStatusbarMessage('Found path #' .. index)
    end
--]]

    -- Tranforms for robot 01
    tfLinks = {}
    for i, link in ipairs {'robot01', 'Pioneer_p3dx_leftWheel', ' ▼158
    158▲ Pioneer_p3dx_rightWheel', 'Pioneer_p3dx_caster_link'} do
      tfLinks[link] = sim.getObjectHandle(link)
    end

    -- Base TF
--[[
    tf = {}
    tf.header.stamp = 0
    tf.header.frame_id = 'frame00'
    tf.child_frame_id = ''
    tf.transform = {
        -- ROS has definition x=front y=side z=up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
--]]
-- --[[
    tf = {
      header = {
        stamp = 0,
        frame_id ='frame00'
      },
      child_frame_id = '',
      transform = {
        -- ROS has definition x=front y=side z=up
        translation = {x = 0, y = 0, z = 0},-- V-rep
        rotation={x = 0, y = 0, z = 0, w = 0} -- V-rep
      }
    }
--]]

-- --[[
    -- Publish joint states
    -- simExtROS_enablePublisher("/joint_states", 4102,  ▼191
    191▲ simros_strmcmd_get_joint_state, sim.handle_all, -1, "")

--[[
```

```
    -- Joint subscriber
    velocitySub = simROS.subscribe('/joint_setpoint', 'sensor_msgs/JointState', ' ▼195
    195▲ controlCallback')

    -- Advertise  pseudo-service = subscribe and publish topic
    pathCurrent = pathHandles['Path1']
    pathCurrent_pose = sim.getObjectPosition(pathCurrent, -1)
    pathSub = simROS.subscribe('path_nb', 'std_msgs/Int32', 'pathCallback')
    poseSub = simROS.subscribe('pose_request', 'geometry_msgs/Pose2D', ' ▼201
    201▲ poseCallback')
    posePub = simROS.advertise('pose_response', 'geometry_msgs/Pose2D')
--]]

    -- Publish robot twist
    -- twistPub = simROS.advertise('twist', 'geometry_msgs/Twist')

    -- Simulation time
    -- clockPub = simROS.advertise('/clock', 'rosgraph_msgs/Clock')
--[[
    local e, x, y, z, alpha, beta, gamma, xyz, abg
    -- Set initial parameters from external configuration files
    e, x = simROS.getParamDouble(rosParamPrefix .. 'robot01/x')
    e, y = simROS.getParamDouble(rosParamPrefix .. 'robot01/y')
    e, z = simROS.getParamDouble(rosParamPrefix .. 'robot01/z')
    e, alpha = simROS.getParamDouble(rosParamPrefix .. 'robot01/alpha')
    e, beta = simROS.getParamDouble(rosParamPrefix .. 'robot01/beta')
    e, gamma = simROS.getParamDouble(rosParamPrefix .. 'robot01/gamma')
    xyz = {x, y, z}
    abg = {alpha, beta, gamma}
    sim.setObjectPosition(robot01Handle, -1, xyz)
    sim.setObjectOrientation(robot01Handle, -1, abg)
]]
  end
end

-- # Runtime -- Actuation
if (sim_call_type == sim.childscriptcall_actuation) then

  -- Publish simulation time
  -- simROS.publish(clockPub, {clock = sim.getSimulationTime()})

  -- Send transforms
  tf.header.stamp = sim.getSimulationTime()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'Pioneer_p3dx_visible') then
    if (link == 'robot01') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      p = sim.getObjectPosition(handle, worldHandle)
      o = sim.getObjectQuaternion(handle, worldHandle)
    else
      tf.header.frame_id = 'robot01'
```

```
      -- Get pose relative to robot
      p = sim.getObjectPosition(handle, robot01Handle)
      o = sim.getObjectQuaternion(handle, robot01Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = p[1]
    tf.transform.translation.y = p[2]
    tf.transform.translation.z = p[3]
    tf.transform.rotation.x = o[1]
    tf.transform.rotation.y = o[2]
    tf.transform.rotation.z = o[3]
    tf.transform.rotation.w = o[4]
    simExtRosInterface_sendTransform(tf)
  end

--[[
  -- Publish twist
  o = sim.getObjectOrientation(robot01Handle, worldHandle)
  v, w = sim.getObjectVelocity(robot01Handle)
  c = math.cos(o[3])
  s = math.sin(o[3])
  simROS.publish(twistPub, {linear =
                    {x = c * v[1] + s * v[2],
                     y = c * v[2] - s * v[1],
                     z = v[3]},
  angular={x = w[1], y = w[2], z = w[3]}})
--]]

  if isint(counter / 100) then
    sim.addStatusbarMessage('Counter: ' .. counter)
    -- if ((counter + 100) % 200 == 0) then
    if (counter % 200 == 0) then
      velocity = 2.0
    else
      velocity = -2.0
    end
    -- Get joint handles
    jointHandles = {}
    -- Pioneer_p3dx_leftMotor
    for i, m in ipairs{'left', 'right'} do
      -- Motor velocity for fixed wheels
      joint = 'Pioneer_p3dx_' .. m .. 'Motor'
      jointHandles[joint] = sim.getObjectHandle(joint)
      sim.setJointTargetVelocity(jointHandles[joint], velocity)
    end
    sim.addStatusbarMessage('Velocity: ' .. velocity)
  end

  -- if isint(counter / 10) then
    -- sim.addStatusbarMessage('Point cloud: ' .. ptCloud)
  -- end
```

```
    -- Update counter
300   counter = counter + 1
end

-- # Clean-up
if (sim_call_type==sim.childscriptcall_cleanup) then
305   if not pluginNotFound then
        -- Following not really needed in a simulation script (i.e. automatically shut ▼306
      306▲  down at simulation end):
        -- simROS.shutdownSubscriber(robot01StatesSub)
            -- simExtRosInterface_shutdownSubscriber(robot01StateSub)
    end
310 end
```

Script 86: Lua script file: vlp16.3.3.2.lua

```
-- ### École centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼22
    22▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim_childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4  ▼40
    40▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 =  ▼41
    41▲ pointCloud2 messages;
  -- Counters
  counter = 0

  -- Handles
  -- Frames
  worldHandle = simGetObjectHandle('frame00')
  -- vlp16Handle = simGetObjectHandle('vlp16')
  vlp16Handle = simGetObjectHandle('vpl16')
```

```
50
  -- Tranform (TF)
  tfLinks = {}
  -- for i,link in ipairs {'vlp16'} do
  for i,link in ipairs {'vpl16'} do
55  tfLinks[link] = simGetObjectHandle(link)
  end

  -- Base TF
  tf = {
60  header = {
      stamp = 0,
      frame_id = 'frame00'
    },
    child_frame_id = '',
65  transform = {
      -- ROS has definition x = front y = side z = up
      translation = {x = 0, y = 0, z = 0}, -- V-rep
      rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
    }
70 }


  -- VLP16
  pointCloudHandle = simGetObjectHandle('velodyneVPL_16_ptCloud')
75 visionSensorHandles = {}
  for i = 1, 4, 1 do
    visionSensorHandles[i] = simGetObjectHandle('velodyneVPL_16_sensor' .. i)
  end

80 -- Setting VLP16 parameters
  frequency = 5 -- Default model 5 Hz
  -- options = 2 + 8 -- bit0 (1) = do not display points, bit1 (2) = display only ▼82
    82▲ current points, bit2 (4) = returned data is polar (otherwise Cartesian), ▼82
    82▲ bit3 (8) = displayed points are emissive
  options = 1 -- No display
  pointSize = 2
85 coloringCloseAndFarDistance = {1, 4}
  displayScaling = 0.999 -- so that points do not appear to disappear in objects

  -- Reading VLP16 sensor data
  vlp16SensorHandle = simExtVision_createVelodyneVPL16(visionSensorHandles, ▼89
    89▲ frequency, options, pointSize, coloringCloseAndFarDistance, displayScaling, ▼89
    89▲  pointCloudHandle)
90


  -- Init publishers and subscribers
  -- Init publishers
--[[ #S#
95 vlp16Pub = simExtROS_enablePublisher('VLP16', 1, ▼95
    95▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
  vlp16Pub = simExtRosInterface_advertise('vlp16', 'sensor_msgs/PointCloud2')
  vlp16Pub = simExtRosInterface_advertise('vpl16/ptcld2', 'sensor_msgs/PointCloud2' ▼97
```

```
    97▲ )
  vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼98
    98▲ VelodyneScan')
  vlp16RawPub = simExtRosInterface_advertise('vpl16/raw', 'velodyne_msgs/ ▼99
    99▲ VelodynePacket')
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼100
    100▲ VelodyneScan')
#E# --]]
  vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2')
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼103
    103▲ VelodynePacket')
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
end

-- # Runtime -- Sensing
if (sim_call_type == sim_childscriptcall_sensing) then
  -- Retrieve sensor data
  data = simExtVision_handleVelodyneVPL16(vlp16SensorHandle, ▼111
    111▲ simGetSimulationTimeStep())
  -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = simGetObjectMatrix(visionSensorHandles[1], -1)

  --[[ #TS#
  -- Debug options
  print('----------------')
  print(bitand(debugOutput, 1))
  print(bitand(debugOutput, 2))
  print(bitand(debugOutput, 4))
  print('----------------')
  #TE# ]]--

  -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
      simRemovePointsFromPointCloud(pointCloud, 0, nil, 0)
    else
      pointCloud = simCreatePointCloud(0.02, 20, 0, pointSize)
    end
  end

  -- Construct the point cloud from raw data: x, y and z coordinates for every ▼133
    133▲ point
  -- Reference: From the original VPL_16 model script
  local pointCloudData = {}
  for i = 0, #data / 3 - 1, 1 do
    -- Temporary assignment to a vector d
    d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
    -- Transformation to sensor frame
    -- No transformation results in point cloud rotated by 90 deg around z
    d = simMultiplyVector(vlp16SensorTransformationMatrix, d)
    data[3 * i + 1] = d[1]
    data[3 * i + 2] = d[2]
```

```
    data[3 * i + 3] = d[3]
    table.insert(pointCloudData, d)
    -- print(d) -- #D#
  end

  -- Display the detected points
  if displayPointCloud == 1 then
    simInsertPointsIntoPointCloud(pointCloud, 0, data)
  end

  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402, ▼162
    162▲ each point has 3 coordinates
--[[ #S#
    for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
#E# --]]
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
--[[ #S#
      table.insert(tpcl, {1, 1, 1})
      table.insert(tpcl, {1.0, 1.0, 1.0})
      table.insert(tpcl, {2, 2, 2})
#E# --]]
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackFloats( ▼187
    187▲ tpcl)}
--[[ #S#
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackUInts( ▼189
    189▲ tpcl)}
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackUInts( ▼190
    190▲ data)} -- CRASH
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackUInts( ▼191
    191▲ tdata)} -- CRASH
```

```
    pointCloudRawData = {stamp = simGetSimulationTimeStep(), data = simPackUInts( ▼192
    192▲ pointCloudData)} -- CRASH
#E# --]]

    if bitand(debugOutput, 2) == 2 then
      print('-*- VLP16 Raw START')
      print('tpcl')
      print(#tpcl)
      print('tdata')
      print(#tdata)
--[[ #S#
      print('h')
      print(type(h))
      print(#h) -- Lua runtime error: [string "[embScript_34496006.lua] SCRIPT  ▼204
    204▲ velodyneVPL..."]:140: attempt to get length of global 'h' (a number value ▼204
    204▲ )
      print(type(tpcl))
      print(type(simPackUInts(tpcl)))
      print(type(simPackFloats(tpcl)))
      print(#simPackUInts(tpcl))
      print(type(tdata))
      print(tdata)
      for i, j in pairs (tdata) do
        print(i, j)
      end
      for i, j in pairs (data) do
        print(i, j)
      end
      print(#simPackUInts(tdata)) -- nil
      print('raw')
      print(type(pointCloudRawData.data))
      print(#pointCloudRawData.data)
      print('-*- VLP16 Raw END')
#E# --]]
    end
    simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
    rmcounter = rmcounter + 1
  end
  if bitand(debugOutput, 2) == 2 then
    print('rmcounter')
    print(rmcounter)
    print('-*- VLP16 Raw END')
--[[ #S#
#E# --]]
  end

  -- Publish point cloud

  -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
  -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼238
    238▲ Software/V-REP/robot.lua
  -- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
  local pointCloudData = {}
```

```
  --[[
  -- The frame of VLP16 shifts all the points to that frame
  -- If frame_id set to sensor frame in current configuration results in point ▼243
    243▲ cloud shifted to the sensor position
  pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(), ▼244
    244▲ frame_id = "vlp16"}
  pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(), ▼245
    245▲ frame_id = "vpl16"}
  --]]
  pointCloudData['header'] = {seq = 0, stamp = simGetSimulationTimeStep(), ▼247
    247▲ frame_id = "frame00"}
  -- Describes the channels and their layout in the binary data blob
  local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
  local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
  local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
  pointCloudData['fields'] = {field_x, field_y, field_z}
  pointCloudData['is_bigendian'] = false
  pointCloudData['is_dense'] = true
  pointCloudData['data'] = simPackFloats(data) --  Works
  -- pointCloudData['data'] = simPackUInts(data) --  Does not work, despite the ▼256
    256▲ data field being of type uint8
  -- VLP16 has 16 row scans, however the point cloud data is not always ordered ▼257
    257▲ and it differs in number of received points
  --[[
  -- Theoretically this should work, but the bytes offset still drop the messages
  pointCloudData['height'] = 16 -- Ordered, 16 scan lines
  pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData. ▼261
    261▲ height)
  pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼262
    262▲ width / pointCloudData.height)
  --]]
  -- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
  pointCloudData['height'] = 1 -- Unordered
  -- Effectively number of cloud points, each having 3 coordinates
  pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
  -- Length of a point in bytes
  pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼269
    269▲ width)
  -- Length of a row in bytes
  pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼271
    271▲ point_step']

  if bitand(debugOutput, 4) == 1 then
    print('-- VLP16 PointCloud2 START')
    print(#pointCloudData.data)
    print(pointCloudData.width * pointCloudData.point_step)
    print(pointCloudData.point_step)
    print(#pointCloudData.data - math.floor(#pointCloudData.data / pointCloudData. ▼278
    278▲ width) * pointCloudData.width)
    print(#pointCloudData.data - pointCloudData.width * pointCloudData.point_step)
    print(pointCloudData.width)
--[[ #S#
    print(type(pointCloudData))
```

```
    print(type(tdata))
#E# --]]
    print('-- VLP16 PointCloud2 END')
  end

  simExtRosInterface_publish(vlp16Pub, pointCloudData)

  -- Send transforms
  tf.header.stamp = simGetSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      position = simGetObjectPosition(handle, worldHandle)
      orientation = simGetObjectQuaternion(handle, worldHandle)
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = simGetObjectPosition(handle, vlp16Handle)
      orientation = simGetObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    simAddStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
    print(#pointCloudData % pointCloudData['height'])
--[[ #S#
    simAddStatusbarMessage('Point cloud: ' .. data)
```

```
    simAddStatusbarMessage('Point cloud: ' .. pointCloudData)
    print('data')
    for i, j in pairs (data) do
      print(i, j)
    end
    print(#data)
    print(#pointCloudData % pointCloudData['height'])
    print('rs')
    print(pointCloudData['row_step'])
#E# --]]
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim_childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 87: Lua script file: vlp16.default.lua

```
if (sim_call_type==sim_childscriptcall_initialization) then
    visionSensorHandles={}
    for i=1,4,1 do
        visionSensorHandles[i]=simGetObjectHandle('velodyneVPL_16_sensor'..i)
    end
    ptCloudHandle=simGetObjectHandle('velodyneVPL_16_ptCloud')
    frequency=5 -- 5 Hz
    options=2+8 -- bit0 (1)=do not display points, bit1 (2)=display only current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 (8)=displayed points are emissive
    pointSize=2
    coloring_closeAndFarDistance={1,4}
    displayScaling=0.999 -- so that points do not appear to disappear in objects
    h=simExtVision_createVelodyneVPL16(visionSensorHandles,frequency,options,pointSize,coloring_closeAndFarDistance,displayScaling,ptCloudHandle)
end

if (sim_call_type==sim_childscriptcall_sensing) then
    data=simExtVision_handleVelodyneVPL16(h,simGetSimulationTimeStep())
--[[
    -- if we want to display the detected points ourselves:
    if ptCloud then
        simRemovePointsFromPointCloud(ptCloud,0,nil,0)
    else
        ptCloud=simCreatePointCloud(0.02,20,0,pointSize)
    end
    m=simGetObjectMatrix(visionSensorHandles[1],-1)
    for i=0,#data/3-1,1 do
        d={data[3*i+1],data[3*i+2],data[3*i+3]}
        d=simMultiplyVector(m,d)
        data[3*i+1]=d[1]
        data[3*i+2]=d[2]
        data[3*i+3]=d[3]
    end
    simInsertPointsIntoPointCloud(ptCloud,0,data)
--]]
end


if (sim_call_type==sim_childscriptcall_cleanup) then
    simExtVision_destroyVelodyneVPL16(h)
end
```

Script 88: Lua script file: vlp16.3.4.0.lua

```
-- ### Ã cole centrale de Nantes
-- ### EMARO+ - 2016--2017
-- ## Master thesis - Source code -- Script -- LUA
-- ## UGV and UAV collaboration in an autonomous infrastructure scenario
-- ## Ernest Skrzypczyk
-- # CoSLAM -- V-REP
-- # Scene 01 -- Sensor -- VLP16
-- # Sensor script
-- # Notes:
-- # * Lua model is referred to as VLP16 in hardware documentation
-- # * Internal functions use the name VPL16
-- ###


-- ## Functions
-- # Helper function for integer
function isint(n)
  return n == math.floor(n)
end

-- # Bitwise operations
-- Reference: http://stackoverflow.com/questions/5977654/lua-bitwise-logical- ▼22
    22▲ operations
local function bitand(a, b)
  local p, c = 1, 0
  while a > 0 and b > 0 do
    local ra, rb = a % 2, b % 2
    if ra + rb > 1 then c = c + p end
    a, b, p = (a - ra) / 2, (b - rb) / 2, p * 2
  end
  return c
end


-- ## Simulation
-- # Initialization
if (sim_call_type == sim.childscriptcall_initialization) then
  -- Parameters
  -- Settings
  displayPointCloud = 0
  -- debugOutput = 2 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4  ▼40
    40▲ = pointCloud2 messages;
  debugOutput = 0 -- 0 = off; +1 = global VPL16 messages; +2 = raw messages; +4 =  ▼41
    41▲ pointCloud2 messages;
  -- Counters
  counter = 0

  -- Handles
  -- Frames
  worldHandle = sim.getObjectHandle('frame00')
  -- vlp16Handle = sim.getObjectHandle('vlp16')
  vlp16Handle = sim.getObjectHandle('vpl16')
```

```
50
  -- Tranform (TF)
  tfLinks = {}
  -- for i,link in ipairs {'vlp16'} do
  for i, link in ipairs {'vpl16'} do
55  tfLinks[link] = sim.getObjectHandle(link)
  end

  -- Base TF
  tf = {
60  header = {
      stamp = 0,
      frame_id = 'frame00'
    },
    child_frame_id = '',
65  transform = {
      -- ROS has definition x = front y = side z = up
      translation = {x = 0, y = 0, z = 0}, -- V-rep
      rotation = {x = 0, y = 0, z = 0, w = 0} -- V-rep
    }
70 }


  -- VLP16
  pointCloudHandle = sim.getObjectHandle('velodyneVPL_16_ptCloud')
75 visionSensorHandles = {}
  for i = 1, 4, 1 do
    visionSensorHandles[i] = sim.getObjectHandle('velodyneVPL_16_sensor' .. i)
  end

80 -- Setting VLP16 parameters
  frequency = 15 -- Default model 5 Hz
  -- options = 2 + 8 -- bit0 (1)=do not display points, bit1 (2)=display only  ▼82
    82▲ current points, bit2 (4)=returned data is polar (otherwise Cartesian), bit3 ▼82
    82▲  (8)=displayed points are emissive
  options = 1 -- No display
  pointSize = 2
85 coloringCloseAndFarDistance = {1, 4}
  displayScaling = 0.999 -- so that points do not appear to disappear in objects

  -- Reading VLP16 sensor data
  vlp16SensorHandle = simVision.createVelodyneVPL16(visionSensorHandles, frequency, ▼89
    89▲  options, pointSize, coloringCloseAndFarDistance, displayScaling,  ▼89
    89▲ pointCloudHandle)
90


  -- Init publishers and subscribers
  -- Init publishers
-- #S#
95 -- vlp16Pub = simExtROS_enablePublisher('VLP16', 1,  ▼95
    95▲ simros_strmcmd_get_depth_sensor_data, pointCloudHandle, 0, '')
  -- vlp16Pub = simROS.advertise('vlp16', 'sensor_msgs/PointCloud2')
  -- vlp16Pub = simROS.advertise('vpl16/ptcld2', 'sensor_msgs/PointCloud2')
```

```
  -- vlp16RawPub = simROS.advertise('vpl16/raw', 'velodyne_msgs/VelodyneScan')
  -- vlp16RawPub = simROS.advertise('vpl16/raw', 'velodyne_msgs/VelodynePacket')
  -- vlp16RawPub = simROS.advertise('vlp16/raw', 'velodyne_msgs/VelodyneScan')
-- #E#
  vlp16Pub = simExtRosInterface_advertise('vlp16/pcl2', 'sensor_msgs/PointCloud2')
  vlp16RawPub = simExtRosInterface_advertise('vlp16/raw', 'velodyne_msgs/ ▼103
    103▲ VelodynePacket')
    simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16Pub)
  simExtRosInterface_publisherTreatUInt8ArrayAsString(vlp16RawPub)
end

-- # Runtime -- Sensing
if (sim_call_type == sim.childscriptcall_sensing) then
  -- Retrieve sensor data
  data = simVision.handleVelodyneVPL16(vlp16SensorHandle, sim. ▼111
    111▲ getSimulationTimeStep())
  -- Retrieve sensor transformation matrix
  vlp16SensorTransformationMatrix = sim.getObjectMatrix(visionSensorHandles[1], ▼113
    113▲ -1)

  -- Debug options
  -- print('----------------')
  -- print(bitand(debugOutput, 1))
  -- print(bitand(debugOutput, 2))
  -- print(bitand(debugOutput, 4))
  -- print('----------------')

  -- Display the detected points
  if displayPointCloud == 1 then
    if pointCloud then
      sim.removePointsFromPointCloud(pointCloud, 0, nil, 0)
    else
      pointCloud = sim.createPointCloud(0.02, 20, 0, pointSize)
    end
  end

  -- Construct the point cloud from raw data: x, y and z coordinates for every ▼131
    131▲ point
  -- Reference: From the original VPL_16 model script
  local pointCloudData = {}
  for i = 0, #data / 3 - 1, 1 do
    -- Temporary assignment to a vector d
    d = {data[3 * i + 1], data[3 * i + 2], data[3 * i + 3]}
    -- Transformation to sensor frame
    -- No transformation results in point cloud rotated by 90 deg around z
    d = sim.multiplyVector(vlp16SensorTransformationMatrix, d)
    data[3 * i + 1] = d[1]
    data[3 * i + 2] = d[2]
    data[3 * i + 3] = d[3]
    table.insert(pointCloudData, d)
    -- print(d)
  end
```

```
  -- Display the detected points
  if displayPointCloud == 1 then
    sim.insertPointsIntoPointCloud(pointCloud, 0, data)
  end

  -- Data for raw velodyne message
  -- Reference: http://docs.ros.org/indigo/api/velodyne_msgs/html/index-msg.html
  local pointCloudRawDataLength = math.floor(#data / 3)
  local rmcounter = 0 -- Raw message counter
  -- print(pointCloudRawDataLength) -- #D#
  while rmcounter * 402 < pointCloudRawDataLength do
    local tdata = {}
    -- Each message has a limit of 1206 bytes
    for i = rmcounter * 402 + 1, (rmcounter + 1) * 402, 1 do -- 1206 / 3 = 402,  ▼160
    160▲ each point has 3 coordinates
-- #S#
    -- for i = 1, 402, 1 do -- 1206 / 3 = 402, each point has 3 coordinates
      -- if not data[ 3 * i + 2 ] then data [ 3 * i + 2 ] = 0 end
      -- if not data[ 3 * i + 1 ] then data [ 3 * i + 1 ] = 0 end
      -- if not data[ 3 * i + 0 ] then data [ 3 * i + 0 ] = 0 end
      -- tdata[i] = {data[3 * i], data[3 * i + 1], data[3 * i + 2]}
-- #E#
      table.insert(tdata, {data[3 * i], data[3 * i + 1], data[3 * i + 2]})
    end

    local tpcl = {}
    -- table.insert(tpcl, {0, 0, 0}) -- #D#
    -- for i = 1, 402, 1 do
    for i = 1, 1, 1 do -- works forcing one entry in table
      table.insert(tpcl, {0, 0, 0, 1, 1, 1, 2, 2, 2})
-- #S#
      -- table.insert(tpcl, {1, 1, 1})
      -- table.insert(tpcl, {1.0, 1.0, 1.0})
      -- table.insert(tpcl, {2, 2, 2})
-- #E#
    end

    local pointCloudRawData = {}
    -- CRASH with multiple objects in data
    pointCloudRawData = {stamp = sim.getSimulationTimeStep(), data = sim. ▼185
    185▲ packFloatTable(tpcl)}
    -- pointCloudRawData = {stamp = sim.getSimulationTimeStep(), data = sim. ▼186
    186▲ packUInt32Table(tpcl)}
    -- pointCloudRawData = {stamp = sim.getSimulationTimeStep(), data = sim. ▼187
    187▲ packUInt32Table(data)} -- CRASH
    -- pointCloudRawData = {stamp = sim.getSimulationTimeStep(), data = sim. ▼188
    188▲ packUInt32Table(tdata)} -- CRASH
    -- pointCloudRawData = {stamp = sim.getSimulationTimeStep(), data = sim. ▼189
    189▲ packUInt32Table(pointCloudData)} -- CRASH
-- #S#
-- #E#

    if bitand(debugOutput, 2) == 2 then
```

```
        print('-*- VLP16 Raw START')
        print('tpcl')
        print(#tpcl)
        print('tdata')
        print(#tdata)
-- #S#
        -- print('h')
        -- print(type(h))
        -- print(#h) -- Lua runtime error: [string "[embScript_34496006.lua] SCRIPT ▼202
    202▲ velodyneVPL...]:140: attempt to get length of global 'h' (a number value ▼202
    202▲ )
        -- print(type(tpcl))
        -- print(type(sim.packUInt32Table(tpcl)))
        -- print(type(sim.packFloatTable(tpcl)))
        -- print(#sim.packUInt32Table(tpcl))
        -- print(type(tdata))
        -- print(tdata)
        -- for i, j in pairs (tdata) do
          -- print(i, j)
        -- end
        -- for i, j in pairs (data) do
          -- print(i, j)
        -- end
        -- print(#sim.packUInt32Table(tdata)) -- nil
        -- print('raw')
        -- print(type(pointCloudRawData.data))
        -- print(#pointCloudRawData.data)
        -- print('-*- VLP16 Raw END')
-- #E#
    end
    simExtRosInterface_publish(vlp16RawPub, pointCloudRawData)
    rmcounter = rmcounter + 1
  end
  if bitand(debugOutput, 2) == 2 then
    print('rmcounter')
    print(rmcounter)
    print('-*- VLP16 Raw END')
-- #S#
-- #E#
  end

  -- Publish point cloud

  -- Publish the IR sensors as a point cloud, this is useful for mapping in ROS
  -- Reference: https://github.com/Nurgak/Virtual-Robot-Challenge/blob/master/ ▼236
    236▲ Software/V-REP/robot.lua
  -- Reference: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
  local pointCloudData = {}
  --[[
  -- The frame of VLP16 shifts all the points to that frame
  -- If frame_id set to sensor frame in current configuration results in point ▼241
    241▲ cloud shifted to the sensor position
  pointCloudData['header'] = {seq = 0, stamp = sim.getSimulationTimeStep(), ▼242
```

```
    242▲ frame_id = "vlp16"}
  pointCloudData['header'] = {seq = 0, stamp = sim.getSimulationTimeStep(),  ▼243
    243▲ frame_id = "vpl16"}
  --]]
245 pointCloudData['header'] = {seq = 0, stamp = sim.getSimulationTimeStep(),  ▼245
    245▲ frame_id = "frame00"}
  -- Describes the channels and their layout in the binary data blob
  local field_x = {name = "x", offset = 0, datatype = 7, count = 1}
  local field_y = {name = "y", offset = 4, datatype = 7, count = 1}
  local field_z = {name = "z", offset = 8, datatype = 7, count = 1}
250 pointCloudData['fields'] = {field_x, field_y, field_z}
  pointCloudData['is_bigendian'] = false
  pointCloudData['is_dense'] = true
  pointCloudData['data'] = sim.packFloatTable(data) --  Works
  -- pointCloudData['data'] = sim.packUInt32Table(data) --  Does not work, despite ▼254
    254▲  the data field being of type uint8
255 -- VLP16 has 16 row scans, however the point cloud data is not always ordered  ▼255
    255▲ and it differs in number of received points
  --[[
  -- Theoretically this should work, but the bytes offset still drop the messages
  pointCloudData['height'] = 16 -- Ordered, 16 scan lines
  pointCloudData['width'] = math.floor(#pointCloudData.data / 3 / pointCloudData. ▼259
    259▲ height)
260 pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼260
    260▲ width / pointCloudData.height)
  --]]
  -- pointCloudData['height'] = 16 -- Ordered, 16 scan lines
  pointCloudData['height'] = 1 -- Unordered
  -- Effectively number of cloud points, each having 3 coordinates
265 pointCloudData['width'] = math.floor(#pointCloudData.data / 3)
  -- Length of a point in bytes
  pointCloudData['point_step'] = math.floor(#pointCloudData.data / pointCloudData. ▼267
    267▲ width)
  -- Length of a row in bytes
  pointCloudData['row_step'] = pointCloudData['width'] * pointCloudData[' ▼269
    269▲ point_step']
270
  if bitand(debugOutput, 4) == 1 then
    print('-- VLP16 PointCloud2 START')
    print(#pointCloudData.data)
    print(pointCloudData.width * pointCloudData.point_step)
275 print(pointCloudData.point_step)
    print(#pointCloudData.data - math.floor(#pointCloudData.data / pointCloudData. ▼276
    276▲ width) * pointCloudData.width)
    print(#pointCloudData.data - pointCloudData.width * pointCloudData.point_step)
    print(pointCloudData.width)
-- #S#
280 -- print(type(pointCloudData))
    -- print(type(tdata))
-- #E#
    print('-- VLP16 PointCloud2 END')
  end
285
```

```
  simExtRosInterface_publish(vlp16Pub, pointCloudData)

  -- Send transforms
  tf.header.stamp = sim.getSimulationTimeStep()
  for link, handle in pairs(tfLinks) do
    -- Get parent
    -- if (link == 'vlp16') then
    if (link == 'vpl16') then
      tf.header.frame_id = 'frame00'
      -- Get pose relative to worldframe
      position = sim.getObjectPosition(handle, worldHandle)
      orientation = sim.getObjectQuaternion(handle, worldHandle)
    else
      -- tf.header.frame_id = 'vlp16'
      tf.header.frame_id = 'vpl16'
      -- Get pose relative to sensor
      position = sim.getObjectPosition(handle, vlp16Handle)
      orientation = sim.getObjectQuaternion(handle, vlp16Handle)
    end

    -- Update TF
    tf.child_frame_id = link
    tf.transform.translation.x = position[1]
    tf.transform.translation.y = position[2]
    tf.transform.translation.z = position[3]
    tf.transform.rotation.x = orientation[1]
    tf.transform.rotation.y = orientation[2]
    tf.transform.rotation.z = orientation[3]
    tf.transform.rotation.w = orientation[4]
    simExtRosInterface_sendTransform(tf)
  end

  -- Update counter
  counter = counter + 1

  if bitand(debugOutput, 1) == 1 and isint(counter / 10) then
    sim.addStatusbarMessage('VLP16 counter: ' .. counter)
    print('--- VLP16 START')
    print('pcl data')
    print(#pointCloudData)
    print('w')
    print(pointCloudData['width'])
    print('h')
    print(pointCloudData['height'])
    print('r')
    print(#pointCloudData % pointCloudData['height'])
-- #S#
    -- sim.addStatusbarMessage('Point cloud: ' .. data)
    -- sim.addStatusbarMessage('Point cloud: ' .. pointCloudData)
    -- print('data')
    -- for i, j in pairs (data) do
      -- print(i, j)
    -- end
```

```
    -- print(#data)
    -- print(#pointCloudData % pointCloudData['height'])
    -- print('rs')
    -- print(pointCloudData['row_step'])
-- #E#
    print('---  VLP16 END')
  end
end

-- # Cleanup
if (sim_call_type == sim.childscriptcall_cleanup) then
  simExtVision_destroyVelodyneVPL16(vlp16SensorHandle)
end
```

Script 89: Python script file: stereo_vision.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Use
#!/usr/bin/env python2
# to force Python2 version
#
## @file
#
# @author Ernest Skrzypczyk
#
# @date 10.06.17
#
# @brief Depth estimation for stereo vision
#
# @detail Estimate depth using the epipolar constraint in a stereo vision system
#
# @cmdp{$, rosrun coslam_vrep stereo_vision}
#
# @github https://github.com/em-er-es/coslam/coslam_vrep
#
# @todo Fix DOXYGEN documentation
#
# @todo Add handling of parameter changes in a seperate class function called from ▼23
    23▲ the main loop
#
# @done Publish pose stamped using class function call
#
# @done Subscribe to both camera topics
#
# @done Publish processed and concatenated stereo image
#
# @done Estimated depth from matching features
#
# @done Publish estimated PointCloud2
#
# @todo Publish disparity image
#
# @todo Implement correctMatches() before triangulation to minimize depth ▼37
    37▲ estimation errors
#
# @todo Set used paramters on the server
#
# @todo Rearrange the whole order of code, move depth estimation to class, add more ▼41
    41▲  options of control
#
# @todo Publish on different topics
#
# @todo Use one switch for all vision output options
#
# @done Fix publishing using the correct frame identificator
#
# @reference http://wiki.ros.org/rospy_tutorials/Tutorials/ ▼49
```

```
    49▲ WritingImagePublisherSubscriber
50  #
    # @note Code written for Python2 and OpenCV3
    #

    #%% Import
55  from __future__ import print_function #Python2 to Python3 compatibility
    #Import dependencies
    ## Passing arguments
    import sys
    ## Time -- Profilling
60  import time
    #from copy import copy
    import cv2 #OpenCV
    # from matplotlib import pyplot as plt #Matplotlib
    import numpy as np #Numpy
65  # import os #OS
    # import logging
    # from pylab import *
    # import scipy as sp

70  ## Import ROS libraries
    # import roslib
    #roslib.load_manifest('stereo_vision')
    import rospy
    #from dynamic_reconfigure.server import Server as DynamicReconfigureServer # Not  ▼74
    74▲ used right now
75  import tf #Transforms

    ## Import messages for vision
    from sensor_msgs.msg import CompressedImage

80  ## Import messages for point cloud
    from sensor_msgs.msg import PointCloud2
    from geometry_msgs.msg import PoseStamped
    import sensor_msgs.point_cloud2 as pc2

85  # V-REP specific
    #from vrep_common.msg import VrepInfo

    # Import custom messages
    # Confirm the PYTHONPATH environment variable content
90

    #%% Script flags
    ## Verbose flag
    VERBOSE = 0

95  ## Debug flag
    DEBUG = False

    ## Loop counter message filter
    LCF = 100
100
```

```
## Feature detector selector
# 0 = ORB
# 1 = AKAZE
# 2 = SIFT
# 3 = SURF
selectFeatureDetectionMethod = 0

## Minimum number of matches
minMatches = 8

#%% Global flags
## Global message flag 1 -- Camera 01
flagSubscriber1 = False

## Global message flag 2 -- Camera 02
flagSubscriber2 = False

# Stereo vision specific
## Camera 01 flag
# Set upon receiving image for left/main/01 camera
flagCamera01 = False
## Camera 02 flag
# Set upon receiving image for right/secondary/02 camera
flagCamera02 = False

#%% Global variables
## Node name using console codes
NodeName = "\033[38;5;160msvde\033[0m" # Colored
# NodeName = "svde" # Monochrome

# ROS specific
## V-REP information
# TP_SVDE_I_0 = "/vrep/info"
## Subscriber topic 01
TP_SVDE_I_1 = "/vrep/stereo_vision/camera01/compressed"
## Subscriber topic 02
TP_SVDE_I_2 = "/vrep/stereo_vision/camera02/compressed"
## Publisher topic 01
TP_SVDE_O_1 = "/stereo_vision/output/compressed"
## Publisher topic 02
TP_SVDE_O_2 = "/stereo_vision/output/pcl2"
## Publisher topic 03
TP_SVDE_O_3 = "/stereo_vision/output/pose"

## Loop counter
LoopCounter = 1

## Node/visualize rate [Hz] = [fps]
rate = 25 # 25 [Hz] = 25 [fps]

## Transform listener instance
tfListener = None
```

```
## Transform wait time [s]
tfWaitTime = 1.0
# tfWaitTime = 2.0

## SimulationTime
initialSimulationTime = None
currentSimulationTime = None

## Variables for input images, detected key points and their descriptors
# @{
inputImage01 = np.array([])
inputImage02 = np.array([])
keyPoints01 = None
keyPoints02 = None
descriptors01 = np.array([])
descriptors02 = np.array([])

## Vision sensor width or resolution in X
vw = 640
# vw = 1280
# vw = 1024
## Vision sensor height or resolution in Y
vh = 480
# vh = 960
# vh = 768
## Video framerate
# @note Currently not used
vf = 30
#vf = 25

## Compression format and extension
# Lossless -- Portable Network Graphics
#compressionFormat = ("png", ".png")
# Lossy -- Joint Photographic Experts Group
compressionFormat = ("jpeg", ".jpg")

## Sort matches
sortMatches = True

## Draw features on processed images
# drawFeatures = True
drawFeatures = False

## Publish processed image
publishProcessed = False
# publishProcessed = True
# publishProcessed = 2 # Detected edges

## Publish matched images
publishMatched = False
# publishMatched = True

## Publish disparity
```

```
publishDisparity = False
# publishDisparity = True

## Publish pose
# publishPose = False
publishPose = True

## Frame identificators
# @{
robot01FrameId = "robot01"
#robot02FrameId = "robot02"
# stereoVisionFrameId = "stereo_vision"
stereoVisionFrameId = "camera01"
worldFrameId = "frame00"
# @}

#%% Functions
## Sets video camera parameters
#
# @return video parameters: width, height, framerate
def cameraSetParameters(videoInput, resx, resy, fps):
  videoInput.set(cv2.CAP_PROP_FRAME_WIDTH, int(resx))
  videoInput.set(cv2.CAP_PROP_FRAME_HEIGHT, int(resy))
  videoInput.set(cv2.CAP_PROP_FPS, float(fps))
  return str(videoInput.get(cv2.CAP_PROP_FRAME_WIDTH)), str(videoInput.get(cv2.CAP_PROP_FRAME_HEIGHT)), str(videoInput.get(cv2.CAP_PROP_FPS))

## Normalize image
#
# Uses upper bound and minimum values
#
# @return normalized image
def normalizeImage(image, upperBound = 255):
  if upperBound == -1:
    normalizedImage = (image - image.min()) * image.max() // (image.max() - image.min())
  else:
    normalizedImage = (image - image.min()) * upperBound // (image.max() - image.min())
  return normalizedImage

## Determine if variable is empty
#
# @reference https://stackoverflow.com/questions/10545385/how-to-check-if-a-variable-is-empty-in-python#10545399
#
# @return logical empty state
def empty(value):
  try:
    value = float(value)
  except ValueError:
    pass
  except TypeError:
```

```
    pass
  return bool(not value)


#%% Node

## Node class definition
#
# Process images from a known stereo vision system, estimate depth of detected ▼264
    264▲ feature points using triangulation of points in normalized coordinates, ▼264
    264▲ estimate pose from matches in sequential frames
class StereoVisionDepthEstimation:

  ## Node main function
  #
  # Initialization
  def __init__(self):
    #%% Get ROS server parameters
    if len(sys.argv) == 1 and rospy.has_param('/stereo_vision'):
      if VERBOSE > 0:
        print("Clear parameters")
      rospy.delete_param('/stereo_vision')
    # Input left
    global TP_SVDE_I_1
    if rospy.has_param('/stereo_vision/input/1'):
      TP_SVDE_I_1 = rospy.get_param('/stereo_vision/input/1', TP_SVDE_I_1)
    else:
      rospy.set_param('/stereo_vision/input/1', TP_SVDE_I_1)
    # Input right
    global TP_SVDE_I_2
    if rospy.has_param('/stereo_vision/input/2'):
      TP_SVDE_I_2 = rospy.get_param('/stereo_vision/input/2', TP_SVDE_I_2)
    else:
      rospy.set_param('/stereo_vision/input/2', TP_SVDE_I_2)
    global selectFeatureDetectionMethod
    if rospy.has_param('/stereo_vision/featuredetectionmethod'):
      selectFeatureDetectionMethod = rospy.get_param('/stereo_vision/ ▼290
    290▲ featuredetectionmethod', selectFeatureDetectionMethod)
    else:
      rospy.set_param('/stereo_vision/featuredetectionmethod', ▼292
    292▲ selectFeatureDetectionMethod)

    #%% Set publisher/subscriber
    ## Publish processed images
    self.publisher01 = rospy.Publisher(TP_SVDE_O_1, CompressedImage, queue_size = ▼296
    296▲ 1)

    ## Publish estimated depth point cloud
    self.publisher02 = rospy.Publisher(TP_SVDE_O_2, PointCloud2, queue_size = 1, ▼299
    299▲ latch = True)

    ## Publish estimated pose (position and orientation in 3D)
    self.publisher03 = rospy.Publisher(TP_SVDE_O_3, PoseStamped, queue_size = 1, ▼302
```

```
    302▲ latch = True)

    ## Subscribe to V-REP information
305 # self.subscriber00 = rospy.Subscriber(TP_SVDE_I_0, VrepInfo, self.vrepTime,  ▼305
    305▲ queue_size = 1)

    ## Subscribe to camera 01
    # @todo Add tranformation matrix to subscriberCallback arguments
    self.subscriber01 = rospy.Subscriber(TP_SVDE_I_1, CompressedImage, self. ▼309
    309▲ subscriberCallback, callback_args = (1, robot01FrameId), queue_size = 1)
310
    ## Subscribe to camera 02
    self.subscriber02 = rospy.Subscriber(TP_SVDE_I_2, CompressedImage, self. ▼312
    312▲ subscriberCallback, callback_args = (2, robot01FrameId), queue_size = 1)

    # Print topics
315 if VERBOSE > 0:
#     rospy.loginfo("[CoSLAM][%s][Init] Subscribers: %s, %s", NodeName,  ▼316
    316▲ TP_SVDE_I_1, TP_SVDE_I_2)
#     rospy.loginfo("[CoSLAM][%s][Init] Publishers: %s, %s", NodeName, TP_SVDE_O_1 ▼317
    317▲ , TP_SVDE_O_2)
      print("[CoSLAM][%s][Init] Subscribers: %s, %s" % (NodeName, TP_SVDE_I_1,  ▼318
    318▲ TP_SVDE_I_2))
      print("[CoSLAM][%s][Init] Publishers: %s, %s" % (NodeName, TP_SVDE_O_1,  ▼319
    319▲ TP_SVDE_O_2))
320
  ## Subscriber callback for V-REP
  #
  # @detail Get simulation time
  def vrepTime(self, inputMessage):
325   global initialSimulationTime
    global currentSimulationTime
    if initialSimulationTime == None:
      initialSimulationTime = inputMessage.simulationTime
      currentSimulationTime = inputMessage.simulationTime
330 else:
      currentSimulationTime = inputMessage.simulationTime
    return currentSimulationTime

  ## Subscriber callback for vision
335 #
  # @detail Process images by detecting feautures and publish processed image
  def subscriberCallback(self, inputMessage, arguments):
    #%% Assign arguments
    ## Convert ROS message to NumPy image array
340 inputImageArray = np.fromstring(inputMessage.data, np.uint8)

    ## Assign transformation
    cameraId = arguments[0]
    cameraIdFrame = arguments[1]
345
    #%% Process data
    if VERBOSE > 1:
```

```
      print("Camera %s detected for unit %s" % (cameraId, cameraIdFrame))

    ## Read input image
    # @see http://docs.opencv.org/3.0-beta/modules/imgcodecs/doc/ ▼351
    351▲ reading_and_writing_images.html#imread
    inputImage = cv2.imdecode(inputImageArray, cv2.COLOR_BGR2GRAY)

    ## Set feature detection method
    #
    # @warning Not compatible with OpenCV3
    # Prefix possible values: "","Grid","Pyramid"
    # Suffix possible values: "FAST","GFTT","HARRIS","MSER","ORB","SIFT","STAR"," ▼358
    358▲ SURF"
    global selectFeatureDetectionMethod
    _selectFeatureDetectionMethod = rospy.get_param('/stereo_vision/ ▼360
    360▲ featuredetectionmethod', selectFeatureDetectionMethod)
    # Check if feature detection method was changed
    if selectFeatureDetectionMethod != _selectFeatureDetectionMethod:
      # Clear stereo vision buffer data
      self.clearStereoVisionBuffer()
      # Update feature detection method
      selectFeatureDetectionMethod = _selectFeatureDetectionMethod
      return -1

    ## Create feature detector
    if selectFeatureDetectionMethod == 1:
      featureDetectionMethod = "AKAZE"
      featureDetector = cv2.AKAZE_create()
    elif selectFeatureDetectionMethod == 2:
      featureDetectionMethod = "SIFT"
      featureDetector = cv2.xfeatures2d.SIFT_create()
    elif selectFeatureDetectionMethod == 3:
      featureDetectionMethod = "SURF"
      featureDetector = cv2.xfeatures2d.SURF_create()
    else: # Default == 0
      featureDetectionMethod = "ORB"
      featureDetector = cv2.ORB_create()

    if VERBOSE > 0:
      timepoint00 = time.time()

    # @todo Store key points in accessible objects
    # keyPoints, descriptors = featureDetector.detectAndCompute(inputImage, None)
    # keyPoints[:, cameraId], descriptors[:, cameraId] = featureDetector. ▼388
    388▲ detectAndCompute(inputImage, None)
    # keyPoints[:][cameraId], descriptors[:][cameraId] = featureDetector. ▼389
    389▲ detectAndCompute(inputImage, None)
    # keyPoints[cameraId], descriptors[cameraId] = featureDetector. ▼390
    390▲ detectAndCompute(inputImage, None)
    if cameraId == 1:
      global keyPoints01
      global descriptors01
      keyPoints01, descriptors01 = featureDetector.detectAndCompute(inputImage, ▼394
```

```
    394▲ None)
395       keyPoints = keyPoints01
#         descriptors = descriptors01
      elif cameraId == 2:
        global keyPoints02
        global descriptors02
400       keyPoints02, descriptors02 = featureDetector.detectAndCompute(inputImage,  ▼400
    400▲ None)
        keyPoints = keyPoints02
#       descriptors = descriptors02

      if VERBOSE > 0:
405       timepoint01 = time.time()
          print('Detector %s found: %s points in: %s sec.' % (featureDetectionMethod,  ▼406
    406▲ len(keyPoints), timepoint01 - timepoint00))

      # Set sorting matches parameter
      global sortMatches
410   sortMatches = rospy.get_param('/stereo_vision/sortmatches', sortMatches)

      if drawFeatures:
        color = (0, 0, 255) # Red
#       color = (181, 6, 206) # Dark magenta
415 #     color = (48, 48, 255) # Red
        # Iterate over all detected key points
        for keyPoint in keyPoints:
          # Get coordinates of key point
          x, y = keyPoint.pt
420       # Draw circle around key point
          cv2.circle(inputImage, (int(x), int(y)), 2, color, -1)

      if cameraId == 1:
        global flagSubscriber1
425     flagSubscriber1 = True
        global inputImage01
        inputImage01 = inputImage
      elif cameraId == 2:
        global flagSubscriber2
430     flagSubscriber2 = True
        global inputImage02
        inputImage02 = inputImage
      else:
        print("No camera identifier found")
435     return -1

      if publishProcessed:
        if VERBOSE > 0:
          print("Publishing processed images")
440     # Create a compressed message
        outputMessage = CompressedImage()
        # Add time stamp
        outputMessage.header.stamp = rospy.Time.now()
        # Add frame identifier
```

```
445       outputMessage.header.frame_id = str(['camera0' + str(cameraId)])
          # Compression format
          outputMessage.format = compressionFormat[0]
          # Publish edge
          if publishProcessed == 2:
450         # inputImage = normalizeImage(inputImage)
            inputImage = cv2.Canny(inputImage, 100, 200)
            # mask = cv2.Canny(inputImage, 100, 200)
            # inputImage = 1 * (inputImage[..., 0] + 2 * mask) + 1 * (inputImage[..., ▼453
  453▲ 1] + 2 * mask) + 1 * (inputImage[..., 2] + 2 * mask)
            # inputImage = cv2.bitwise_and(inputImage, inputImage, mask)
455       # Compress processed image to ROS message
          outputMessage.data = np.array(cv2.imencode(compressionFormat[1], inputImage) ▼456
  456▲ [1]).tostring()
          self.publisher01.publish(outputMessage)

    ## Publish stereo image
460 #
    # @detail Publish stereo image based on specified paramters
    def publishStereoImage(self, inputImage01, inputImage02):
      if publishProcessed:
        return -1
465
      # Create a compressed message
      outputMessage = CompressedImage()
      # Add time stamp
      outputMessage.header.stamp = rospy.Time.now()
470   # Compression format
      global compressionFormat
      outputMessage.format = compressionFormat[0]

      if publishDisparity:
475     if VERBOSE > 0:
          print("Publishing disparity image")
        nd = 16
        bs = 5
        stereo = cv2.StereoBM_create(numDisparities = nd, blockSize = bs)
480     # i1 = cv2.cvtColor(inputImage01, cv2.COLOR_GRAY2BGR)
        # i2 = cv2.cvtColor(inputImage02, cv2.COLOR_GRAY2BGR)
        # @error error: /build/opencv/src/opencv-3.2.0/modules/imgproc/src/color.cpp ▼482
  482▲ :9765: error: (-215) scn == 1 && (dcn == 3 || dcn == 4) in function ▼482
  482▲ cvtColor
        # i1 = cv2.cvtColor(inputImage01, cv2.CV_8UC1)
        # i2 = cv2.cvtColor(inputImage02, cv2.CV_8UC1)
485     # @error cv2.error: /build/opencv/src/opencv-3.2.0/modules/imgproc/src/color ▼485
  485▲ .cpp:9765: error: (-215) scn == 1 && (dcn == 3 || dcn == 4) in function ▼485
  485▲ cvtColor
        if inputImage01.shape[2] == 3:
          i1 = cv2.cvtColor(inputImage01, cv2.COLOR_BGR2GRAY)
          i2 = cv2.cvtColor(inputImage02, cv2.COLOR_BGR2GRAY)
        else:
490       i1 = cv2.cvtColor(inputImage01, cv2.CV_8UC1)
          i2 = cv2.cvtColor(inputImage02, cv2.CV_8UC1)
```

```
      try:
        outputImage = stereo.compute(i1, i2)
        # outputImage = stereo.compute(inputImage01, inputImage02)
        # Write for debugging
        cv2.imwrite('stereo_vision-03-depth-map-nd' + str(nd) + '-bs' + str(bs) + '-inputImage01.png', inputImage01)
        cv2.imwrite('stereo_vision-03-depth-map-nd' + str(nd) + '-bs' + str(bs) + '-inputImage02.png', inputImage02)
        cv2.imwrite('stereo_vision-03-depth-map-nd' + str(nd) + '-bs' + str(bs) + '-i1.png', i1)
        cv2.imwrite('stereo_vision-03-depth-map-nd' + str(nd) + '-bs' + str(bs) + '-i2.png', i2)
        cv2.imwrite('stereo_vision-03-depth-map-nd' + str(nd) + '-bs' + str(bs) + '-depth.png', outputImage)
      except:
        print("Error in disparity image estimation")
        return
    else:
      if VERBOSE > 0:
        print("Publishing stereo image")
      # Concatenate images
      outputImage = np.concatenate((inputImage01, inputImage02), axis = 1)

    # Compress processed image to ROS message
    outputMessage.data = np.array(cv2.imencode(compressionFormat[1], outputImage)[1]).tostring()

    # Publish new image
    self.publisher01.publish(outputMessage)

  ## Publish PointCloud2
  #
  # @detail Publish estimated point cloud in PointCloud2 format
  def publishPointCloud(self, inputPointCloud):
    if VERBOSE > 0:
      print("Publishing estimated PointCloud2")
    if VERBOSE > 2:
      print("Estimated input point cloud\n%s" % (inputPointCloud))

    # Initialize PointCloud2
    estimatedPointCloud = PointCloud2()
    # Add frame identifier
    global stereoVisionFrameId
    global worldFrameId
    estimatedPointCloud.header.frame_id = stereoVisionFrameId + "r"

    # ''' #SWS#
    global tfListener
    try:
      tfListener.waitForTransform(stereoVisionFrameId + "r", worldFrameId, rospy.Time(0), rospy.Duration(tfWaitTime)) # rospy.Time(0) frames as strings seem to be necessary
      # tfListener.waitForTransform("camera01", "frame00", rospy.Time(0), rospy.
```

```
    536▲ Duration(tfWaitTime)) # rospy.Time(0) frames as strings seem to be  ▼536
    536▲ necessary
      # tfListener.waitForTransform("camera01r", "frame00", rospy.Time(0), rospy. ▼537
    537▲ Duration(tfWaitTime)) # rospy.Time(0) frames as strings seem to be  ▼537
    537▲ necessary
    except:
      print("Failed to wait for frame %s with reference to %s. Abort publishing  ▼539
    539▲ PointCloud2" % (stereoVisionFrameId + "r", worldFrameId))
      return
    # ''' #SWE#
    # Add time stamp
    # Use current ROS time
    estimatedPointCloud.header.stamp = rospy.Time.now()
    # ''' #SWS#
    # Use V-REP time based approach
    # global initialSimulationTime
    # global currentSimulationTime
    # Use V-REP time based approach using a system call, without using VrepInfo
    # currentSimulationTime = os.system("rostopic echo -n 1 /vrep/info | awk '/ ▼550
    550▲ simulationTime:/{getline; print $2}'") # OS call method
    # Update header time stamp
    # estimatedPointCloud.header.stamp = initialSimulationTime +  ▼552
    552▲ currentSimulationTime
    # ''' #SWE#
    ## Create PointCloud2 message
    # @note Setting point cloud paramters is not necessary, since this is done by  ▼555
    555▲ sensor_msgs.point_cloud2.create_cloud() call
    # @reference http://docs.ros.org/indigo/api/sensor_msgs/html/ ▼556
    556▲ point__cloud2_8py_source.html
    # @reference http://answers.ros.org/question/202787/using-pointcloud2-data- ▼557
    557▲ getting-xy-points-in-python/
    outputPointCloud = pc2.create_cloud_xyz32(estimatedPointCloud.header,  ▼558
    558▲ inputPointCloud)
    # Alternatively fields can be defined individually, reference from PointCloud2 ▼559
    559▲ .fields instance
    # outputPointCloud = pc2.create_cloud(estimatedPointCloud.header,  ▼560
    560▲ estimatedPointCloud.fields, inputPointCloud)

    if VERBOSE > 2:
      print("Estimated output point cloud\n%s" % (outputPointCloud))
    # Publish new image
    try:
      self.publisher02.publish(outputPointCloud)
    except:
      print("Failed to publish PointCloud2 message")

  ## Publish PoseStamped
  #
  # @detail Publish estimated pose in PoseStamped format
  def publishPose(self, inputMatrix):
    if VERBOSE > 0:
      print("Publishing estimated PoseStamped")
    if VERBOSE > 2:
```

```
      print("Estimated input pose matrix\n%s" % (inputMatrix))

    # Initialize PoseStamped
    estimatedPose = PoseStamped()
    # Add frame identifier
    global stereoVisionFrameId
    global worldFrameId
    estimatedPose.header.frame_id = stereoVisionFrameId + "r"

    # ''' #SWS#
    global tfListener
    try:
      tfListener.waitForTransform(stereoVisionFrameId + "r", worldFrameId, rospy. ▼589
    589▲ Time(0), rospy.Duration(tfWaitTime)) # rospy.Time(0) frames as strings  ▼589
    589▲ seem to be necessary
      # tfListener.waitForTransform("camera01", "frame00", rospy.Time(0), rospy. ▼590
    590▲ Duration(tfWaitTime)) # rospy.Time(0) frames as strings seem to be  ▼590
    590▲ necessary
      # tfListener.waitForTransform("camera01r", "frame00", rospy.Time(0), rospy. ▼591
    591▲ Duration(tfWaitTime)) # rospy.Time(0) frames as strings seem to be  ▼591
    591▲ necessary
    except:
      print("Failed to wait for frame %s with reference to %s. Abort publishing  ▼593
    593▲ PoseStamped" % (stereoVisionFrameId + "r", worldFrameId))
      return
    # Add time stamp
    # Use current ROS time
    estimatedPose.header.stamp = rospy.Time.now()
    # Create PoseStamped message
    # Use TF transformations for position from transformation matrix
    # estimatedPose.pose.position = tf.transformations.euler_from_matrix( ▼600
    600▲ inputMatrix) # This is for Euler angles
    estimatedPose.pose.position.x = (inputMatrix[0, 3])
    estimatedPose.pose.position.y = (inputMatrix[1, 3])
    estimatedPose.pose.position.z = (inputMatrix[2, 3])
    # Use TF transformations for orientation in quaternions from transformation  ▼604
    604▲ matrix
    # estimatedPose.pose.orientation = tf.transformations.quaternion_from_matrix( ▼605
    605▲ inputMatrix)
    q = tf.transformations.quaternion_from_matrix(inputMatrix)
    estimatedPose.pose.orientation.x = q[0]
    estimatedPose.pose.orientation.y = q[1]
    estimatedPose.pose.orientation.z = q[2]
    estimatedPose.pose.orientation.w = q[3]

    if VERBOSE > 2:
      print("Estimated output pose\nPosition: %s\nOrientation: %s" % ( ▼613
    613▲ estimatedPose.pose.position, estimatedPose.pose.orientation))
    # Publish estimated pose
    try:
      self.publisher03.publish(estimatedPose)
    except:
      print("Failed to publish PoseStamped message")
```

```
620   ## Clear buffer for stereo vision
      #
      # @details Clears all saved variables relevant for stereo vision
      def clearStereoVisionBuffer(self):
        if VERBOSE > 0:
625       print("Clearing stereo vision buffer")
        # Reference to global variables
        global flagSubscriber1
        global flagSubscriber2
        global inputImage01
630     global inputImage02
        global keyPoints01
        global keyPoints02
        global descriptors01
        global descriptors02
635     # Clear variables
        flagSubscriber1 = False
        flagSubscriber2 = False
        inputImage01 = None
        inputImage02 = None
640     keyPoints01 = None
        keyPoints02 = None
        descriptors01 = None
        descriptors02 = None

645     return

      ## Convert Euler angles to rotation matrix
      def Euler2RotationMatrix(self, alpha, beta, gamma):
        # Assign parameters
650     alpha = np.deg2rad(-90)
        beta = np.deg2rad(0)
        gamma = np.deg2rad(180)
        # Calculate partial rotation matrices
        Ra = np.matrix([[np.cos(alpha), np.sin(alpha), 0], [-np.sin(alpha), np.cos( ▼654
        654▲ alpha), 0], [0, 0, 1]])
655     Rb = np.matrix([[np.cos(beta), 0, np.sin(beta)], [0, 1, 0], [-np.sin(beta), 0, ▼655
        655▲  np.cos(beta)]])
        Rc = np.matrix([[1, 0, 0], [0, np.cos(gamma), -np.sin(gamma)], [0, np.sin( ▼656
        656▲ gamma), np.cos(gamma)]])
        # Compute final rotation matrix
        R = Rc * Rb * Ra

660     return R

      # Perform pose estimation
      ## @todo Add pose estimation class function
      # def poseEstimation(self):
665     #
        # return
```

```
#%% Main function
def main(args):
  #%% Initiliaze

  ## Call node class
  SVDE = StereoVisionDepthEstimation()

  ## Initialize rospy
  #
  # Use ROS binary name
  # roscpp_initialize(sys.argv)
  rospy.init_node('stereo_vision', anonymous = True)

  ## Set frequency rate for visualization node
  rosrate = rospy.Rate(rate)

  # Global variable references
  global VERBOSE
  global LoopCounter
  global tfListener
  global flagSubscriber1
  global flagSubscriber2
  global inputImage01
  global inputImage02

  # Initial time
  # initialTime = rospy.Time.now()
  # initialTime = os.system("rostopic echo -n 1 /vrep/info | awk '/simulationTime ▼696
    696▲ :/{getline; print $2}'") # An OS call to get time

  ## Define a transform listener
  tfListener = tf.listener.TransformListener()

  ## Dimensions of one partial stereo image
  global vh, vw
  height, width = vh, vw
  cu, cv = height / 2, width / 2

  ## @note Using camera 01 as reference

  # Stereo camera system
  ## Rotation matrix between camera 01 and camera 02
  ## No camera rotation
  R = np.eye(3)
  # Center
  ## Translation between camera 01 and camera 02
  tx = 0.2 # Actual translation in world frame
  # Cameras are rotated at -90°, 0°, +/-180°
  # tx = -1 * tx # Fixes upside problem of estimated point cloud
  ty = 0.0
  tz = 0.0
  T = np.matrix([[0, 0, 0, tx], [0, 0, 0, ty], [0, 0, 0, tz]]) # For convenience  ▼719
    719▲ in 3x4 format
```

```
720   t = T[:, 3] # Translation vector
      # Transformation matrix
      #M = merge [R|t]

      # V-REP camera parameters
725   ## Perspective camera angle -- horizontal field of view
      # @reference http://www.coppeliarobotics.com/helpFiles/en/ ▼726
        726▲ visionSensorPropertiesDialog.htm
      fovh = 60
      ## Field of view vertical
      # V-REP calculates it as min(resolutionY / resolutionX, resolutionX / ▼729
        729▲ resolutionY)
730   # Not known
      fovv = 0
      # Pixel ratio
      ku = kv = 1
      # Intrinsic parameters
735   ## Calculate focal length based on provided parameters
      # @reference http://www.forum.coppeliarobotics.com/viewtopic.php?f=9&t=4283
      if fovh != 0 and fovv == 0:
        focalLength = width / 2 / np.tan(np.deg2rad(fovh / 2))
        fovv = np.rad2deg(np.arctan(2 * focalLength / width) * 2)
740   else:
        focalLength = height / 2 / np.tan(np.deg2rad(fovv / 2))

      # Set camera 01 matrix
      camera01Matrix = np.matrix([[ku * focalLength, 0, cu], [0, kv * focalLength, cv ▼744
        744▲ ], [0, 0, 1]])
745   # Set camera 02 matrix
      # It is assumed that the stereo vision system is homogeneous and therefore ▼746
        746▲ intrinsic parameters are identical
      camera02Matrix = R * camera01Matrix

      # Extrinsic rotation matrix between robot frame and camera01
750   # xR = Euler2RotationMatrix(-90, 0, -180)
      # Center
      ## Extrinsic translation between world frame and camera 01
      # xtx = 0.0
      # xty = 0.0
755   # xtz = 0.0
      # xT = np.matrix([[0, 0, 0, tx], [0, 0, 0, ty], [0, 0, 0, tz]]) # For ▼756
        756▲ convenience in 3x4 format
      # xt = T[:, 3] # Translation vector

      # Extrinsic parameters relative to camera01
760   # Can be reused as projection matrices
      # camera01MatrixExtrinsic = np.matrix([[xR[0, 0], xR[0, 1], xR[0, 2], 0], [xR[1, ▼761
        761▲ 0], xR[1, 1], xR[1, 2], 0], [xR[2, 0], xR[2, 1], xR[2, 2], 0]])
      # camera01MatrixExtrinsic = xR * np.matrix([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, ▼762
        762▲ 1, 0]])
      # camera02MatrixExtrinsic = R * np.matrix([[1, 0, 0, tx], [0, 1, 0, ty], [0, 0, ▼763
        763▲ 1, tz]])
      # camera02MatrixExtrinsic = R * camera01MatrixExtrinsic[:, :3] + T
```

```
765
    ## Initial stereo vision system relative pose
    pMr = np.matrix(np.eye(4))
    ## Initial stereo vision system pose
    pM = np.matrix(np.eye(4))
770 ## Local camera input for time instance k
    inputImage01Pose = np.array([])
    ## Local camera input for time instance k+1
    inputImage02Pose = np.array([])
    ## Frames between sequential instances
775 frameSkip = 8
    ## Frame counter
    frameCounter = 0

    if VERBOSE > 2:
780   print("Camera matrices: \n%s\n%s" % (camera01Matrix, camera02Matrix))
      print("Initial %s pose: \n%s" % (stereoVisionFrameId, pM))
      print("Field of view: \nhorizontal: %s\nvertical: %s" % (fovh, fovv))

    while not rospy.is_shutdown():
785   # Main loop
      ## @todo Add function for updating ROS server parameters, also from command ▼786
      786▲ line and invoke in the main loop
      ## @todo Add dictionary holding for each variable a relative path so that the ▼787
      787▲ function call can be generalized
      global publishPose
      global publishProcessed
790   global publishMatched
      # global publishStereoImage
      if rospy.has_param('/stereo_vision/verbose'):
        VERBOSE = rospy.get_param('/stereo_vision/verbose', VERBOSE)
      else:
795     rospy.set_param('/stereo_vision/verbose', VERBOSE)
      if rospy.has_param('/stereo_vision/frameskip'):
        frameSkip = rospy.get_param('/stereo_vision/frameskip', frameSkip)
      else:
        rospy.set_param('/stereo_vision/frameskip', frameSkip)
800   if rospy.has_param('/stereo_vision/publish/pose'):
        publishPose = rospy.get_param('/stereo_vision/publish/pose', publishPose)
      else:
        rospy.set_param('/stereo_vision/publish/pose', publishPose)
      # if rospy.has_param('/stereo_vision/publish/stereoimage'):
805   #   publishStereoImage = rospy.get_param('/stereo_vision/publish/stereoimage', ▼805
      805▲  publishStereoImage)
      # else:
      #   rospy.set_param('/stereo_vision/publish/stereoimage', publishStereoImage)
      if rospy.has_param('/stereo_vision/publish/processed'):
        publishProcessed = rospy.get_param('/stereo_vision/publish/stereoimage',  ▼809
      809▲ publishProcessed)
810   else:
        rospy.set_param('/stereo_vision/publish/processed', publishProcessed)
      if rospy.has_param('/stereo_vision/publish/matched'):
        publishMatched = rospy.get_param('/stereo_vision/publish/matched',  ▼813
```

```
    813▲ publishMatched)
    else:
815   rospy.set_param('/stereo_vision/publish/matched', publishMatched)

    if VERBOSE and (not LoopCounter % LCF or LoopCounter == 1):
      rospy.loginfo("[CoSLAM][%s][Main] Loop: %d", NodeName, LoopCounter) # //DB
    if VERBOSE > 2:
820   print("[CoSLAM][%s][Main] Subscriber flags [1, 2]: %s, %s" % (NodeName,  ▼820
    820▲ flagSubscriber1, flagSubscriber2)) # //DB

    # Prevent drawing feature points as edges
    if publishProcessed == 2:
      drawFeatures = False
825

    # Depth estimation
    if (flagSubscriber1 == True) and (flagSubscriber2 == True):
      if VERBOSE > 0:
        print("Performing depth estimation")
830

      # Access key points
      global keyPoints01
      global keyPoints02
      global descriptors01
835   global descriptors02
      if VERBOSE > 1:
        print("Key points: %d, %d" % (len(keyPoints01), len(keyPoints02)))
#     print("Descriptors: %d, %d" % (descriptors01, descriptors02))

840   # Initialize brute force matcher
      matcher = cv2.BFMatcher_create()

      # Match points
      try:
845     matches = matcher.match(descriptors01, descriptors02)
      except:
        print("Error in matching")
        continue

850   # Sort matches according to distance
      if sortMatches:
        matches = sorted(matches, key = lambda keypoint:keypoint.distance)
      ## @todo Add more options to control the quality of matches

855   try:
        # Assign source
        matched01 = np.float32([keyPoints01[match.queryIdx].pt for match in  ▼857
    857▲ matches]).reshape(-1, 1, 2)
        matched02 = np.float32([keyPoints02[match.trainIdx].pt for match in  ▼858
    858▲ matches]).reshape(-1, 1, 2)
      except:
860     print("Error determining matches")
        continue
```

```
      # Find essential matrix for the stereo image pair
      ## @todo Add RANSAC threshold parameter
      ## @todo Assemble essential matrix per hand and compare results
      essentialMatrix, mask = cv2.findEssentialMat(matched01, matched02, method = ▼866
    866▲ cv2.RANSAC, threshold = 5.0)

      # Reduce matches to just inliers
      matchesMask = []
      j = 0
      for i in xrange(len(matches)):
#     for i in xrange(1): # Use the first match
          try:
            if VERBOSE > 1:
              print(["Mask i " + str(i) + ": " + str(mask[i]) + ", M1 " + str( ▼875
    875▲ matched01[i][0]) + ", M2 " + str(matched02[i][0])])
            if mask[i, 0]:
              matchesMask.append(mask[i, 0])
              matched01[j] = matched01[i]
              matched02[j] = matched02[i]
              j += 1
          except:
            print("Error in matched points reduction")
            continue

      if len(matches) < minMatches:
        SVDE.clearStereoVisionBuffer()
        if VERBOSE > 0:
          print("Aborted depth estimation. Too few correspondences")
        continue

      # Resize vectors holding matched points to inliers
      matched01 = matched01[:j]
      matched02 = matched02[:j]

      if VERBOSE > 0:
        print("Reduction of %d matched points by %d (%.2f%%)" % (len(matches), len ▼896
    896▲ (matches) - j, float(len(matches) - j) / len(matches) * 100))

      # Draw matches
      if publishMatched:
        # Set drawing parameters
#       drawParameters = dict(matchColor = (0, 255, 0), singlePointColor = None, ▼901
    901▲ matchesMask = matchesMask, flags = 2) # #ERR# This fails, so using try ▼901
    901▲ method necessary
          drawParameters = dict(matchColor = (0, 255, 0), singlePointColor = None, ▼902
    902▲ flags = 2)
          # Store matches in concatenated image
          try:
            inputImage = cv2.drawMatches(inputImage01, keyPoints01, inputImage02, ▼905
    905▲ keyPoints02, matches[:min(10, len(matches))], None, **drawParameters) # ▼905
    905▲ Draw first 10 matches
#           inputImage = cv2.drawMatches(inputImage01, keyPoints01, inputImage02, ▼906
    906▲ keyPoints02, matches[:10], None, **drawParameters) # Draw first 10 ▼906
```

```
    906▲ matches
#           inputImage = cv2.drawMatches(inputImage01, keyPoints01, inputImage02,  ▼907
    907▲ keyPoints02, matches[:1], None, **drawParameters) # Draw first match
            # Concatenated image dimensions
            height, width, _ = inputImage.shape

            if VERBOSE > 1:
              print("Processing matched images @ resolution: %d x %d" % (inputImage. ▼912
    912▲ shape[0], inputImage.shape[1]))
            # Split images
            # Split concatenated image into left image
#           inputImage01 = inputImage[0:inputImage.shape[0], 0:inputImage.shape[1] / ▼915
    915▲  2]
            inputImage01 = inputImage[0:height, 0:(width / 2)]
            # Split concatenated image into right image
#           inputImage02 = inputImage[0:inputImage.shape[0], inputImage.shape[1] /  ▼918
    918▲ 2:inputImage.shape[1]]
            inputImage02 = inputImage[0:height, (width / 2):width]
          except:
            print("Could not draw matches")

        # if publishStereoImage:
        # StereoVisionDepthEstimation.publishStereoImage(SVDE, inputImage01,  ▼924
    924▲ inputImage02)
        SVDE.publishStereoImage(inputImage01, inputImage02)

        ## Triangulate matched points
        ## @reference https://www.scss.tcd.ie/Gerard.Lacey/Gerard_Lacey_Homepage/ ▼928
    928▲ CS4053_Course_files/Lecture%20003%20Depth%20from%202%20cameras.pdf
        #
        # Fix this

        # Set counters
        i, j = 0, 0

        # Initialize point cloud
        pointcloud = []

        ## Estimate projected points in 3D
        # @reference https://www.scss.tcd.ie/Gerard.Lacey/Gerard_Lacey_Homepage/ ▼939
    939▲ CS4053_Course_files/Lecture%20003%20Depth%20from%202%20cameras.pdf
        # slide 9
        #

        if VERBOSE > 1:
          print("Matched points length %d" % (len(matched01[:])))

        # For all matched points
        while i < len(matched01[:]):
          if VERBOSE > 2:
            print("Matched point %d" % (i))
          # Assign local pixel coordinates of stereo camera system
          x01, y01 = matched01[i][0]
```

```
        x02, y02 = matched02[i][0]
#       P01 = (x01, y01)
#       P02 = (x02, y02)
955
        # Convert to normalized coordinates using intrinsic camera matrix
        # Transpose is just for easier writing of a vector
        P01 = np.linalg.inv(camera01Matrix) * np.matrix([x01, y01, 1]).transpose()
        P02 = np.linalg.inv(camera02Matrix) * np.matrix([x02, y02, 1]).transpose()
960 #       disparity = x01 - x02 # Pixel coordiantes
#       disparity = P01 - P02 # Normalized coordiantes
        disparity = (P01 - P02)[0] # Normalized coordiantes, using only X  ▼962
    962▲ coordinate since translation along X axis

        try:
965       # z = (tx * focalLength) / disparity #
          z = tx / disparity # Depth in normalized coordiantes
          # z = tx // (disparity + 1 // (vw * vh)) # Depth in normalized  ▼967
    967▲ coordiantes with routine preventing division by zero taking resolution  ▼967
    967▲ into account
        except:
          if VERBOSE > 0:
970         print("Disparity near/equal null", "Skipping")
          i += 1
          continue
        # if VERBOSE > 3:
        #   for k in xrange(len(z)):
975     #     print("z[", k, "] component: ", z[0, k])
        if z < 0:
          if VERBOSE > 0:
            print("Depth negative: ", z, "Skipping")
          i += 1
980       continue

        # Assuming rectified images -- Horizontal translation only
        if np.abs(P01[1] - P02[1]) > 1:
          if VERBOSE > 0:
985         print("Vertical translation by %s pixel detected. Skipping" % (P01[1]  ▼985
    985▲ - P02[1]))
          continue

#       x = tx / disparity * (x01 + x02 - 2 * inputImage01.shape[0]) / 2
#       y = tx / disparity * (y01 + y02 - 2 * inputImage01.shape[1]) / 2
990     x = P01[0] * z # Determine 3D point X coordinate
        y = P01[1] * z # Determine 3D point Y coordinate

        if VERBOSE > 1:
          print("Triangulate from normalized coordinates: ", P01.transpose(), P02. ▼994
    994▲ transpose())
995       # print("Triangulated depth: ", z)
          print("Triangulated point in 3D: ", x, y, z)

        # Compose point cloud
        pointcloud.append([x[0, 0], y[0, 0], z[0, 0]]) # Required for accessing  ▼999
```

```
999▲ numpy matrix elements
1000        if VERBOSE > 3:
              print(x[0, 0], y[0, 0], z[0, 0])

            # Increase counters
            j += 1
1005        i += 1

          if VERBOSE > 2:
            print("Point cloud: %s" % (pointcloud))

1010      # Publish assembled point cloud as PointCloud2
          SVDE.publishPointCloud(pointcloud)

          flagSubscriber1 = False
          flagSubscriber2 = False
1015
        # Pose estimation
        if publishPose and flagSubscriber1 == True:
        # if flagSubscriber1 == True and inputImage01[0] != None:
          # Access global frame, key points and descriptors
1020      # global inputImage01
          # global keyPoints01
          # global descriptors01

          # Skip frameSkip frames
1025      if frameCounter != frameSkip:
            frameCounter += 1
          else:

            # Buffer holding last frame
1030        if inputImage01Pose.shape[0] == height:
              if VERBOSE > 0:
                print("Performing pose estimation")
              # Asign buffer holding current frame
              inputImage02Pose = inputImage01
1035
              # Create local copies of key points and descriptors
              poseKeyPoints02 = keyPoints01
              poseDescriptors02 = descriptors01

1040          if VERBOSE > 1:
                print("Keypoints: %d, %d" % (len(poseKeyPoints01), len( ▼1041
1041▲ poseKeyPoints02)))

              # Initialize brute force matcher
              matcher = cv2.BFMatcher_create()
1045
              # Match points
              try:
                matches = matcher.match(poseDescriptors01, poseDescriptors02)
              except:
1050            print("Error in matching for pose")
```

```
            continue
          if VERBOSE > 0:
            print("Matches found for pose: %s" % (len(matches)))

1055      # Sort matches according to distance
          if sortMatches:
            matches = sorted(matches, key = lambda keypoint:keypoint.distance)
          ## @todo Add more options to control the quality of matches

1060      if len(matches) < minMatches:
            print("Too few initial matches for pose estimation")
          else:
            try:
              matched01 = np.float32([poseKeyPoints01[match.queryIdx].pt for  ▼1064
1064▲ match in matches]).reshape(-1, 1, 2)
1065          matched02 = np.float32([poseKeyPoints02[match.trainIdx].pt for  ▼1065
1065▲ match in matches]).reshape(-1, 1, 2)
              if VERBOSE > 1:
                print("Matches: %d, %d" % (len(matched01), len(matched02)))
            except:
              print("Error determining matches")
1070          # pass

            # Find essential matrix between two consecutive frames
            ## @todo Add RANSAC threshold parameter
            ## @todo Assemble essential matrix per hand and compare results
1075        essentialMatrix, mask = cv2.findEssentialMat(matched01, matched02,  ▼1075
1075▲ method = cv2.RANSAC, threshold = 5.0)

            ## @todo Try alternative approach using essential matrix directly
            # Filter out unnecessary points
            # Use x1 E x2 = 0 as filter condition
1080        if VERBOSE > 3:
              print("Essential matrix: \n%s" % (essentialMatrix))

            # Recover relative pose
            try:
1085          inliers = cv2.recoverPose(essentialMatrix, matched01, matched02,  ▼1085
1085▲ camera01Matrix, R, t, mask)
              if VERBOSE > 1:
                print("Pose recovered based on %s matches\nRotation: %s\ ▼1087
1087▲ nTranslation: %s" % (len(inliers), R, t))
            except:
              print("Could not recover pose from essential matrix. Skipping")
1090          inliers = 0
              inputImage02Pose = 0
              pass

            if inliers < minMatches:
1095          print("Too few matches for pose estimation")
            else:
              if VERBOSE > 0:
                print("Updating relative and absolute pose estimations")
```

```
                # Relative pose (orientation and translation)
1100            pMr[0:3, 0:3] = R
                pMr[0:3, 3] = t
                # Absolute pose (orientation and translation)
                pM = pM * pMr
                if VERBOSE > 1:
1105              print("Relative pose: %s\nAbsolute pose: %s" % (pMr, pM))

                # Publish pose
                if publishPose:
                  SVDE.publishPose(pM)
1110
                # Shift buffers
                inputImage01Pose = inputImage02Pose
                inputImage02Pose = np.array([])

1115        # First iteration -- Both buffers empty
            # elif publishPose and inputImage01Pose.shape[0] == 0 and inputImage02Pose. ▼1116
            1116▲ shape[0] == 0 and descriptors01.shape[0] == 0:
            # Shape: height, width, channels/layers
            elif publishPose and inputImage01Pose.shape[0] == 0:
              if inputImage01.shape[0] == height:
1120            inputImage01Pose = inputImage01
                poseKeyPoints01 = keyPoints01
                poseDescriptors01 = descriptors01
                if VERBOSE > 0:
                  print("Initiliaze pose estimation")
1125          else:
                if VERBOSE > 0:
                  print("Waiting for formatted vision data stream from both cameras: h: % ▼1127
            1127▲ d, w: %d" % (height, width))

            # Sleep to conform node frequency rate
1130        try:
              rosrate.sleep()
            except rospy.ROSInterruptException:
              # Clear node parameters
              # Useful disabled when debugging
1135  #       rospy.delete_param('/stereo_vision')
              pass
            except KeyboardInterrupt:
              # Warn of keyboard interrupt signal
              rospy.logwarn("[CoSLAM][%s][Main] Shutting down node", NodeName)
1140
            ## Update loop counter
            LoopCounter = LoopCounter + 1

            ## Main loop end
1145


        #%% Main call
        ## Script run condition
        if __name__ == '__main__':
```

```
1150 main(sys.argv)
```

Script 90: Python script file: licp.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Use
#!/usr/bin/env python2
# to force Python2 version
#
## @file
#
# @author Ernest Skrzypczyk
#
# @date 10.06.17
#
# @brief Local point cloud registration using ICP from PCL
#
# @detail Registration of an obtained point cloud from a stereo vision system into a point cloud acquired by a laser sensor
#
# @cmdp{$, rosrun coslam_vrep licp}
#
# @github https://github.com/em-er-es/coslam/coslam_vrep
#
# @todo Fix DOXYGEN documentation
#
# @todo Subscribe to both point cloud topics
#
# @note Code written for Python2 and OpenCV3
#

#%% Import
from __future__ import print_function #Python2 to Python3 compatibility
#Import dependencies
## Passing arguments
import sys
## Time -- Profilling
import time
#from copy import copy
import cv2 #OpenCV
# from matplotlib import pyplot as plt #Matplotlib
import numpy as np #Numpy
# import os #OS
# import logging
# from pylab import *
# import scipy as sp

## Import ROS libraries
# import roslib
#roslib.load_manifest('stereo_vision')
import rospy
#from dynamic_reconfigure.server import Server as DynamicReconfigureServer # Not used right now
import tf #Transforms

```

```
## Import messages for point cloud
from sensor_msgs.msg import PointCloud2
from geometry_msgs.msg import PoseStamped
import sensor_msgs.point_cloud2 as pc2

## Import PCL bindings
import pcl
## Import PCL registration methods
from pcl.registration import icp, gicp, icp_nl

# V-REP specific
#from vrep_common.msg import VrepInfo

# Import custom messages
# Confirm the PYTHONPATH environment variable content

#%% Script flags
## Verbose flag
VERBOSE = 3

## Debug flag
DEBUG = False

## Loop counter message filter
LCF = 100

## Minimum number of matches
minMatches = 8

## Minimum number of matches
selectICPMethod = 1

#%% Global flags
## Global message flag 1 -- VLP16
flagSubscriber1 = False

## Global message flag 2 -- Stereo vision
flagSubscriber2 = False

#%% Global variables
## Node name using console codes
NodeName = "\033[38;5;160mlicp\033[0m" # Colored
# NodeName = "licp" # Monochrome

# ROS specific
## V-REP information
# TP_LICP_I_0 = "/vrep/info"
## Subscriber topic 01
TP_LICP_I_1 = "/vlp16/pcl2filtered"
## Subscriber topic 02
TP_LICP_I_2 = "/stereo_vision/output/pcl"
## Publisher topic 01
TP_LICP_O_1 = "/licp/output/pose"
```

```
## Publisher topic 02
TP_LICP_O_2 = "/licp/output/pcl"

## Loop counter
LoopCounter = 1

## Node/visualize rate [Hz] = [fps]
rate = 25 # 25 [Hz] = 25 [fps]

## Transform listener instance
tfListener = None

## Transform wait time [s]
tfWaitTime = 1.0
# tfWaitTime = 2.0

## SimulationTime
initialSimulationTime = None
currentSimulationTime = None

## Variables for input images, detected key points and their descriptors
inputPointCloud01 = None
inputPointCloud02 = None

## Maximum iterations for the ICP routine
maximumIterations = 1000

## Vision sensor width or resolution in X
vw = 640
# vw = 1280
# @}
# vw = 1024
## Vision sensor height or resolution in Y
vh = 480
# vh = 960
# vh = 768
## Video framerate
# @note Currently not used
vf = 30
#vf = 25

## Sort matches
sortMatches = True

## Publish matched images
# publishMatched = False
publishMatched = True

## Publish pose
# publishPose = False
publishPose = True

## Frame identificators
```

```
# @{
robot01FrameId = "robot01"
#robot02FrameId = "robot02"
# stereoVisionFrameId = "stereo_vision"
stereoVisionFrameId = "camera01"
worldFrameId = "frame00"
# @}

#%% Functions

#%% Node

## Node class definition
#
# Process images from a known stereo vision system, estimate depth of detected ▼171
    171▲ feature points using triangulation of points in normalized coordinates, ▼171
    171▲ estimate pose from matches in sequential frames
class LocalIterativeClosestPoint:

  ## Node main function
  #
  # Initialization
  def __init__(self):
    #%% Get ROS server parameters
    if len(sys.argv) == 1 and rospy.has_param('/licp'):
      if VERBOSE:
        print("Clear parameters")
      rospy.delete_param('/licp')
    # Main input
    global TP_LICP_I_1
    if rospy.has_param('/licp/input/1'):
      TP_LICP_I_1 = rospy.get_param('/licp/input/1', TP_LICP_I_1)
    else:
      rospy.set_param('/licp/input/1', TP_LICP_I_1)
    # Secondary input -- to be matched against main input
    global TP_LICP_I_2
    if rospy.has_param('/licp/input/2'):
      TP_LICP_I_2 = rospy.get_param('/licp/input/2', TP_LICP_I_2)
    else:
      rospy.set_param('/licp/input/2', TP_LICP_I_2)
    # ICP method
    global selectICPMethod
    if rospy.has_param('/licp/method'):
      selectICPMethod = rospy.get_param('/licp/method', selectICPMethod)
    else:
      rospy.set_param('/licp/method', selectICPMethod)

    #%% Set publisher/subscriber
    ## Publish estimated pose (position and orientation in 3D)
    self.publisher01 = rospy.Publisher(TP_LICP_O_1, PoseStamped, queue_size = 1, ▼204
    204▲ latch = True)

    ## Publish corrected point cloud
```

```
    self.publisher02 = rospy.Publisher(TP_LICP_O_2, PointCloud2, queue_size = 1, ▼207
    207▲ latch = True)

    ## Subscribe to V-REP information
    # self.subscriber00 = rospy.Subscriber(TP_LICP_I_0, VrepInfo, self.vrepTime, ▼210
    210▲ queue_size = 1)

    ## Subscribe to main cloud
    self.subscriber01 = rospy.Subscriber(TP_LICP_I_1, PointCloud2, self. ▼213
    213▲ subscriberCallback, callback_args = (worldFrameId, robot01FrameId), ▼213
    213▲ queue_size = 1)

    ## Subscribe to secondary cloud
    self.subscriber02 = rospy.Subscriber(TP_LICP_I_2, PointCloud2, self. ▼216
    216▲ subscriberCallback, callback_args = (robot01FrameId, stereoVisionFrameId ▼216
    216▲ + "r"), queue_size = 1)

    # Print topics
    if VERBOSE:
#     rospy.loginfo("[CoSLAM][%s][Init] Subscribers: %s, %s", NodeName, ▼220
    220▲ TP_LICP_I_1, TP_LICP_I_2)
#     rospy.loginfo("[CoSLAM][%s][Init] Publishers: %s, %s", NodeName, TP_LICP_O_1 ▼221
    221▲ , TP_LICP_O_2)
      print("[CoSLAM][%s][Init] Subscribers: %s, %s" % (NodeName, TP_LICP_I_1, ▼222
    222▲ TP_LICP_I_2))
      print("[CoSLAM][%s][Init] Publishers: %s, %s" % (NodeName, TP_LICP_O_1, ▼223
    223▲ TP_LICP_O_2))

  ## Subscriber callback for V-REP
  #
  # @detail Get simulation time
  def vrepTime(self, inputMessage):
    global initialSimulationTime
    global currentSimulationTime
    if initialSimulationTime == None:
      initialSimulationTime = inputMessage.simulationTime
      currentSimulationTime = inputMessage.simulationTime
    else:
      currentSimulationTime = inputMessage.simulationTime
    return currentSimulationTime

  ## Subscriber callback for vision
  #
  # @detail Process images by detecting feautures and publish processed image
  def subscriberCallback(self, inputMessage, arguments):
    #%% Assign arguments
    ## Convert ROS message to NumPy image array
    inputPointCloudArray = np.fromstring(inputMessage.data, np.float32)

    ## Create PCL point cloud instance
    inputPointCloud = pcl.PointCloud()
    ## Convert PointCloud2 message data to PCL point cloud
    # @reference https://github.com/strawlab/python-pcl/blob/master/examples/ ▼249
```

```
    249▲ kdtree.py
250     pointcloud = []
        for i in xrange(len(inputPointCloudArray)):
          try:
            inputPointCloudArray[i + 2]
          except:
255         pass
          else:
            # Compose point cloud
            pointcloud.append([inputPointCloudArray[i], inputPointCloudArray[i + 1],  ▼258
        258▲ inputPointCloudArray[i + 2]])
            if VERBOSE > 3:
260           print([inputPointCloudArray[i], inputPointCloudArray[i + 1],  ▼260
        260▲ inputPointCloudArray[i + 2]])

        # Convert point cloud list to PCL point cloud format
        inputPointCloud.from_list(pointcloud)
        # Convert point cloud numpy array to PCL point cloud format
265     # inputPointCloud.from_array(inputPointCloudArray)

        ## Assign
        sourceIdFrame = arguments[0]
        targetIdFrame = arguments[1]
270

        #%% Process data
        if VERBOSE > 1:
          print("Point cloud received. Source: %s, target: %s" % (sourceIdFrame,  ▼273
        273▲ targetIdFrame))

275     if sourceIdFrame == worldFrameId:
          global flagSubscriber1
          flagSubscriber1 = True
          global inputPointCloud01
          inputPointCloud01 = inputPointCloud
280     elif sourceIdFrame == stereoVisionFrameId + "r":
          global flagSubscriber2
          flagSubscriber2 = True
          global inputPointCloud02
          inputPointCloud02 = inputPointCloud
285     else:
          print("No known identifier found")
          return -1

    ## Perform ICP
290 #
    # @detail Perform iterative closest point registration using specified method on
    # def localICP(self, inputPointCloud01, inputPointCloud02)
    def localICP(self):
      if VERBOSE:
295     print("Publishing estimated PointCloud2")
      global inputPointCloud01
      global inputPointCloud02
      global maximumIterations
```

```
    global transformationMatrix
300
    # converged, transf, estimate, fitness = algo(source, target,
    ## @todo Implement different ICP methods
    converged, transformationMatrix, estimate, fitness = icp(inputPointCloud01,  ▼303
    303▲ inputPointCloud02, max_iter = maximumIterations)
    if VERBOSE > 0:
305   print("Converged: %s\nTransformation matrix: \n%s\nEstimate: %s\nFitness: %s ▼305
    305▲ " % (converged, transformationMatrix, estimate, fitness))

    return transformationMatrix

  ## Publish PointCloud2
310 #
  # @detail Publish estimated point cloud in PointCloud2 format
  def publishPointCloud(self, inputPointCloud):
    if VERBOSE:
      print("Publishing estimated PointCloud2")
315 if VERBOSE > 2:
      print("Estimated input point cloud\n%s" % (inputPointCloud))

    # Initialize PointCloud2
    estimatedPointCloud = PointCloud2()
320 # Add frame identifier
    global stereoVisionFrameId
    global tfListener
    global worldFrameId
    try:
325   tfListener.waitForTransform(stereoVisionFrameId, worldFrameId, rospy.Time(0) ▼325
    325▲ , rospy.Duration(tfWaitTime))
    except:
      print("Failed to wait for frame %s with reference to %s. Abort publishing  ▼327
    327▲ PointCloud2" % (stereoVisionFrameId, worldFrameId))
      return
    # Add time stamp
330 # Use current ROS time
    estimatedPointCloud.header.stamp = rospy.Time.now()
    ## Create PointCloud2 message
    # @note Setting pointcloud paramters is not necessary, since this is done by  ▼333
    333▲ sensor_msgs.point_cloud2.create_cloud() call
    # @reference http://docs.ros.org/indigo/api/sensor_msgs/html/ ▼334
    334▲ point__cloud2_8py_source.html
335 # @reference http://answers.ros.org/question/202787/using-pointcloud2-data- ▼335
    335▲ getting-xy-points-in-python/
    outputPointCloud = pc2.create_cloud_xyz32(estimatedPointCloud.header,  ▼336
    336▲ inputPointCloud)
    # Alternatively fields can be defined individually, reference from PointCloud2 ▼337
    337▲ .fields instance
    # outputPointCloud = pc2.create_cloud(estimatedPointCloud.header,  ▼338
    338▲ estimatedPointCloud.fields, inputPointCloud)

340 if VERBOSE > 2:
      print("Estimated output point cloud\n%s" % (outputPointCloud))
```

```
    # Publish new image
    try:
      self.publisher02.publish(outputPointCloud)
    except:
      print("Failed to publish PointCloud2 message")

  ## Publish PoseStamped
  #
  # @detail Publish estimated pose in PoseStamped format
  def publishPose(self, inputMatrix):
    if VERBOSE:
      print("Publishing estimated PoseStamped")
    if VERBOSE > 2:
      print("Estimated input pose matrix\n%s" % (inputMatrix))

    # Initialize PoseStamped
    estimatedPose = PoseStamped()
    # Add frame identifier
    global stereoVisionFrameId
    estimatedPose.header.frame_id = stereoVisionFrameId + "r"

    global tfListener
    # global stereoVisionFrameId
    global worldFrameId
    try:
      tfListener.waitForTransform(stereoVisionFrameId, worldFrameId, rospy.Time(0) ▼367
    367▲ , rospy.Duration(tfWaitTime))
    except:
      print("Failed to wait for frame %s with reference to %s. Abort publishing  ▼369
    369▲ PoseStamped" % (stereoVisionFrameId, worldFrameId))
      return
    # Add time stamp
    # Use current ROS time
    estimatedPose.header.stamp = rospy.Time.now()
    # Create PoseStamped message
    # Use TF transformations for position in quaternions from transformation  ▼375
    375▲ matrix
    estimatedPose.pose.position = tf.transformations.euler_from_matrix(inputMatrix ▼376
    376▲ )
    # Use TF transformations for orientation in quaternions from transformation  ▼377
    377▲ matrix
    estimatedPose.pose.orientation = tf.transformations.quaternion_from_matrix( ▼378
    378▲ inputMatrix)

    if VERBOSE > 2:
      print("Estimated output pose\nPosition: \n%s\nOrientation: %s" % ( ▼381
    381▲ estimatedPose.position, estimatedPose.orientation))
    # Publish pose
    try:
      self.publisher03.publish(estimatedPose)
    except:
      print("Failed to publish PoseStamped message")
```

```
  ## Clear buffer for stereo vision
  #
390  # @details Clears all relevant saved variables
  def clearPointCloudBuffer(self):
    if VERBOSE:
      print("Clearing point cloud buffer")
    # Reference to global variables
395   global flagSubscriber1
    global flagSubscriber2
    # Clear variables
    flagSubscriber1 = False
    flagSubscriber2 = False
400
    return

  ## Convert Euler angles to rotation matrix
  def Euler2RotationMatrix(alpha, beta, gamma):
405   # Assign parameters
    alpha = np.deg2rad(-90)
    beta = np.deg2rad(0)
    gamma = np.deg2rad(180)
    # Calculate partial rotation matrices
410   Ra = np.matrix([[np.cos(alpha), np.sin(alpha), 0], [-np.sin(alpha), np.cos( ▼410
    410▲ alpha), 0], [0, 0, 1]])
    Rb = np.matrix([[np.cos(beta), 0, np.sin(beta)], [0, 1, 0], [-np.sin(beta), 0, ▼411
    411▲  np.cos(beta)]])
    Rc = np.matrix([[1, 0, 0], [0, np.cos(gamma), -np.sin(gamma)], [0, np.sin( ▼412
    412▲ gamma), np.cos(gamma)]])
    # Compute final rotation matrix
    R = Rc * Rb * Ra
415
    return R

  # Perform pose estimation
  ## @todo Add pose estimation class function
420  # def poseEstimation(self):
    #
    # return


425 #%% Main function
def main(args):
  #%% Initiliaze

  ## Call node class
430  LICP = LocalIterativeClosestPoint()

  ## Initialize rospy
  #
  # Use ROS binary name
435  # roscpp_initialize(sys.argv)
  rospy.init_node('licp', anonymous = True)
```

```
## Set frequency rate for visualization node
rosrate = rospy.Rate(rate)

# Global variable references
global VERBOSE
global LoopCounter
global tfListener
global flagSubscriber1
global flagSubscriber2
global inputPointCloud01
global inputPointCloud02

# Initial time
# initialTime = rospy.Time.now()
# initialTime = os.system("rostopic echo -n 1 /vrep/info | awk '/simulationTime ▼452
  452▲ :/{getline; print $2}'") # An OS call to get time

## Define a transform listener
tfListener = tf.listener.TransformListener()

## Dimensions of one partial stereo image
global vh, vw
height, width = vh, vw
cu, cv = height / 2, width / 2

## @note Using camera 01 as reference

# Stereo camera system
## Rotation matrix between camera 01 and camera 02
## No camera rotation
R = np.eye(3)
# Center
## Translation between camera 01 and camera 02
tx = 0.2 # Actual translation in world frame
# Cameras are rotated at -90°, 0°, +/-180°
# tx = -1 * tx # Fixes upside problem of estimated point cloud
ty = 0.0
tz = 0.0
T = np.matrix([[0, 0, 0, tx], [0, 0, 0, ty], [0, 0, 0, tz]]) # For convenience  ▼475
  475▲ in 3x4 format
t = T[:, 3] # Translation vector
# Transformation matrix
#M = merge [R|t]

## Target pose
pMt = np.matrix(np.eye(4))

# if VERBOSE > 2:
  # print("Camera matrices: \n%s\n%s" % (camera01Matrix, camera02Matrix))
  # print("Initial %s pose: \n%s" % (stereoVisionFrameId, camera01Pose))

while not rospy.is_shutdown():
  # Main loop
```

```
    ## @todo Add function for updating ROS server parameters, also from command ▼489
    489▲ line and invoke in the main loop
490 if rospy.has_param('/licp/verbose'):
      VERBOSE = rospy.get_param('/licp/verbose', VERBOSE)
    else:
      rospy.set_param('/licp/verbose', VERBOSE)

495 if VERBOSE and (not LoopCounter % LCF or LoopCounter == 1):
      rospy.loginfo("[CoSLAM][%s][Main] Loop: %d", NodeName, LoopCounter) # //DB
    if VERBOSE > 2:
      print("[CoSLAM][%s][Main] Subscriber flags [1, 2]: %s, %s" % (NodeName, ▼498
    498▲ flagSubscriber1, flagSubscriber2)) # //DB

500 # Perform ICP
    if (flagSubscriber1 == True) and (flagSubscriber2 == True):
      if VERBOSE:
        print("Performing iterative closest point registration")

505   # Access key points
      global inputPointCloud01
      global inputPointCloud02

      if VERBOSE > 2:
510     print("Point cloud: %s" % (pointcloud))

      # Publish assembled point cloud as PointCloud2
      LICP.publishPointCloud(pointcloud)

515   flagSubscriber1 = False
      flagSubscriber2 = False

    # Sleep to conform node frequency rate
    try:
520   rosrate.sleep()
    except rospy.ROSInterruptException:
      # Clear node parameters
      # Useful disabled when debugging
#     rospy.delete_param('/licp')
525   pass
    except KeyboardInterrupt:
      # Warn of keyboard interrupt signal
      rospy.logwarn("[CoSLAM][%s][Main] Shutting down node", NodeName)

530 ## Update loop counter
    LoopCounter = LoopCounter + 1

    ## Main loop end

535

#%% Main call
## Script run condition
if __name__ == '__main__':
  main(sys.argv)
```

Script 91: Python script file: stereo_image_preprocessor.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Use
#!/usr/bin/env python2
# to force Python2 version
#
## @file
#
# @author Ernest Skrzypczyk
#
# @date 10.06.17
#
# @brief Preprocessor for stereo_image_proc
#
# @detail Provide necessary data from simulation to stereo_image_proc package
#
# @cmdp{$, rosrun coslam_vrep stereo_image_preprocessor}
#
# @github https://github.com/em-er-es/coslam/coslam_vrep
#
# @todo Fix DOXYGEN documentation
#
# @done Subscribe to both camera topics
#
# @done Published processed stereo images
#
# @reference http://wiki.ros.org/rospy_tutorials/Tutorials/ ▼27
    27▲ WritingImagePublisherSubscriber
#
# @reference http://wiki.ros.org/stereo_image_proc
#
# @note Code written for Python2 and OpenCV3
#

#%% Import
from __future__ import print_function #Python2 to Python3 compatibility
#Import dependencies
## Passing arguments
import sys
## Time -- Profilling
import time
#from copy import copy
import cv2 #OpenCV
# from matplotlib import pyplot as plt #Matplotlib
import numpy as np #Numpy
# import os #OS
# import logging
# from pylab import *
# import scipy as sp

## Import ROS libraries
# import roslib
```

```
#roslib.load_manifest('stereo_vision')
import rospy
#from dynamic_reconfigure.server import Server as DynamicReconfigureServer # Not ▼54
    54▲ used right now
import tf #Transforms

## Import messages for vision
# from cv_bridge import CvBridge #D#
from sensor_msgs.msg import CameraInfo
from sensor_msgs.msg import CompressedImage
from sensor_msgs.msg import Image

# Import custom messages
# Confirm the PYTHONPATH environment variable content

#%% Script flags
## Verbose flag
VERBOSE = 1

## Debug flag
DEBUG = False

## Loop counter message filter
LCF = 1000

#%% Global flags
## Global message flag 1 -- Camera 01
flagSubscriber1 = False

## Global message flag 2 -- Camera 02
flagSubscriber2 = False

# Stereo vision specific
## Camera 01 flag
# Set upon receiving image for left/main/01 camera
flagCamera01 = False
## Camera 02 flag
# Set upon receiving image for right/secondary/02 camera
flagCamera02 = False

#%% Global variables
## Node name using console codes
NodeName = "\033[38;5;160mSIPP\033[0m" # Colored
# NodeName = "SIPP" # Monochrome

# ROS specific
## Subscriber topic 01
TP_SIPP_I_1 = "/vrep/stereo_vision/camera01/compressed"
## Subscriber topic 02
TP_SIPP_I_2 = "/vrep/stereo_vision/camera02/compressed"
## Publisher topic 01
TP_SIPP_O_1 = "/left/image_raw/compressed"
# TP_SIPP_O_1 = "/left/image_raw"
```

```
# TP_SIPP_O_1 = "/left/image_rect_color"
## Publisher topic 02
TP_SIPP_O_2 = "/right/image_raw/compressed"
# TP_SIPP_O_2 = "/right/image_raw"
# TP_SIPP_O_2 = "/right/image_rect_color"
## Publisher topic 03
TP_SIPP_O_3 = "/left/camera_info"
## Publisher topic 04
TP_SIPP_O_4 = "/right/camera_info"

## Loop counter
LoopCounter = 1

## Node/visualize rate [Hz] = [fps]
rate = 25 # 25 [Hz] = 25 [fps]

## Transform listener instance
tfListener = None

## Transform wait time [s]
tfWaitTime = 1.0

## Variables for input images, detected keypoints and their descriptors
# @{
inputImage01 = None
inputImage02 = None
# @}

## Vision sensor width or resolution in X
vw = 640
# vw = 1280
# vw = 1024
## Vision sensor height or resolution in Y
vh = 480
# vh = 960
# vh = 768

## Publish disparity
# publishProcessed = False
publishProcessed = True

## Selector
selector = False

## Frame identificators
# @{
robot01FrameId = "robot01"
stereoVisionFrameId = "stereo_vision"
stereoVisionFrameId01 = "camera01"
stereoVisionFrameId02 = "camera02"
worldFrameId = "frame00"
# @}
```

```
#%% Functions


#%% Node

## Node class definition
#
# Processed images from a known stereo vision system and estimates depth of ▼164
    164▲ detected feature points using triangulation of points in normalized ▼164
    164▲ coordinates
class StereoImagePreprocessor:

  ## Node main function
  #
  # Initialization
  def __init__(self):
    #%% Get ROS server parameters
    if len(sys.argv) == 1 and rospy.has_param('/stereo_vision_preprocessor'):
      if VERBOSE:
        print("Clear parameters")
      rospy.delete_param('/stereo_vision_preprocessor')
    # Input left
    global TP_SIPP_I_1
    if rospy.has_param('/stereo_vision_preprocessor/input/1'):
      TP_SIPP_I_1 = rospy.get_param('/stereo_vision_preprocessor/input/1', ▼179
    179▲ TP_SIPP_I_1)
    else:
      rospy.set_param('/stereo_vision_preprocessor/input/1', TP_SIPP_I_1)
    # Input right
    global TP_SIPP_I_2
    if rospy.has_param('/stereo_vision_preprocessor/input/2'):
      TP_SIPP_I_2 = rospy.get_param('/stereo_vision_preprocessor/input/2', ▼185
    185▲ TP_SIPP_I_2)
    else:
      rospy.set_param('/stereo_vision_preprocessor/input/2', TP_SIPP_I_2)
    global selector
    if rospy.has_param('/stereo_vision_preprocessor/selector'):
      selectFeatureDetectionMethod = rospy.get_param('/stereo_vision_preprocessor/ ▼190
    190▲ featuredetectionmethod', selector)
    else:
      rospy.set_param('/stereo_vision_preprocessor/selector', selector)

    ## OpenCV bridge instance
    # self.bridge = CvBridge() #D#

    #%% Set publisher/subscriber
    ## Publish processed image -- left
    # self.publisher01 = rospy.Publisher(TP_SIPP_O_1, Image, queue_size = 1, latch ▼199
    199▲  = True)
    self.publisher01 = rospy.Publisher(TP_SIPP_O_1, CompressedImage, queue_size =  ▼200
    200▲ 1, latch = True)

    ## Publish processed image -- right
```

```
    # self.publisher02 = rospy.Publisher(TP_SIPP_O_2, Image, queue_size = 1, latch ▼203
    203▲  = True)
     self.publisher02 = rospy.Publisher(TP_SIPP_O_2, CompressedImage, queue_size =  ▼204
    204▲ 1, latch = True)
205
    ## Publish camera info -- left
     self.publisher03 = rospy.Publisher(TP_SIPP_O_3, CameraInfo, queue_size = 1,  ▼207
    207▲ latch = True)

    ## Publish camera info -- right
210  self.publisher04 = rospy.Publisher(TP_SIPP_O_4, CameraInfo, queue_size = 1,  ▼210
    210▲ latch = True)

    ## Subscribe to camera 01
     self.subscriber01 = rospy.Subscriber(TP_SIPP_I_1, CompressedImage, self. ▼213
    213▲ subscriberCallback, callback_args = (01, stereoVisionFrameId), queue_size ▼213
    213▲  = 1)

215 ## Subscribe to camera 02
     self.subscriber02 = rospy.Subscriber(TP_SIPP_I_2, CompressedImage, self. ▼216
    216▲ subscriberCallback, callback_args = (02, stereoVisionFrameId), queue_size ▼216
    216▲  = 1)

    # Print topics
     if VERBOSE:
220 #      rospy.loginfo("[CoSLAM][%s][Init] Subscribers: %s, %s", NodeName,  ▼220
    220▲ TP_SIPP_I_1, TP_SIPP_I_2)
    #      rospy.loginfo("[CoSLAM][%s][Init] Publishers: %s, %s, %s, %s", NodeName,  ▼221
    221▲ TP_SIPP_O_1, TP_SIPP_O_2, TP_SIPP_O_3, TP_SIPP_O_4)
       print("[CoSLAM][%s][Init] Subscribers: %s, %s" % (NodeName, TP_SIPP_I_1,  ▼222
    222▲ TP_SIPP_I_2))
       print("[CoSLAM][%s][Init] Publishers: %s, %s, %s, %s" % (NodeName,  ▼223
    223▲ TP_SIPP_O_1, TP_SIPP_O_2, TP_SIPP_O_3, TP_SIPP_O_4))

225

  ## Subscriber callback for vision
  #
  # @detail Processes images by detecting feautures and publishing processed image
  def subscriberCallback(self, inputMessage, arguments):
230   #%% Assign arguments
    ## Convert ROS message to NumPy image array
    inputImageArray = np.fromstring(inputMessage.data, np.uint8)

    ## Assign transformation
235 cameraId = arguments[0]
    cameraIdFrame = arguments[1]

    #%% Process data
     if VERBOSE > 1:
240    print("Camera %s detected for unit %s" % (cameraId, cameraIdFrame))

    ## Read input image
    # @see http://docs.opencv.org/3.0-beta/modules/imgcodecs/doc/ ▼243
```

```
    243▲ reading_and_writing_images.html#imread
    # inputImage = cv2.imdecode(inputImageArray, cv2.CV_8UC1)
    # inputImage = cv2.imdecode(inputImageArray, cv2.COLOR_BGR2GRAY)
    inputImage = cv2.imdecode(inputImageArray, cv2.IMREAD_COLOR)

    if cameraId == 1:
      global flagSubscriber1
      flagSubscriber1 = True
      global inputImage01
      inputImage01 = inputImage
    elif cameraId == 2:
      global flagSubscriber2
      flagSubscriber2 = True
      global inputImage02
      inputImage02 = inputImage
    else:
      print("No camera identifier found")
      return -1

    # if VERBOSE > 0:
    # if VERBOSE:
      # print("Publishing processed images")
    # Create a compressed message
    # outputMessage = Image()
    outputMessage = CompressedImage()
    # Add time stamp
    outputMessage.header.stamp = rospy.Time.now()
    # Add frame identifier
    outputMessage.header.frame_id = "camera0" + str(cameraId)
    # Compress processed image to ROS message
    outputMessage.data = np.array(cv2.imencode('.jpg', inputImage)[1]).tostring()
    # outputMessage.data = self.bridge.cv2_to_imgmsg(inputImage, 'bgr8') #D#
    # cv2.imshow("test", inputImage)
    # cv2.waitKey(20)
    # outputMessage.data = np.array(inputImage).tostring()

    # Construct camera information message
    if cameraId == 1:
      cameraMatrix = camera01Matrix
      projectionMatrix = camera01MatrixExtrinsic
    elif cameraId == 2:
      cameraMatrix = camera02Matrix
      projectionMatrix = camera02MatrixExtrinsic
    # eval("global camera" + str(cameraId) + "Matrix")
    # eval("global camera" + str(cameraId) + "MatrixExtrinsic")

    outputMatrix = CameraInfo()
    outputMatrix.header = outputMessage.header
    # outputMatrix.K = eval("camera" + str(cameraId) + "Matrix").ravel()
    # outputMatrix.K = [cameraMatrix.ravel()]
    # outputMatrix.K = cameraMatrix.tolist()
    outputMatrix.K = cameraMatrix
    outputMatrix.D = [0.000000, 0.000000, 0.000000, 0.000000, 0.000000]
```

```
    outputMatrix.R = [0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,  ▼296
    296▲ 0.000000, 0.000000, 0.000000]
    outputMatrix.P = [1.000000, 0.000000, 0.000000, 0.000000, 0.000000, 1.000000,  ▼297
    297▲ 0.000000, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000]
    outputMatrix.height = vh
    outputMatrix.width = vw
300 outputMatrix.distortion_model = 'plumb_bob'
    outputMatrix.binning_x = 0
    outputMatrix.binning_y = 0
    outputMatrix.roi.x_offset = 0
    outputMatrix.roi.y_offset = 0
305 outputMatrix.roi.height = vh
    outputMatrix.roi.width = vw
    outputMatrix.roi.do_rectify = 0

    global tfListener
310 try:
      tfListener.waitForTransform("camera0" + str(cameraId), "frame00", rospy.Time ▼311
    311▲ (0), rospy.Duration(tfWaitTime)) # rospy.Time(0) seems to be necessary
    except:
      print("Failed to wait for frame %s with reference to %s. Abort publishing  ▼313
    313▲ PointCloud2" % (stereoVisionFrameId, worldFrameId))
      return
315

    if cameraId == 1:
      self.publisher01.publish(outputMessage)
      self.publisher03.publish(outputMatrix)
    elif cameraId == 2:
320   self.publisher02.publish(outputMessage)
      self.publisher04.publish(outputMatrix)
    eval("self.publisher0" + str(cameraId) + ".publish(outputMessage)")
    self.publisher01.publish(outputMessage)


325
  def clearStereoVisionBuffer(self):
    if VERBOSE:
      print("Clearing stereo vision buffer")
    # Reference to global variables
330 global flagSubscriber1
    global flagSubscriber2
    global inputImage01
    global inputImage02
    global keyPoints01
335 global keyPoints02
    global descriptors1
    global descriptors2
    # Clear variables
    flagSubscriber1 = False
340 flagSubscriber2 = False
    inputImage01 = None
    inputImage02 = None
    keyPoints01 = None
    keyPoints02 = None
```

```
345     descriptors1 = None
        descriptors2 = None

        return


350
#%% Main function
def main(args):
  #%% Initiliaze

  ## Call node class
  SIPP = StereoImagePreprocessor()

  ## Initialize rospy
  #
  # Use ROS binary name
  # roscpp_initialize(sys.argv)
  rospy.init_node('stereo_image_preprocessor', anonymous = True)

  ## Set frequency rate for visualization node
  rosrate = rospy.Rate(rate)

  # Global variable references
  global VERBOSE
  global LoopCounter
  global tfListener
  global flagSubscriber1
  global flagSubscriber2
  global inputImage01
  global inputImage02

  ## Define a transform listener
  tfListener = tf.listener.TransformListener()

  ## Dimensions of one partial stereo image
  global vh, vw
  height, width = vh, vw
  cu, cv = height / 2, width / 2

  ## @note Using camera 01 as reference

  # Stereo camera system
  ## Rotation matrix between camera 01 and camera 02
  ## No camera rotation
  R = np.eye(3)
  # Center
  ## Translation between camera 01 and camera 02
  tx = 0.2
  ty = 0.0
  tz = 0.0
  T = np.matrix([[0, 0, 0, tx], [0, 0, 0, ty], [0, 0, 0, tz]]) # For convenience ▼395
    395▲ in 3x4 format
  t = T[:, 3] # Translation vector
```

```
  # V-REP camera parameters
  ## Perspective camera angle -- horizontal field of view
400 # @reference http://www.coppeliarobotics.com/helpFiles/en/ ▼400
    400▲ visionSensorPropertiesDialog.htm
  fovh = 60
  ## Field of view vertical
  # V-REP calculates it as min(resolutionY / resolutionX, resolutionX / ▼403
    403▲ resolutionY)
  # Not known
405 fovv = 0
  # Pixel ratio
  ku = kv = 1
  # Intrinsic parameters
  ## Calculate focal length based on provided parameters
410 # @reference http://www.forum.coppeliarobotics.com/viewtopic.php?f=9&t=4283
  if fovh != 0 and fovv == 0:
    focalLength = width / 2 / np.tan(np.deg2rad(fovh / 2))
    fovv = np.arctan(2 * focalLength / width) * 2
  else:
415   focalLength = height / 2 / np.tan(np.deg2rad(fovv / 2))

  # Set camera 01 matrix
  global camera01Matrix
  # camera01Matrix = np.matrix([[ku * focalLength, 0, cu], [0, kv * focalLength, ▼419
    419▲ cv], [0, 0, 1]])
420 camera01Matrix = [ku * focalLength, 0, cu, 0, kv * focalLength, cv, 0, 0, 1]
  # Set camera 02 matrix
  # It is assumed that the stereo vision system is homogeneous and therefore ▼422
    422▲ intrinsic parameters are identical
  global camera02Matrix
  camera02Matrix = camera01Matrix
425

  # Extrinsic parameters relative to camera01
  # Can be reused as projection matrices
  global camera01MatrixExtrinsic
  global camera02MatrixExtrinsic
430 camera01MatrixExtrinsic = np.matrix([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0]])
  camera02MatrixExtrinsic = np.matrix([[1, 0, 0, tx], [0, 1, 0, ty], [0, 0, 1, tz ▼431
    431▲ ]])

  if VERBOSE > 2:
    print("Camera matrices: \n%s\n%s" % (camera01Matrix, camera02Matrix))
435

  while not rospy.is_shutdown():
    ## Main loop
    VERBOSE = rospy.get_param('/stereo_vision_preprocessor/verbose', VERBOSE)
#   if VERBOSE:
440   if VERBOSE and not LoopCounter % LCF:
      rospy.loginfo("[CoSLAM][%s][Main] Loop: %d", NodeName, LoopCounter) # //DB
    if VERBOSE > 2:
      print("[CoSLAM][%s][Main] Subscriber flags [1, 2]: %s, %s" % (NodeName, ▼443
    443▲ flagSubscriber1, flagSubscriber2)) # //DB
```

```
445       ''' #SWS#
          ##
          if (flagSubscriber1 == True) and (flagSubscriber2 == True):
            if VERBOSE:
              print("Processing stereo images set")
450
            # if publishStereoImage:
            # StereoImagePreprocessor.publishStereoImage(SIPP, inputImage01,  ▼452
          452▲ inputImage02)
            SIPP.publishStereoImage(inputImage01, inputImage02)

455         ## Disparity
#           stereo = cv2.StereoBM_create(numDisparities = 16, blockSize = 15)
#           inputImage01 = stereo.compute(inputImage01, inputImage02)

            flagSubscriber1 = False
460         flagSubscriber2 = False

          # ''' #SWE#

          # Sleep to conform node frequency rate
465       try:
            rosrate.sleep()
          except rospy.ROSInterruptException:
            # Clear node parameters
            # Useful disabled when debugging
470 #       rospy.delete_param('/stereo_vision_preprocessor')
            pass
          except KeyboardInterrupt:
            # Warn of keyboard interrupt signal
            rospy.logwarn("[CoSLAM][%s][Main] Shutting down node", NodeName)
475
          ## Update loop counter
          LoopCounter = LoopCounter + 1

          ## Main loop end
480


#%% Main call
## Script run condition
if __name__ == '__main__':
485   main(sys.argv)
```

Script 92: Text file: CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
PROJECT(coslam_vrep)

## Add support for C++11, supported in ROS Kinetic and newer
## @warn Adding c++11 definitions might break PCL calls, which is often compiled without it, making this package unusable
## @note Use add_compile_options(-std=c++11) for specific targets
#add_definitions(-std=c++11)
#set(CMAKE_CXX_FLAGS_DEBUG "${CMAKE_CXX_FLAGS_DEBUG} -O1")

#Custom command to pass on arguments
if(DEFINED DEBUG)
  execute_process(
    COMMAND sed -i "s/#define DEBUG .*/#define DEBUG ${DEBUG}/" "${CMAKE_SOURCE_DIR}/include/${PROJECT_NAME}/${PROJECT_NAME}.hpp"
  )
endif()
if(DEFINED DEBUG_LC)
  execute_process(
    COMMAND sed -i "s/#define DEBUG_LC .*/#define DEBUG_LC ${DEBUG_LC}/" "${CMAKE_SOURCE_DIR}/include/${PROJECT_NAME}/${PROJECT_NAME}.hpp"
  )
endif()
if(DEFINED VERBOSE)
  execute_process(
    COMMAND sed -i "s/#define VERBOSE .*/#define VERBOSE ${VERBOSE}/" "${CMAKE_SOURCE_DIR}/include/${PROJECT_NAME}/${PROJECT_NAME}.hpp"
  )
endif()

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  cv_bridge
  geometry_msgs
  image_transport
  pcl_conversions
  pcl_ros
  roscpp
  rospy
  sensor_msgs
  velodyne_msgs
  vrep_common
)

## System dependencies are found with CMake's conventions
find_package(OpenCV REQUIRED)
find_package(PCL REQUIRED)
#find_package(VISP REQUIRED)

#############
```

```
## Python  ##
#############

## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
catkin_python_setup()

################################################
## Declare ROS messages, services and actions ##
################################################

## To declare and build messages, services or actions from within this
## package, follow these steps:
## * Let MSG_DEP_SET be the set of packages whose message types you use in
##   your messages/services/actions (e.g. std_msgs, actionlib_msgs, ...).
## * In the file package.xml:
##   * add a build_depend tag for "message_generation"
##   * add a build_depend and a run_depend tag for each package in MSG_DEP_SET
##   * If MSG_DEP_SET isn't empty the following dependency has been pulled in
##     but can be declared for certainty nonetheless:
##     * add a run_depend tag for "message_runtime"
## * In this file (CMakeLists.txt):
##   * add "message_generation" and every package in MSG_DEP_SET to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * add "message_runtime" and every package in MSG_DEP_SET to
##     catkin_package(CATKIN_DEPENDS ...)
##   * uncomment the add_*_files sections below as needed
##     and list every .msg/.srv/.action file to be processed
##   * uncomment the generate_messages entry below
##   * add every package in MSG_DEP_SET to generate_messages(DEPENDENCIES ...)

## Generate messages in the 'msg' folder
# add_message_files(
#   FILES
#   Message1.msg
#   Message2.msg
# )

## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )

## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )
```

```
## Generate added messages and services with any dependencies listed here
# generate_messages(
#   DEPENDENCIES
#   std_msgs  # Or other packages containing msgs
# )

################################################
## Declare ROS dynamic reconfigure parameters ##
################################################

## To declare and build dynamic reconfigure parameters within this
## package, follow these steps:
## * In the file package.xml:
##   * add a build_depend and a run_depend tag for "dynamic_reconfigure"
## * In this file (CMakeLists.txt):
##   * add "dynamic_reconfigure" to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * uncomment the "generate_dynamic_reconfigure_options" section below
##     and list every .cfg file to be processed

## Generate dynamic reconfigure parameters in the 'cfg' folder
# generate_dynamic_reconfigure_options(
#   cfg/DynReconf1.cfg
#   cfg/DynReconf2.cfg
# )

##################################
## catkin specific configuration ##
##################################
## The catkin_package macro generates cmake config files for your package
## Declare things to be passed to dependent projects
## INCLUDE_DIRS: uncomment this if you package contains header files
## LIBRARIES: libraries you create in this project that dependent projects also ▼134
   134▲ need
## CATKIN_DEPENDS: catkin_packages dependent projects also need
## DEPENDS: system dependencies of this project that dependent projects also need
catkin_package(
  INCLUDE_DIRS include
  ${catkin_INCLUDE_DIRS}
  ${OpenCV_INCLUDE_DIRS}
  ${PCL_INCLUDE_DIRS}
# ${VISP_INCLUDE_DIRS}
#  LIBRARIES coslam_vrep
#  CATKIN_DEPENDS other_catkin_pkg
#  DEPENDS system_lib
)

###########
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
```

```
#include_directories(include)
include_directories(include ${PCL_INCLUDE_DIRS})

## Declare a C++ library
#add_library(${PROJECT_NAME}
# src/${PROJECT_NAME}/coslam_vrep.cpp
# src/${PROJECT_NAME}/vlp16.cpp
#)
link_directories(${PCL_LIBRARY_DIRS})

## Add cmake target dependencies of the library
## as an example, code may need to be generated before libraries
## either from message generation or dynamic reconfigure
# add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${ ▼167
    167▲ catkin_EXPORTED_TARGETS})
add_definitions(${PCL_DEFINITIONS})


## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/coslam_vrep_node.cpp)
add_executable(vlp16 source/vlp16.cpp)
add_executable(rbctl source/control_robot.cpp)
if (TEST)
  add_executable(test source/_ros_node.cpp)
  add_executable(test_velodyne source/test_velodyne.cpp)
endif()

## Rename C++ executable without prefix
## The above recommended prefix causes long target names, the following renames  ▼183
    183▲ the
## target back to the shorter version for ease of user use
## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg  ▼185
    185▲ someones_pkg_node"
# set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX  ▼186
    186▲ "")

## Add cmake target dependencies of the executable
## same as for the library above
# add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${ ▼190
    190▲ catkin_EXPORTED_TARGETS})

## Specify libraries to link a library or executable target against
#target_link_libraries(${PROJECT_NAME}_node
if (TEST)
  target_link_libraries(test
    ${catkin_LIBRARIES}
  )

  target_link_libraries(test_velodyne
    ${catkin_LIBRARIES}
  )
```

```
endif()

target_link_libraries(vlp16
  ${catkin_LIBRARIES}
  ${PCL_INCLUDE_DIRS}
)

## All targets below are compiled with these additional flags
## @reference https://cmake.org/cmake/help/v3.3/command/add_compile_options.html
add_compile_options(-std=c++11)
target_link_libraries(rbctl
  ${catkin_LIBRARIES}
)

#############
## Install ##
#############

# all install targets should use catkin DESTINATION variables
# See http://ros.org/doc/api/catkin/html/adv_user_guide/variables.html

## Mark executable scripts (Python etc.) for installation
## in contrast to setup.py, you can choose the destination
catkin_install_python(PROGRAMS
  scripts/python/stereo_vision.py
  scripts/python/stereo_image_preprocessor.py
  scripts/python/licp.py
  # scripts/python/test.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)

## Mark executables and/or libraries for installation
# install(TARGETS ${PROJECT_NAME} ${PROJECT_NAME}_node
#   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark cpp header files for installation
# install(DIRECTORY include/${PROJECT_NAME}/
#   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
#   FILES_MATCHING PATTERN "*.h"
#   PATTERN ".svn" EXCLUDE
# )

## Mark other files for installation (e.g. launch and bag files, etc.)
install(FILES
  camera01.launch
  camera02.launch
  stereo_vision.launch
  vrep.launch
  vrep_playback.launch
  vrep_session_latest.launch
```

```
  vrep_session.launch
  vrep_ugv_01.launch
  vrep_ugv_02.launch
  vrep_ugv_03.launch
  vrep_ugv_04.launch
  vrep_ugv_05.launch
  vrep_ugv_uav_01.launch
  DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
)

##############
## Testing ##
##############

## Add gtest based cpp test target and link libraries
# catkin_add_gtest(${PROJECT_NAME}-test test/test_coslam_vrep.cpp)
# if(TARGET ${PROJECT_NAME}-test)
#   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_NAME})
# endif()

## Add folders to be run by python nosetests
# catkin_add_nosetests(test)
```

Script 93: C++ source file: control_robot.cpp

```
/*!
 * @file
 * @author Ernest Skrzypczyk
 *
 * @date 20.05.17
 *
 * @brief Control of robot using JointState and JointSetStateData message formats
 *
 * Command prototype: <b>rosrun coslam_vrep control_robot _rate:=25 _samplesize:=4 _sampling:=0</b>
 * @param rate: Sampling frequency of the node <!25 [Hz]>
 * @param samplesize: Number of elements that are averaged/subsampled <!4 [1]>
 * @param sampling: Selects if the raw data should be subsampled after a certain delay or averaged over a certain period <!0 [1]>
 *    - sampling 0 sets subsampling
 *    - sampling !0 sets averaging with given number of samples
 *
 * DESCRIPTION
 * Publish control using JointState and JointSetStateData message formats for robot either directly or based on subscribed state.
 *
 * Project github repository
 *
 * @see https://github.com/em-er-es/coslam/coslam_vrep
 *
 */

// Headers
#include <iostream>
#include <ros/ros.h>
#include <sstream>
#include <string>
#include <std_msgs/String.h>
#include <sensor_msgs/JointState.h>
#include <vrep_common/simRosGetObjectHandle.h>
#include <vrep_common/JointSetStateData.h>

// Package parameters and configuration
#include "coslam_vrep/coslam_vrep.hpp"
// Node specific parameters and configuration
//#include "coslam_vrep/.hpp"

// #S#
//! @todo Integrate preprocessor definitions into main package and node headers
#ifndef DEBUG
  #define DEBUG 0
#endif
#define NN_NODE "rbctl"
#define TP_RBCTL_I_1 "/vrep/joints/robot01/state"
#define TP_RBCTL_O_1 "/vrep/joints/robot01/control"
// #E#
```

```
50 // Global variables updated in the SubscriberCallback function, processed and  ▼50
       50▲ published.

   //! Node name using console codes
   // char NodeName[64] = C2 NN_NODE CR; // The size is necessary for the GNU/Linux  ▼53
       53▲ console codes //~COLOR
   char NodeName[16] = NN_NODE; // The size is necessary for the GNU/Linux console  ▼54
       54▲ codes //~COLORLESS
55 // char NodeName[20] = PP; // The size is necessary for the GNU/Linux console codes ▼55
       55▲  //~COLOR8

   ros::Subscriber Sub;
   ros::Publisher Pub;

60 // # Topics
   //! Topic holding robot01 state
   char Topic1[64] = TP_RBCTL_I_1;
   //! Topic holding robot01 control
   char Topic2[64] = TP_RBCTL_O_1;
65


   void subscriberCallback(const std_msgs::StringPtr& msg) {
   // void subscriberCallback(const sensor_msgs::JointStateConstPtr& msg) {
     int x = 01;
70   ROS_INFO("[%s][%s][Sub][x]: %d", PKGNAME, NN_NODE, x);
   //! @todo Add the auto read structure from sensorbased example
     // ROS_INFO("[%s][%s][Sub][joint][position][velocity][effort]: %s, %.2f, %.2f,  ▼72
       72▲ %.2f", PKGNAME, NN_NODE, msg->name[0], msg->position[0], msg->velocity, msg ▼72
       72▲ ->effort);
   }

75 void Publisher(const std_msgs::StringPtr& msg) {
   // void publisherCall(const sensor_msgs::JointStatePtr& msg) {
     int x = 10;
     // Pub.publish(x);
     ROS_INFO("[%s][%s][Pub][x]: %d", PKGNAME, NN_NODE, x);
80 }


   int main(int argc, char **argv)
   {
85 //! ## Initialization
   ros::init(argc, argv, NN_NODE); //! Name of the preprocessor node
   ros::start(); //! ROS internal function, neccessary to be called

   float rate_frequency = 25;
90

   ros::NodeHandle Node;

   //! - Subscriber initialization with topic, queue size and callback function
   ros::Subscriber Sub = Node.subscribe(TP_RBCTL_I_1, 1, subscriberCallback);
95

   //! - Publisher initialization with topic, message format and queue size
```

```
// ros::Publisher Pub = Node.advertise<std_msgs::String>(TP_RBCTL_O_1, 1);
// ros::Publisher Pub = Node.advertise<sensor_msgs::JointState>(TP_RBCTL_O_1, 1);
ros::Publisher Pub = Node.advertise<vrep_common::JointSetStateData>(TP_RBCTL_O_1, 99);

//! - Publishing rate [Hz]
ros::Rate frequency(rate_frequency);

//! - Publisher variables for conventional messages
std_msgs::String jointName;
//! - Nodehandle for subscriber and publisher
sensor_msgs::JointState robot01State;
vrep_common::JointSetStateData robot01Control;

//! Control position
double position = 0;

//! Control velocity
double velocity = 0;

//! Control effort
double effort = 0;

//! Loop counter holder
unsigned int LoopCounter = 0;
unsigned int lc = 0;
bool flag01 = 0;

//! ## Main loop
do {
  //! @todo Add the service call structure from sensorbased example
  //! @todo Add dummy message contruction
  ros::ServiceClient client = Node.serviceClient<vrep_common::simRosGetObjectHandle>("/vrep/simRosGetObjectHandle");
  client.waitForExistence();
  vrep_common::simRosGetObjectHandle vrepService;

  for (auto joint: {"Pioneer_p3dx_leftMotor", "Pioneer_p3dx_rightMotor"}) {
    ros::spinOnce();
    // srv.request.objectName = joint;
    std::cout << "Handle for " << joint << std::endl;
    // if (client.call(vrepService)) {
    //  std::cout << "Handle for " << joint << ": " << vrepService.response.handle << std::endl;
    //  joint_setpoint_.handles.data.push_back(srv.response.handle);
    //  joint_setpoint_.setModes.data.push_back(2);
    //  joint_setpoint_.values.data.push_back(0);
    // }
  }

  /* code reference
  std::vector<std::string> topics;
  topics.push_back(std::string("/lsd_slam/liveframes"));
```

```
  topics.push_back(std::string("/lsd_slam/keyframes"));
  topics.push_back(std::string("/lsd_slam/graph"));
  */

  /*
  robot01Control.values.data[0] = position;
  robot01Control.values.data[1] = velocity;
  robot01Control.values.data[2] = effort;
  robot01Control.name = std_msgs::String "testjoint";
  robot01Control.name = "testjoint";
  */
//  Pub.publish(robot01Control);
//  publisherCall(robot01Control);

  /*
  double robotx = 0;
  double roboty = 0;
  double robott = 0;

  double targetx = 10;
  double targety = -5;
  double targett = 90;

  double et = targett - robott;
  float kp = 0.1;
  float kv = 1.0;
  float tolerance = 5;

  float robot01wheelLlimit = 10;
  float robot01wheelRlimit = 10;

  if (std::abs(et) > tolerance) {
    std::cout << "Robot01 correcting heading" << std::endl;
    float robot01wheelL = kp * et;
    float robot01wheelR = - kp * et;

    if (robot01wheelL >= robot01wheelLlimit) robot01wheelL = robot01wheelLlimit;
    else if (robot01wheelL <= -robot01wheelLlimit) robot01wheelL = - ▼184
    184▲ robot01wheelLlimit;
    if (robot01wheelR >= robot01wheelRlimit) robot01wheelR = robot01wheelRlimit;
    else if (robot01wheelR <= -robot01wheelRlimit) robot01wheelR = - ▼186
    186▲ robot01wheelRlimit;

  } else {
    std::cout << "Robot01 reached heading" << std::endl;
    double dx = targetx - robotx;
    double dy = targety - roboty;
    double d = std::sqrt(dx * dx + dy * dy);

    float robot01wheelL = kv * d;
    float robot01wheelR = kv * d;

  }
```

```
  std::cout << "Robot01 velocities: L = " << robot01wheelL << ", R = " <<  ▼199
    199▲ robot01wheelR << std::endl;
200 // */

  // if (sampling != 0) frequency.sleep();
  // else if (flag01 == 0) {
  //  lc = LoopCounter;
205 //  flag01 = 1;
  // }
  // if (LoopCounter - lc > sampling) {
  //  frequency.sleep();
  //  flag01 = 0;
210 // }
  //! - Increase or reset loop counter
  if (LoopCounter == UINT_MAX) LoopCounter = 0; else LoopCounter++;

} while (ros::ok());
215 //! ## Main loop end

return 0;
}
```

Script 94: C++ source file: _ros_node.cpp

```
/*!
 * @file
 * @author Ernest Skrzypczyk
 *
 * @date DD.MM.YY
 *
 * @brief Skeleton node
 *
 * Command prototype: <b>rosrun coslam_vrep node _rate:=25 _samplesize:=4 _sampling ▼9
   9▲ :=0</b>
 * @param rate: Sampling frequency of the node <!25 [Hz]>
 * @param samplesize: Number of elements that are averaged/subsampled <!4 [1]>
 * @param sampling: Selects if the raw data should be subsampled after a certain  ▼12
   12▲ delay or averaged over a certain period <!0 [1]>
 *    - sampling 0 sets subsampling
 *    - sampling !0 sets averaging with given number of samples
 *
 * DESCRIPTION
 * Filter the raw data from Velodyne Puck (VLP16) system and publish it.
 *
 * Project github repository
 *
 * @see https://github.com/em-er-es/coslam/coslam_vrep
 *
 */

// Headers
#include <iostream>
#include <ros/ros.h>
#include <sstream>
#include <string>
#include <std_msgs/String.h>
/*
// System
#include <cstdint>
#include <stdint.h>
#include <inttypes.h> //! Definitions: uint32_t
// ROS
#include "geometry_msgs/Pose.h"
#include "geometry_msgs/Pose2D.h"
#include "sensor_msgs/PointCloud2.h"
#include "std_msgs/String.h"
#include "tf/tf.h"
// Velodyne messages
#include "velodyne_msgs/VelodyneScan.h" // Unused
#include "velodyne_msgs/VelodynePacket.h" // Unused
// PCL
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/conversions.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
```

```
#include <pcl/PCLPointCloud2.h>
#include <pcl_conversions/pcl_conversions.h>
#include <pcl/filters/voxel_grid.h>
*/
// Package parameters and configuration
#include "coslam_vrep/coslam_vrep.hpp"
// Node specific parameters and configuration
//#include "coslam_vrep/.hpp"


/* TODO
 * FIX DOXYGEN documentation
// */

// #S#
#define NN_NODE "test"
#define DEBUG_LC 100000 //!< Debug loop counter filter
#define TP_NODE_1 "topic1"
#define TP_NODE_2 "topic2"
// #E#

// Global variables updated in the SubscriberCallback function, processed and  ▼72
    72▲ published.

//! Node name using console codes
// char NodeName[64] = C1 NN_NODE CR; // The size is necessary for the GNU/Linux  ▼75
    75▲ console codes //~COLOR
char NodeName[16] = NN_NODE; // The size is necessary for the GNU/Linux console  ▼76
    76▲ codes //~COLORLESS
// char NodeName[20] = PP; // The size is necessary for the GNU/Linux console codes ▼77
    77▲  //~COLOR8

ros::Subscriber Sub;
ros::Publisher Pub;

//! Global variable
uint32_t x;

// # Topics
//! Topic holding VLP16 raw scan data
char Topic1[64] = TP_NODE_1;
//! Topic holding VLP16 point cloud data in PointCloud2 format
char Topic2[64] = TP_NODE_2;


/*!
/*!
 * @brief Subscriber callback
 *
 * Subscribe to motion capture data from @ref mocap_optitrack node and read  ▼96
    96▲ position and orientation from Optitrack node.
 *
 * @param msg Message generated by @ref mocap_optitrack node in format:
```

```
 * - Position x [m]
 * - Position y [m]
 * - Orientation [rad]
 *
 * @see https://github.com/ros-drivers/mocap_optitrack
 *
 * @return NULL
 */

void subscriberCallback(const std_msgs::StringPtr& msg) {
  x++;
  ROS_INFO("[%s][%s][Sub][x]: %d", PKGNAME, NN_NODE, x);
}

/*!
/*!
 * @brief Publisher
 *
 * Publish.
 *
 * @param msg Message generated by @ref mocap_optitrack node in format:
 * - Position x [m]
 * - Position y [m]
 * - Orientation [rad]
 *
 * @see https://github.com/ros-drivers/mocap_optitrack
 *
 * @return NULL
 */

void Publisher(const std_msgs::StringPtr& msg) {
  x--;
  Pub.publish(x);
  ROS_INFO("[%s][%s][Pub][x]: %d", PKGNAME, NN_NODE, x);
}

/*!
/*!
 * @brief Node main
 *
 * Initialize variables, nodehandle, subscribe to motion capture data from @ref  ▼139
    139▲ mocap_optitrack node and publish position and orientation after  ▼139
    139▲ processing with time stamp.
 * The position and orientation are published along with timestamp in custom  ▼140
    140▲ defined message format @ref rollo::Pose2DStamped.
 *
 * @param rate: Sampling frequency of the node <!25 [Hz]>
 * @param samplesize: Number of elements that are averaged/subsampled <!4 [1]>
 * @param sampling: Selects if the raw data should be subsampled after a certain  ▼144
    144▲ delay or averaged over a certain period <!0 [1]>
 *    - sampling 0 sets subsampling
 *    - sampling !0 sets averaging
 *
```

```
 * @return 0
 */

int main(int argc, char **argv)
{

//! ## Initialization
ros::init(argc, argv, NN_NODE); //! Name of the preprocessor node
ros::start(); //! ROS internal function, neccessary to be called

//! - Nodehandle for subscriber and publisher
ros::NodeHandle Node;

//! - Subscriber initialization with topic, queue size and callback function
ros::Subscriber Sub = Node.subscribe(TP_NODE_1, 1, subscriberCallback);

//! - Publisher initialization with topic, message format and queue size
ros::Publisher Pub = Node.advertise<std_msgs::String>(TP_NODE_2, 1);

//! - Node arguments using command line
int rate_frequency;
int samplesize;
int sampling; //! Sampling is either done using subsampling (0) or simple  ▼170
    170▲ averaging (1)

//! - Initialize node parameters from launch file or command line.
//! Use a private node handle so that multiple instances of the node can be run  ▼173
    173▲ simultaneously
//! while using different parameters.
/*! @todo Check if private_node_handle_ is really necessary as name */
/*! @todo Check if private_node_handle_ is really necessary as name */
ros::NodeHandle NodeParameters("~");
//! Set default values for node parameters
NodeParameters.param("rate", rate_frequency, int(25));
NodeParameters.param("samplesize", samplesize, int(4));
NodeParameters.param("sampling", sampling, int(0));

//! - Publishing rate [Hz]
ros::Rate frequency(rate_frequency);

//! - Publisher variables for conventional messages
std_msgs::String PubMessage;
std::stringstream StringStream;

//! - Loop counter holder
unsigned int LoopCounter = 0;

//! ## Main loop
do {
  if (! (LoopCounter % DEBUG_LC)) ROS_INFO("[%s][%s][Debug][Counter]: %d", PKGNAME ▼195
    195▲ , NN_NODE, LoopCounter); // #DB#
  if (! (LoopCounter % DEBUG_LC)) ROS_INFO("[%s][%s][Debug][Sampling paramter]: %d ▼196
    196▲ ", PKGNAME, NN_NODE, sampling); // #DB#
```

```
    ros::spinOnce();

    //! For averaging sleep for time defined by rate before reading states from the
200 PubMessage.data = "test" + std::to_string(x);
    Pub.publish(PubMessage);
    if (sampling != 0) frequency.sleep();
    //! - Increase loop counter
    LoopCounter++;
205

} while (ros::ok());
//! ## Main loop end

return 0;
210 }
```

Script 95: C++ source file: test_velodyne.cpp

```
#include <ros/ros.h>
#include <iostream>
// PCL specific includes
#include <sensor_msgs/PointCloud2.h>
#include <pcl_conversions/pcl_conversions.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>

#include <pcl/filters/voxel_grid.h>

ros::Publisher pub;

void
cloud_cb (const sensor_msgs::PointCloud2ConstPtr& inputCloud)
{
  // ### Filter
  // #### Initialization
  // Shared pointer to PCL point cloud in XYZ format
  pcl::PointCloud<pcl::PointXYZ>::Ptr vlp16pclPointCloud(new pcl::PointCloud<pcl:: ▼19
    19▲ PointXYZ>); // creates a shared pointer
  pcl::PointCloud<pcl::PointXYZ>::Ptr vlp16pclPointCloudFiltered(new pcl:: ▼20
    20▲ PointCloud<pcl::PointXYZ>); // creates a shared pointer
  // Shared pointer to PCL point cloud in XYZI format
  // pcl::PointCloud<pcl::PointXYZI>::Ptr pclCloud(new pcl::PointCloud<pcl:: ▼22
    22▲ PointXYZI>); // creates a shared pointer
  // Convert from PointCloud2 format message to PCL format
  pcl::fromROSMsg(*inputCloud, *vlp16pclPointCloud);

/*
  unsigned int j = 0;
  for (size_t i = 0; i < vlp16pclPointCloud->points.size(); ++i) {
    if ((vlp16pclPointCloud->points[i].x > 16)
        || (vlp16pclPointCloud->points[i].y > 16)
        || (vlp16pclPointCloud->points[i].z > 16)
      // ) {
        || (std::abs(vlp16pclPointCloud->points[i].x) < 0.01)
        || (std::abs(vlp16pclPointCloud->points[i].y) < 0.01)
        || (std::abs(vlp16pclPointCloud->points[i].z) < 0.01)) {
         // Remove point meeting condition
         vlp16pclPointCloud->erase(vlp16pclPointCloud->begin() + i);
         // Assuming an unorganized point cloud with height = 1
         vlp16pclPointCloud->width--;
         j++;
      }
    // Resize filtered cloud if necessary
    if (j > 0) vlp16pclPointCloud->resize(vlp16pclPointCloud->points.size() - j);
  }
// */

// /*
  unsigned int j = 0;
  vlp16pclPointCloudFiltered->height = 1;
```

```
  vlp16pclPointCloudFiltered->width = 0;
  vlp16pclPointCloudFiltered->header.frame_id = "frame00";
  for (size_t i = 0; i < vlp16pclPointCloud->points.size(); ++i) {
    if ((vlp16pclPointCloud->points[i].x < 16)
       && (vlp16pclPointCloud->points[i].y < 16)
       && (vlp16pclPointCloud->points[i].z < 16)
      // ) {
       && (std::abs(vlp16pclPointCloud->points[i].x) > 0.01)
       && (std::abs(vlp16pclPointCloud->points[i].y) > 0.01)
       && (std::abs(vlp16pclPointCloud->points[i].z) > 0.02)) {
        // Remove point meeting condition
        vlp16pclPointCloudFiltered->push_back(vlp16pclPointCloud->points[i]);
        // Assuming an unorganized point cloud with height = 1
        vlp16pclPointCloudFiltered->width++;
        j++;
      }
    // Resize filtered cloud if necessary
    if (j > 0) vlp16pclPointCloudFiltered->resize(vlp16pclPointCloudFiltered-> ▼67
    67▲ points.size());
  }
// */

  // bool vg = false;
  bool vg = true;
/*
  if (vg) {
    //! Create filter object
    pcl::VoxelGrid<pcl::PointXYZ> voxelgrid;
    //! Set input
    voxelgrid.setInputCloud(vlp16pclPointCloud);
    //! Set downsample
    voxelgrid.setDownsampleAllData(false);
    voxelgrid.setDownsampleAllData(false);
    //! Set leaf object parameters
    //! @todo add node arguments for more control over downsampling
    voxelgrid.setLeafSize(0.0001f, 0.0001f, 0.00001f);
    // voxelgrid.setLeafSize(0.00001f, 0.00001f, 0.000025f); // Too much
    //! Set output and filter
    // voxelgrid.filter(*vlp16pclPointCloud); // Results in minor noise, since  ▼87
    87▲ overridden
    voxelgrid.filter(*vlp16pclPointCloudFiltered);
  } else {
    pcl::copyPointCloud(*vlp16pclPointCloud, *vlp16pclPointCloudFiltered);
  }
// */

  // ### Publish processed data
  // Create a new PointCloud2 object
  sensor_msgs::PointCloud2 outputCloud;

    // Convert from PCL to PointCloud2 format
  if (vg) pcl::toROSMsg(*vlp16pclPointCloudFiltered, outputCloud); else pcl:: ▼99
    99▲ toROSMsg(*vlp16pclPointCloud, outputCloud);
```

```
  // pcl::toROSMsg(*vlp16pclPointCloud, outputCloud);
  size_t b = vlp16pclPointCloud->size();
  size_t a = vlp16pclPointCloudFiltered->size();
  std::cout << "Point cloud size before:\t" << vlp16pclPointCloud->size() << "\ ▼103
    103▲ tPoint cloud size after:\t" << vlp16pclPointCloudFiltered->size() << "\ ▼103
    103▲ tReduction:\t" << float(b - a) / b * 100 << "%" << std::endl;
  // Publish filtered message
  pub.publish(outputCloud);
}

int
main (int argc, char** argv)
{
  // Initialize ROS
  ros::init (argc, argv, "tvlp16");
  ros::NodeHandle nh;

  // Create a ROS subscriber for the input point cloud
  ros::Subscriber sub = nh.subscribe<sensor_msgs::PointCloud2> ("/vrep/vlp16/pcl2 ▼116
    116▲ ", 1, cloud_cb);

  // Create a ROS publisher for the output point cloud
  pub = nh.advertise<sensor_msgs::PointCloud2> ("/vlp16/pcl2filtered", 1);

  // Spin
  ros::spin ();
}
```

Script 96: C++ source file: vlp16.cpp

```
/*!
 * @file
 * @author Ernest Skrzypczyk
 *
 * @includedoc headers/header2.cpp
 *
 * @date 28.04.17
 *
 * @brief Preprocessor for Velodyne Puck (VLP16) measurements
 *
 * @details Filter the simuated or raw data from Velodyne Puck (VLP16) system and ▼11
   11▲ publish it.
 *
 * @cmdp{$, rosrun coslam_vrep vlp16 _rate:=25 _samplesize:=4 _sampling:=0}
 *
 * @param bool filterNoise: Use noise filtering <!1>
 * @param float filternoisethreshold: Threshold for noise filter <!1 [int]>
 * @param bool filtervoxelgrid: Use VoxelGrid filtering <!true [bool]>
 * @param int rate: Sampling frequency of the node <!25 [Hz]>
 * @param int samplesize: Number of elements that are averaged/subsampled <!4 [1]>
 * @param queue:
 *     - int input 1 subscriber queue size
 *     - int output 1 publisher queue size
 *     - int parts 2 number of partial clouds of a full scan
 *
 * @repository https://github.com/em-er-es/coslam/coslam_vrep
 *
 * @done Properly implement parameter setting from server, currently global ▼27
   27▲ filterNoise is not being updated from initial declaration
 *
 * @todo FIX DOXYGEN documentation
 *
 * @todo Fix quadrant detection issue by making sure messages are updated if the ▼31
   31▲ same quadrant is used and message counter is increased only if buffer ▼31
   31▲ increased, change queueSizeInput to something different for PCL merging ▼31
   31▲ like inputParts
 *
 * @todo Aquire 4 messages or adjust V-REP VLP16 Lua script to get a full point ▼33
   33▲ cloud
 *
 * @todo Alternatively take one partial cloud at a time, it might be more efficient
 *
 * @todo VoxelGrid filter does not deliver useful output in current scenario, it ▼37
   37▲ makes more sense to call it after the full pointcloud is constructed
 *
 * @todo Implement raw message processing; Probably just convert to PointCloud2 and ▼39
   39▲  then continue with the developed algorithm
 *
 * @todo Implement pointCloud2 to octree
 *
 */
```

```
45
// # Headers
// System
#include <cmath>
/* // #S#
50 #include <cstdint> // Requires C++11
// #E# */
#include <stdint.h>
#include <iostream>
#include <inttypes.h> // Definitions: uint32_t
55 #include <sstream>

// ROS
#include <ros/ros.h>
// ROS: PointCloud2 message format
60 #include <sensor_msgs/PointCloud2.h>
// /*
// ROS: Velodyne message format
#include "velodyne_msgs/VelodyneScan.h" // Unused
#include "velodyne_msgs/VelodynePacket.h" // Unused
65 // */
/* // #S#
#include "geometry_msgs/Pose.h"
#include "geometry_msgs/Pose2D.h"
#include "sensor_msgs/PointCloud2.h"
70 #include "std_msgs/String.h"
#include "tf/tf.h"
#include <message_filters/subscriber.h>
#include <message_filters/time_synchronizer.h>
// #E# */
75

// PCL
/* // #S#
#include <pcl/PolygonMesh.h>
#include <pcl/common/common.h>
80 #include <pcl/common/common_headers.h>
// #E# */
// #include <pcl_ros/transforms.h> // #D#
#include <pcl_conversions/pcl_conversions.h>
#include <pcl/conversions.h>
85 #include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
#include <pcl/PCLPointCloud2.h>
90 #include <pcl/filters/voxel_grid.h>

// Package parameters and configuration
#include "coslam_vrep/coslam_vrep.hpp"

95 // Node specific parameters and configuration
#include "coslam_vrep/vlp16.hpp"
```

```
// # Global declarations
// Node name using console codes
#if COLORCODES == 1
  //! Node name -- Color version
  char NodeName[64] = C1 NN_VLP16 CR; // The size is necessary for the GNU/Linux  ▼103
    103▲ console codes //~COLOR
#else
  //! Node name
  //! @done Use color codes properly
  char NodeName[16] = NN_VLP16; // The size is reduced without GNU/Linux console  ▼107
    107▲ codes //~NCOLOR
#endif

//! Subcriber instance
ros::Subscriber SubscriberPCL2;
//! Publisher instance
ros::Publisher PublisherPCL2;

//! Output cloud object
pcl::PointCloud<pcl::PointXYZ>::Ptr vlp16pclPointCloudOutput(new pcl::PointCloud< ▼116
    116▲ pcl::PointXYZ>); // Creates a shared pointer
pcl::PointCloud<pcl::PointXYZ>::Ptr vlp16pclPointCloudBuffer(new pcl::PointCloud< ▼117
    117▲ pcl::PointXYZ>); // Creates a shared pointer

// ## Counters
//! Output message counter
unsigned short messagecounter = 0;

//! Quadrant counter
unsigned short quadrantcounter = 0;

// ## Flags
//! Passthrough/relay flag
bool flagrelay = 0;

// ## Settings
//! Input partial point clouds
unsigned int inputParts = 2; // 2 is sufficient for 360 degrees in current  ▼132
    132▲ implementation of VLP16 model in V-REP
//! Input queue size for PCL2 message
unsigned int queueSizeInput = 1;
//! Output queue size for PCL2 message
unsigned int queueSizeOutput = 1;
//! Output point cloud frame identificator
// std::string frameId = "frame00"; // World frame
std::string frameId = "vlp16"; // Sensor frame

//! Noise filtering switch
bool filterNoise = 1;
//! Noise filtering theshold
// float filterNoiseThreshold = 0.02; // Filters floor completely
float filterNoiseThreshold = 0.001; // Captures floor
```

```
//! Noise filtering theshold for maximum practicle sensor distance
float filterNoiseThresholdMax = 16; // Maximum in V-REP simulations
// float filterNoiseThresholdMax = 4;
//! VoxelGrid filtering switch
bool filterVoxelGrid = 0;
//! VoxelGrid downsample switch
bool filterVoxelGridDownsample = 0;
//! @{ Leaf size parameters in X, Y and Z
float lx = 0.1;
float ly = 0.1;
float lz = 0.1;
//! @}

// # Functions
/*!
 * @brief Subscriber callback for PointCloud2 message format
 *
 * Subscribe to VLP16 range laser data from @ref vrep.dox simulation and filter.
 *
 * @todo Fix Doxygen documentation
 * * replace vrep.dox with link name
 *
 * @param inputCloud Message generated by @ref vrep.dox
 * - Assumed to be unorganized, ergo inputCloud.height = 1
 *
 * @see http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
 * @see http://www.pointclouds.org/documentation/tutorials/voxel_grid.php
 * @see http://wiki.ros.org/voxel_grid
 *
 * @todo Implement downsample for VoxelGrid filter
 *
 * @return NULL
 */

void SubscriberCallbackRaw(const velodyne_msgs::VelodyneScan& inputRawCloud) {

}

/*!
 * @brief Subscriber callback for PointCloud2 message format
 *
 * Subscribe to VLP16 range laser data from @ref vrep.dox simulation and filter.
 *
 * @todo Fix Doxygen documentation
 * * replace vrep.dox with link name
 *
 * @param inputCloud Message generated by @ref vrep.dox
 * - Assumed to be unorganized, ergo inputCloud.height = 1
 *
 * @see http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html
 * @see http://www.pointclouds.org/documentation/tutorials/voxel_grid.php
 * @see http://wiki.ros.org/voxel_grid
 *
```

```
 * @todo Implement downsample for VoxelGrid filter
 *
 * @return NULL
 *
 * @include test.dox
 *
 * @include test.tex
 *
 * @includedoc test.tex
 *
 * @includedoc test.dox
 *
 * \f{algorithm}\begin{algorithmic}\STATE \$test \leftarrow 1\$ \end{algorithmic}\ f}
 *
 */

void SubscriberCallbackPCL2(const sensor_msgs::PointCloud2ConstPtr& inputCloud) {
// void SubscriberCallbackPCL2(const sensor_msgs::PointCloud2ConstPtr& inputCloud1, const sensor_msgs::PointCloud2ConstPtr& inputCloud2) {
// void SubscriberCallbackPCL2(const sensor_msgs::PointCloud2ConstPtr& inputCloud, const sensor_msgs::PointCloud2ConstPtr& inputCloud2) {
  // ### Parameters
  // Check ROS parameter server for settings
  ros::param::get("~filter/noise/sw", filterNoise);
  ros::param::get("~filter/voxelgrid/sw", filterVoxelGrid);

/* #S#
  //! @todo Implement dynamic queue
  int t = queueSizeInput;
  ros::param::get("~queue/input", queueSizeInput);
#if VERBOSE>0
  if (t != queueSizeInput) ROS_INFO("[%s][%s][Settings][Queue]: Input = %d", PKGNAME, NN_VLP16, queueSizeInput);
#endif
// #E# */

  if (filterNoise || filterVoxelGrid) flagrelay = 0; else
  {
  // if (! filterNoise && ! filterVoxelGrid) {
#if VERBOSE>0
    if (! flagrelay) ROS_INFO("[%s][%s][Node]: Working as a relay", PKGNAME, NN_VLP16);
#endif
    flagrelay = 1;
  }

  // ### Relay
  // Relay message and return
  if (flagrelay) {
    // Publish unfiltered message
    PublisherPCL2.publish(inputCloud);
    // Exit without further operations
```

```
    return;
  }

  // Clear output point cloud if first message received
  if (messagecounter == 0) {
    vlp16pclPointCloudOutput->clear();
    // Set unorganized point cloud parameters
    vlp16pclPointCloudOutput->height = 1;
    vlp16pclPointCloudOutput->width = 0;
    // Set world frame reference (frame00)
    vlp16pclPointCloudOutput->header.frame_id = "frame00";
    // Reset quadrant counter
    quadrantcounter = 0;
  }

  // ### Filter
  // #### Initialization
  // Shared pointer to PCL point cloud in XYZ format
  pcl::PointCloud<pcl::PointXYZ>::Ptr vlp16pclPointCloud(new pcl::PointCloud<pcl:: ▼265
    265▲ PointXYZ>); // Creates a shared pointer
  pcl::PointCloud<pcl::PointXYZ>::Ptr vlp16pclPointCloudFiltered(new pcl:: ▼266
    266▲ PointCloud<pcl::PointXYZ>); // Creates a shared pointer
  // Shared pointer to PCL point cloud in XYZI format
  // pcl::PointCloud<pcl::PointXYZI>::Ptr pclCloud(new pcl::PointCloud<pcl:: ▼268
    268▲ PointXYZI>); // creates a shared pointer
  // Convert from PointCloud2 format message to PCL format
  pcl::fromROSMsg(*inputCloud, *vlp16pclPointCloud);

  // Check if input point cloud is unorganized
  if (vlp16pclPointCloud->height != 1) ROS_WARN("[%s][%s][Sub][Warn]: Point cloud ▼273
    273▲ %s is of height %d", PKGNAME, NN_VLP16, TP_VLP16_O_PCL, ▼273
    273▲ vlp16pclPointCloud->height);
  if (vlp16pclPointCloud->isOrganized()) ROS_WARN("[%s][%s][Sub][Warn]: Point ▼274
    274▲ cloud %s is organized", PKGNAME, NN_VLP16, TP_VLP16_O_PCL);

  // Average coordinates to determine quadrant
  float ax = 0;
  float ay = 0;
  float az = 0;
  size_t t;

  // Filter data
  // #### Filter noise start
  //! @section filternoise Filter out noise introduced by V-REP in planes XY, YZ, ▼284
    284▲ ZX located at origin
  if (filterNoise) {
    // Update threshold parameter
    ros::param::get("~filter/noise/threshold", filterNoiseThreshold);

#if VERBOSE>0
    ROS_INFO("[%s][%s][Sub][Noise filtering]: Threshold = %f", PKGNAME, NN_VLP16, ▼290
    290▲ filterNoiseThreshold);
    ROS_INFO("[%s][%s][Sub][Noise filtering][Processing]", PKGNAME, NN_VLP16);
```

```
#endif

    // Iterate over all points
    //!@done Fix noise filtering routine
    //!@todo Try to optimize noise filtering routine
    // Set unorganized point cloud parameters
    vlp16pclPointCloudFiltered->height = 1;
    vlp16pclPointCloudFiltered->width = 0;
    // Set world frame reference (frame00)
    vlp16pclPointCloudFiltered->header.frame_id = "frame00";

    // Iterate over input point cloud
    for (size_t i = 0; i < vlp16pclPointCloud->points.size(); ++i) {

#if DEBUG>1
      bool _a = 0;
      if (! (i % 10)) {
        std::cout << "Processing point " << i
        << ": x = " << vlp16pclPointCloud->points[i].x
        << " y = " << vlp16pclPointCloud->points[i].y
        << " z = " << vlp16pclPointCloud->points[i].z
        << " t = " << filterNoiseThreshold;
      }
#endif

      // Check inverse filtering conditions
      if ((vlp16pclPointCloud->points[i].x < filterNoiseThresholdMax)
         && (vlp16pclPointCloud->points[i].y < filterNoiseThresholdMax)
         && (vlp16pclPointCloud->points[i].z < filterNoiseThresholdMax)
         && (std::abs(vlp16pclPointCloud->points[i].x) > filterNoiseThreshold)
         && (std::abs(vlp16pclPointCloud->points[i].y) > filterNoiseThreshold)
         && (std::abs(vlp16pclPointCloud->points[i].z) > filterNoiseThreshold)) {
          // Add point meeting condition
          vlp16pclPointCloudFiltered->push_back(vlp16pclPointCloud->points[i]);
          // Increase size/width for an unorganized point cloud with height = 1
          vlp16pclPointCloudFiltered->width++;
          ax += vlp16pclPointCloud->points[i].x;
          ay += vlp16pclPointCloud->points[i].y;
          az += vlp16pclPointCloud->points[i].z;

#if DEBUG>1
          if (! (i % 10)) {
            std::cout << " adding" << std::endl; // #DB#
            _a = 1;
          }
#endif

      } // Condition met end

#if DEBUG>1
      if (! (i % 10) && ! _a) {
        std::cout << std::endl; // #DB#
        _a = 0;
```

```
345         }
#endif

        // Set point size temporary variable
        t = i;
350     } // Iteration end

#if DEBUG>0
        // if (! (LoopCounter % DEBUG_LC)) ROS_INFO("[%s][%s][Debug][Counter]: %d",  ▼353
        353▲ PKGNAME, NN_NODE, LoopCounter); // #DB#
        // Casting used for size_t => int conversion
355     ROS_INFO("[%s][%s][Sub][Debug][Height [1], Width [1], Row step [B]]: %d, %d, % ▼355
        355▲ d", PKGNAME, NN_VLP16, vlp16pclPointCloud->height, vlp16pclPointCloud-> ▼355
        355▲ width, static_cast<int>(vlp16pclPointCloud->points.size())); // #DB#
#endif

        // Resize filtered cloud if points were added
        if (vlp16pclPointCloudFiltered->width > 0) vlp16pclPointCloudFiltered->resize( ▼359
        359▲ vlp16pclPointCloudFiltered->points.size());
360     ax /= t;
        ay /= t;
        az /= t;

#if VERBOSE>0
365 /* // #S#
        size_t b = vlp16pclPointCloud->size();
        size_t a = vlp16pclPointCloudFiltered->size();
// #E# */
        unsigned int b = vlp16pclPointCloud->width;
370     unsigned int a = vlp16pclPointCloudFiltered->width;
#endif

#if DEBUG>0
        // Casting used for size_t => int conversion
375     ROS_INFO("[%s][%s][Sub][Debug][Height [1], Width [1], Row step [B]]: %d, %d, % ▼375
        375▲ d", PKGNAME, NN_VLP16, vlp16pclPointCloudFiltered->height,  ▼375
        375▲ vlp16pclPointCloudFiltered->width, static_cast<int>( ▼375
        375▲ vlp16pclPointCloudFiltered->points.size())); // #DB#
#endif

#if VERBOSE>0
        ROS_INFO("[%s][%s][Sub][Noise filtering][PCL reduction]: %d (%4.2f %%)",  ▼379
        379▲ PKGNAME, NN_VLP16, b - a, static_cast<double>(float(b - a) / b * 100));
380     ROS_INFO("[%s][%s][Sub][Noise filtering][Complete]", PKGNAME, NN_VLP16);
#endif

  } // Filter noise end

385 // Update message counter flag
  bool updatecounter = 1;

  //! @todo Display messages only if more than one partials is used
  // Merge point clouds
```

```
390   if (messagecounter == 0) {
        // Copy message into buffer
        *vlp16pclPointCloudBuffer = *vlp16pclPointCloudFiltered;
        messagecounter++;

395 #if VERBOSE>0
        ROS_INFO("[%s][%s][Sub][Merging PCL][Initial message]", PKGNAME, NN_VLP16);
        ROS_INFO("[%s][%s][Sub][Merging PCL] Message counter = %d, input queue = %d",  ▼397
        397▲ PKGNAME, NN_VLP16, messagecounter, queueSizeInput);
      #endif

400   } else if (messagecounter < inputParts) {
        // Merge condition
        // Update quadrant counter
        //! @note Quadrants are relative to the vlp16 frame, which is a refence to  ▼403
        403▲ world frame frame00
        // Check if quadrant counter is already stored
405     if (ax > 0 && ay > 0) {
          if (queueSizeInput == 2 && quadrantcounter & 0b0011) return; if ( !  ▼406
        406▲ quadrantcounter & 0b0001) quadrantcounter |= 1 << 0; // 1st quadrant
        } else if (ax < 0 && ay > 0) {
          if (queueSizeInput == 2 && quadrantcounter & 0b0011) return; if ( !  ▼408
        408▲ quadrantcounter & 0b0010) quadrantcounter |= 1 << 1; // 2nd quadrant
        } else if (ax < 0 && ay < 0) {
410       if (queueSizeInput == 2 && quadrantcounter & 0b1100) return; if ( !  ▼410
        410▲ quadrantcounter & 0b0100) quadrantcounter |= 1 << 2; // 3rd quadrant
        } else if (ax > 0 && ay < 0) {
          if (queueSizeInput == 2 && quadrantcounter & 0b1100) return; if ( !  ▼412
        412▲ quadrantcounter & 0b0100) quadrantcounter |= 1 << 3; // 4th quadrant
        } else {

415   #if VERBOSE>0
        ROS_INFO("[%s][%s][Sub][Merging PCL] Duplicate quadrant message detected.  ▼416
        416▲ Updating", PKGNAME, NN_VLP16);
        // ROS_INFO("[%s][%s][Sub][Merging PCL] Duplicate quadrant message detected",  ▼417
        417▲ PKGNAME, NN_VLP16); // #D#
      #endif

420     // Do not increase counter if same quadrant has been received
        updatecounter = 0;
        }

        // Copy buffer into output
425     *vlp16pclPointCloudOutput = *vlp16pclPointCloudBuffer;
        // Merge pointclouds by adding current point cloud to buffer
        *vlp16pclPointCloudOutput = *vlp16pclPointCloudOutput + * ▼427
        427▲ vlp16pclPointCloudFiltered;
        // Increase message counter if new quadrant message received
        if (updatecounter) messagecounter++;
430
      #if VERBOSE>0
        ROS_INFO("[%s][%s][Sub][Merging PCL] Message counter = %d", PKGNAME, NN_VLP16, ▼432
        432▲  messagecounter);
```

```
#endif

  }

  // Publish once collected all messages
  if (messagecounter == inputParts) {

    // #### Filter VoxelGrid start
    //! @section filtervoxelgrid Perform VoxelGrid filtering, possibly with downsampling
    //! @see http://www.pointclouds.org/documentation/tutorials/voxel_grid.php
    //! For alternative approach
    //! @sa http://wiki.ros.org/voxel_grid
    //! @todo Implement downsample all data
    if (filterVoxelGrid) {

#if VERBOSE>0
      ROS_INFO("[%s][%s][Sub][VoxelGrid filtering][Processing]", PKGNAME, NN_VLP16 );
#endif

      // Clear point cloud buffer
      vlp16pclPointCloudBuffer->clear();
      // Copy merged point cloud into buffer
      *vlp16pclPointCloudBuffer = *vlp16pclPointCloudOutput;
      // Clear output point cloud
      vlp16pclPointCloudOutput->clear();
      // Set width before VoxelGrid filter
      uint32_t prewidth = 0;
      // Create filter object
      pcl::VoxelGrid<pcl::PointXYZ> voxelgrid;
      // Set input based on noise filter
      voxelgrid.setInputCloud(vlp16pclPointCloudBuffer);
      prewidth = vlp16pclPointCloudBuffer->width;
      // Set downsample
      voxelgrid.setDownsampleAllData(filterVoxelGridDownsample);
      // Set leaf object parameters
      //! @todo add node arguments for more control over downsampling
      voxelgrid.setLeafSize(lx, ly, lz);
      // Set output and filter
      voxelgrid.filter(*vlp16pclPointCloudOutput);
      // Set unorganized point cloud parameters
      vlp16pclPointCloudOutput->height = 1;
      vlp16pclPointCloudOutput->width = 0;
      // Set world frame reference (frame00)
      vlp16pclPointCloudOutput->header.frame_id = "frame00";
      // Resize filtered cloud if points were added
      vlp16pclPointCloudOutput->resize(vlp16pclPointCloudOutput->points.size());

#if VERBOSE>0
      ROS_INFO("[%s][%s][Sub][VoxelGrid filtering][PCL reduction]: %d", PKGNAME,  NN_VLP16, (int)prewidth - vlp16pclPointCloudFiltered->width);
      ROS_INFO("[%s][%s][Sub][VoxelGrid filtering][Complete]", PKGNAME, NN_VLP16);
```

```
#endif

485     } // Filter VoxelGrid end

        // ### Publish processed data
        // Create a new PointCloud2 object
        sensor_msgs::PointCloud2 outputCloud;
490
        // Convert from PCL to PointCloud2 format
        pcl::toROSMsg(*vlp16pclPointCloudOutput, outputCloud);

#if VERBOSE>0
495     ROS_INFO("[%s][%s][Sub][Merging PCL] Sending merged message", PKGNAME,  ▼495
        495▲ NN_VLP16);
#endif

        // Publish filtered message
        PublisherPCL2.publish(outputCloud);
500
        // Reset message counter
        messagecounter = 0;
    }
}
505
/*!
 * @brief Initialize variables, nodehandle, subscribe to VLP16 sensor data from  ▼507
    507▲ @ref vrep.dox , filter and publish
 *
 * @param rate: Sampling frequency of the node <!25 [Hz]>
510 * @param samplesize: Number of elements that are averaged/subsampled <!4 [1]>
 * @param sampling: Selects if the raw data should be subsampled after a certain  ▼511
    511▲ delay or averaged over a certain period <!0 [1]>
 *     - sampling 0 sets subsampling
 *     - sampling !0 sets averaging
 *
515 * @return 0
 */

int main (int argc, char** argv) {
    // ### Initialization
520 ros::init(argc, argv, NN_VLP16); // Name of the preprocessor node
    // ros::start(); // ROS internal function, neccessary to be called

    // Nodehandle for subscriber and publisher
    ros::NodeHandle VLP16Node;
525
    // ### Node arguments
    // Frequency rate of node
    int frequencyrate;
    // Sample size
530 int samplesize;
    // Sampling is either done using subsampling (0) or simple average (1)
    int sampling;
```

```
  // ### Initialize node parameters from parameter server, launch file or command line.
  // Use a private node handle so that multiple instances of the node can be run simultaneously while using different parameters.
  ros::NodeHandle NodeParameters("~");

  // ### Clear node parameters
  // NodeParameters.deleteParam("");
  // Node specific
  //! @todo Implement rate, samplesize and sampling approach or drop it completely
  // NodeParameters.param<int>("rate", frequencyrate, int(25));
  //! @todo Implement dynamic queue
  //! @done Implement dynamic input queue for first run
  NodeParameters.param<int>("queue/parts", int(inputParts));
  NodeParameters.param<int>("queue/input", int(queueSizeInput));
  NodeParameters.param<int>("queue/output", int(queueSizeOutput));
  NodeParameters.param<std::string>("frameid", frameId);
  // Filter specific
  // Noise filter specific
  //! @todo Initial filter switches delaration overrides command line values
  //! @done Initial filter switches delaration overrides command line values
  NodeParameters.param<bool>("filter/noise/sw", filterNoise, bool(filterNoise));
  NodeParameters.param<float>("filter/noise/threshold", filterNoiseThreshold, float(filterNoiseThreshold));
  NodeParameters.param<float>("filter/noise/maxdistance", filterNoiseThresholdMax, float(filterNoiseThresholdMax));
  // VoxelGrid filter specific
  NodeParameters.param<bool>("filter/voxelgrid/sw", filterVoxelGrid, bool(filterVoxelGrid));
  NodeParameters.param<float>("filter/voxelgrid/leafsize/x", lx, float(lx));
  NodeParameters.param<float>("filter/voxelgrid/leafsize/y", ly, float(ly));
  NodeParameters.param<float>("filter/voxelgrid/leafsize/z", lz, float(lz));
  NodeParameters.param<bool>("filter/voxelgrid/downsample", filterVoxelGridDownsample, bool(filterVoxelGridDownsample));

  // Update parameter server
  // Save node parameters
  NodeParameters.setParam("queue/parts", int(inputParts));
  NodeParameters.setParam("queue/input", int(queueSizeInput));
  NodeParameters.setParam("queue/output", int(queueSizeOutput));
  NodeParameters.setParam("frameid", frameId);
  NodeParameters.setParam("filter/noise/sw", filterNoise);
  NodeParameters.setParam("filter/noise/threshold", filterNoiseThreshold);
  NodeParameters.setParam("filter/noise/maxdistance", filterNoiseThresholdMax);
  NodeParameters.setParam("filter/voxelgrid/sw", filterVoxelGrid);
  NodeParameters.setParam("filter/voxelgrid/leafsize/x", lx);
  NodeParameters.setParam("filter/voxelgrid/leafsize/y", ly);
  NodeParameters.setParam("filter/voxelgrid/leafsize/z", lz);
  NodeParameters.setParam("filter/voxelgrid/downsample", filterVoxelGridDownsample);

  // ### Communication
```

```
  // //! Subscriber for raw Velodyne messages
580 // ros::Subscriber RawPCL = VLP16Node.subscribe(TopicVLP16Raw, 1024, ▼580
    580▲ SubscriberCallbackRaw);
  // Subscriber for PointCloud2 messages
  SubscriberPCL2 = VLP16Node.subscribe<sensor_msgs::PointCloud2>(TP_VLP16_I_PCL, ▼582
    582▲ queueSizeInput, SubscriberCallbackPCL2);

  // Publisher
585 PublisherPCL2 = VLP16Node.advertise<sensor_msgs::PointCloud2>(TP_VLP16_O_PCL, ▼585
    585▲ queueSizeOutput);

  // ### Main

#if VERBOSE>0
590 ROS_INFO("[%s][%s][Settings][Node]: Frequency rate = %d, Sample size = %d, ▼590
    590▲ Sampling = %d", PKGNAME, NN_VLP16, frequencyrate, samplesize, sampling);
  ROS_INFO("[%s][%s][Settings][Filter]: Noise = %d, VoxelGrid = %d", PKGNAME, ▼591
    591▲ NN_VLP16, filterNoise, filterVoxelGrid);
  if (! filterNoise && ! filterVoxelGrid) ROS_INFO("[%s][%s][Settings][Filtering]: ▼592
    592▲  Disabled", PKGNAME, NN_VLP16);
  else {
    if (filterNoise) ROS_INFO("[%s][%s][Settings][Filter][Noise] Threshold = %f", ▼594
    594▲ PKGNAME, NN_VLP16, filterNoiseThreshold);
595  if (filterVoxelGrid) ROS_INFO("[%s][%s][Settings][Filter][VoxelGrid] X = %f, Y ▼595
    595▲  = %f, Z = %f", PKGNAME, NN_VLP16, lx, ly, lz);
  }
#endif

  if (! filterNoise && ! filterVoxelGrid) {
600 #if VERBOSE>0
    ROS_INFO("[%s][%s][Node]: Working as a relay", PKGNAME, NN_VLP16);
    ROS_INFO("[%s][%s][Settings][Node][Input]: %s", PKGNAME, NN_VLP16, ▼602
    602▲ TP_VLP16_I_PCL);
    ROS_INFO("[%s][%s][Settings][Node][Output]: %s", PKGNAME, NN_VLP16, ▼603
    603▲ TP_VLP16_O_PCL);
#endif
605 flagrelay = 1;
  }

  //! @todo Implement rate, samplesize and sampling approach or drop it completely
  // ROS spin
610 ros::spin();

  // Clear node parameters after exit
  NodeParameters.deleteParam("");

615 return 0;
}
```

Script 97: C++ source file: ros_server_skeleton.cpp

```
// Copyright 2006-2016 Coppelia Robotics GmbH. All rights reserved.
// marc@coppeliarobotics.com
// www.coppeliarobotics.com
//
// -------------------------------------------------------------------
// THIS FILE IS DISTRIBUTED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
// WARRANTY. THE USER WILL USE IT AT HIS/HER OWN RISK. THE ORIGINAL
// AUTHORS AND COPPELIA ROBOTICS GMBH WILL NOT BE LIABLE FOR DATA LOSS,
// DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR
// MISUSING THIS SOFTWARE.
//
// You are free to use/modify/distribute this file for whatever purpose!
// -------------------------------------------------------------------
//
// This file was automatically created for V-REP release V3.3.2 on August 29th 2016

#include "../include/vrep_plugin_skeleton/ros_server_skeleton.h"
#include "../include/v_repLib.h"

ros::NodeHandle* ROS_server::node = NULL;

// Services:
ros::ServiceServer ROS_server::displayText_server;

// Publishers:
ros::Publisher ROS_server::objectCount_publisher;

// Subscribers:
ros::Subscriber ROS_server::addStatusBarMessage_subscriber;

bool ROS_server::initialize()
{
    int argc = 0;
    char** argv = NULL;
    ros::init(argc,argv,"vrep");

    if(!ros::master::check())
        return(false);

    node=new ros::NodeHandle("~");

    // Enable the services:
    displayText_server = node->advertiseService("displayText",ROS_server:: ▼43
    43▲ displayText_service);

    // Enable the publishers:
    objectCount_publisher=node->advertise<std_msgs::Int32>("objectCount",1);

    // Enable the subscribers:
    addStatusBarMessage_subscriber=node->subscribe("addStatusbarMessage",1,& ▼49
    49▲ ROS_server::addStatusbarMessage_callback);

```

```
    return(true);
}

void ROS_server::shutDown()
{
    // Disable the subscribers:
    addStatusBarMessage_subscriber.shutdown();

    // Disable the publishers:
    objectCount_publisher.shutdown();

    // Disable the services:
    displayText_server.shutdown();

    // Shut down:
    ros::shutdown();
}

void ROS_server::instancePass()
{ // When simulation is not running, we "spinOnce" here:
    int simState=simGetSimulationState();
    if ((simState&sim_simulation_advancing)==0)
        spinOnce();
}

void ROS_server::mainScriptAboutToBeCalled()
{ // When simulation is running, we "spinOnce" here:
    spinOnce();
}

void ROS_server::simulationAboutToStart()
{
}

void ROS_server::simulationEnded()
{
}

void ROS_server::spinOnce()
{
    // Disable error reporting (it is enabled in the service processing part, but ▼91
    91▲ we don't want error reporting for publishers/subscribers)
    int errorModeSaved;
    simGetIntegerParameter(sim_intparam_error_report_mode,&errorModeSaved);
    simSetIntegerParameter(sim_intparam_error_report_mode, ▼94
    94▲ sim_api_errormessage_ignore);

    //Handle all streaming (publishers)
    streamAllData();

    //Process all requested services and topic subscriptions
    ros::spinOnce();
```

```
    // Restore previous error report mode:
    simSetIntegerParameter(sim_intparam_error_report_mode,errorModeSaved);
}

// Services:
bool ROS_server::displayText_service(vrep_skeleton_msg_and_srv::displayText::Request &req,vrep_skeleton_msg_and_srv::displayText::Response &res)
{
    res.dialogHandle=simDisplayDialog("Message from a ROS node",req.textToDisplay.c_str(),sim_dlgstyle_message,NULL,NULL,NULL,NULL);
    return true;
}

// Publishers:
void ROS_server::streamAllData()
{
    // Take care of publishers here (i.e. have them publish their data):
    std_msgs::Int32 objCnt;
    int index=0;
    int h=0;
    while (h>=0)
        h=simGetObjects(index++,sim_handle_all);
    objCnt.data=index-1;
    objectCount_publisher.publish(objCnt);
}

// Subscribers:
void ROS_server::addStatusbarMessage_callback(const std_msgs::String::ConstPtr& msg)
{
    simAddStatusbarMessage(msg->data.c_str());
}
```

Script 98: Text file: _ros_node.txt

```
Run a publisher for topic1
$ rostopic pub -r 10 /topic1 std_msgs/String "subscriber test"

Run the test node with a sampling paramter
$ ~/.ros/workspace/devel/.private/coslam_vrep/lib/coslam_vrep/test _sampling:=12

Run a subscriber for topic2
$ rostopic echo /topic2
```

Script 99: Python script file: setup.py

```
#!/usr/bin/env python
from distutils.core import setup
from catkin_pkg.python_setup import generate_distutils_setup

# Fetch values from package.xml
setup_args = generate_distutils_setup(
# version='0.5', # Imported from generate_distutils_setup
  scripts=['scripts/python/stereo_vision.py',
    'scripts/python/stereo_image_preprocessor.py'],
  packages=['coslam_vrep'],
  package_dir={'': 'source'},
  )

setup(**setup_args)
```

Script 100: Text file: _nc_calibration-camera-kannala-brandt-model.txt

```
Calibrate drone camera with kannala-brandtt model

Store from P matrix the fx fy cx cy
Make a cfg file with those values
Install rectified image node in drone
Rectify the image
Record rectified image full size
Use LSD SLAM with rectified image and cfg file crop to 640 480
```

Script 101: Text file: vrep-installation.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Report
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP -- Installation on Arch Linux

# General information
In order to install V-REP on Arch Linux use the AUR package or the modified ▼9
    9▲ PKGBUILD based on the vrep_system_install.sh.

# Important notes
* It might be necessary to force python2 to run certain parts of the build code in ▼12
    12▲ order to provide compatibility
```

Script 102: Text file: _guide.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Guide
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# General title -- subtitle

# General information
General information

# Important notes
* Point A
* Point B

# Syntax elements
<el1> - filename
<el2> - file tag

# Examples
Example running cmd:
cmd <el1> -c <el2>

Example 1:
cmd filename -c tag
```

Script 103: General file: vrep-ros-plugins.patch

```
--- src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼1
    1▲ /CMakeLists.txt  2016-09-16 08:57:57.000000000 +0200
+++ src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼2
    2▲ /CMakeLists.txt  2017-04-27 21:58:08.049993713 +0200
@@ -26,7 +26,7 @@

 add_custom_command(OUTPUT
     ${CMAKE_BINARY_DIR}/generated/adv.cpp ${CMAKE_BINARY_DIR}/generated/pub.cpp ${ ▼6
    6▲ CMAKE_BINARY_DIR}/generated/ros_msg_io.cpp ${CMAKE_BINARY_DIR}/generated/ ▼6
    6▲ ros_msg_io.h ${CMAKE_BINARY_DIR}/generated/ros_srv_io.cpp ${CMAKE_BINARY_DIR ▼6
    6▲ }/generated/ros_srv_io.h ${CMAKE_BINARY_DIR}/generated/srvcall.cpp ${ ▼6
    6▲ CMAKE_BINARY_DIR}/generated/srvcli.cpp ${CMAKE_BINARY_DIR}/generated/srvsrv. ▼6
    6▲ cpp ${CMAKE_BINARY_DIR}/generated/sub.cpp
-    COMMAND python ${CMAKE_SOURCE_DIR}/tools/generate_ros_stuff.py ${ ▼7
    7▲ CMAKE_SOURCE_DIR}/meta/messages.txt ${CMAKE_SOURCE_DIR}/meta/services.txt ${ ▼7
    7▲ CMAKE_BINARY_DIR}/generated/
+    COMMAND python2 ${CMAKE_SOURCE_DIR}/tools/generate_ros_stuff.py ${ ▼8
    8▲ CMAKE_SOURCE_DIR}/meta/messages.txt ${CMAKE_SOURCE_DIR}/meta/services.txt ${ ▼8
    8▲ CMAKE_BINARY_DIR}/generated/
     DEPENDS ${CMAKE_SOURCE_DIR}/meta/messages.txt ${CMAKE_SOURCE_DIR}/meta/ ▼9
    9▲ services.txt ${CMAKE_SOURCE_DIR}/tools/generate_ros_stuff.py)

 if ( (EXISTS ${CMAKE_SOURCE_DIR}/include/stubs.h) AND (EXISTS ${CMAKE_SOURCE_DIR}/ ▼11
    11▲ src/stubs.cpp) )
```

Script 104: Text file: _guidelines.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Guidelines
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# General title -- subtitle

# General information
General information

# Important notes
* Point A
* Point B

# Syntax elements
<el1> - filename
<el2> - file tag

# Examples
Example running cmd:
cmd <el1> -c <el2>

Example 1:
cmd filename -c tag
```

Script 105: Text file: lua-scripts.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Convention
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# Lua scripts -- V-REP scripts

# General information
Lua scripts cannot be loaded from the outside and are stored in $VREP_ROOT with the ▼9
    9▲  filename structure embScript_[0-9]*[0-9] once opened.

# Important notes
* Starts with a header
* Sections divided by two newlines
* Subsections divided by one newline
* Tabulator indents according to hierarchy
* Spaces between operators and objects for increased readability

# Syntax style
* smallname -- local and temporary variables
* normalName -- normal variables
* CapitalName -- global and persistent variables
* CRUCIALNAME -- crucial variables
* helperfunction -- additional functions
* relevantFuction -- normal functions
* ImportantFuction -- important functions
* CRUCIALFUNCTION -- crucial functions

# Examples
Example running cmd:
cmd <el1> -c <el2>

Example 1:
cmd filename -c tag
```

Script 106: General file: vrep-custom-msgs.patch

```
--- src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼1
    1▲ /meta/messages.txt 2016-09-16 08:57:57.000000000 +0200
+++ src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼2
    2▲ /meta/messages.txt 2017-04-27 21:59:57.189993621 +0200
@@ -139,6 +139,8 @@
 trajectory_msgs/JointTrajectoryPoint
 trajectory_msgs/MultiDOFJointTrajectory
 trajectory_msgs/MultiDOFJointTrajectoryPoint
+velodyne_msgs/VelodyneScan
+velodyne_msgs/VelodynePacket
 visualization_msgs/ImageMarker
 visualization_msgs/InteractiveMarker
 visualization_msgs/InteractiveMarkerControl
--- src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼12
    12▲ /package.xml 2016-09-16 08:57:57.000000000 +0200
+++ src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼13
    13▲ /package.xml 2017-04-27 21:58:39.029993687 +0200
@@ -26,6 +26,7 @@
   <build_depend>tf2_msgs</build_depend>
   <build_depend>tf2_sensor_msgs</build_depend>
   <build_depend>trajectory_msgs</build_depend>
+  <build_depend>velodyne_msgs</build_depend>
   <build_depend>visualization_msgs</build_depend>
   <run_depend>roscpp</run_depend>
   <run_depend>libopencv-dev</run_depend>
@@ -50,6 +51,7 @@
   <run_depend>tf2_msgs</run_depend>
   <run_depend>tf2_sensor_msgs</run_depend>
   <run_depend>trajectory_msgs</run_depend>
+  <run_depend>velodyne_msgs</run_depend>
   <run_depend>visualization_msgs</run_depend>
 </package>

--- src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼30
    30▲ /CMakeLists.txt 2016-09-16 08:57:57.000000000 +0200
+++ src/V-REP_PRO_EDU_V3_3_2_64_Linux/programming/ros_packages/v_repExtRosInterface ▼31
    31▲ /CMakeLists.txt 2017-04-27 21:58:08.049993713 +0200
@@ -10,8 +10,8 @@
 endif()

 catkin_package(
-  CATKIN_DEPENDS roscpp rosconsole cv_bridge image_transport tf brics_actuator ▼36
    36▲ roslib actionlib_msgs control_msgs diagnostic_msgs geometry_msgs map_msgs ▼36
    36▲ nav_msgs pcl_msgs sensor_msgs shape_msgs std_msgs tf2_geometry_msgs ▼36
    36▲ tf2_msgs tf2_sensor_msgs trajectory_msgs visualization_msgs
-  DEPENDS roscpp rosconsole cv_bridge image_transport tf brics_actuator roslib ▼37
    37▲ actionlib_msgs control_msgs diagnostic_msgs geometry_msgs map_msgs nav_msgs ▼37
    37▲ pcl_msgs sensor_msgs shape_msgs std_msgs tf2_geometry_msgs tf2_msgs ▼37
    37▲ tf2_sensor_msgs trajectory_msgs visualization_msgs)
+  CATKIN_DEPENDS roscpp rosconsole cv_bridge image_transport tf brics_actuator ▼38
    38▲ roslib actionlib_msgs control_msgs diagnostic_msgs geometry_msgs map_msgs ▼38
    38▲ nav_msgs pcl_msgs sensor_msgs shape_msgs std_msgs tf2_geometry_msgs ▼38
```

```
    38▲ tf2_msgs tf2_sensor_msgs trajectory_msgs velodyne_msgs visualization_msgs
+   DEPENDS roscpp rosconsole cv_bridge image_transport tf brics_actuator roslib  ▼39
    39▲ actionlib_msgs control_msgs diagnostic_msgs geometry_msgs map_msgs nav_msgs ▼39
    39▲  pcl_msgs sensor_msgs shape_msgs std_msgs tf2_geometry_msgs tf2_msgs  ▼39
    39▲ tf2_sensor_msgs trajectory_msgs velodyne_msgs visualization_msgs)
40
  file(MAKE_DIRECTORY ${CMAKE_BINARY_DIR}/generated)
```

Script 107: Text file: vrep-custom-messages.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Guide
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# V-REP -- Custom messages

# General information
In order to run V-REP with custom messages the v_repExtRosInterface plugin needs to ▼9
    9▲ be modified and recompiled within the V-REP environment.

# Important notes
* Add custom messages to package.xml in build and runtime dependencies respectively
* Add custom messages types to meta/messages.txt
* Add custom messages to CMakeLists.txt in CATKIN_DEPENDS and DEPENDS
* Make sure the compiled plugin is in V-REP environment, otherwise message window ▼15
    15▲ will continue reporting unknown message type
* More information can be found in the README.txt inside v_repExtRosInterface ▼16
    16▲ directory inside V-REP folder, typically located in $VREP_ROOT/programming/ ▼16
    16▲ ros_packages/catkin/src/

# Resources
* https://github.com/fferri/v_repExtRosInterface
```

Script 108: Text file: ros-nodes.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Convention
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# ROS -- Nodes

# General information

# Important notes
* In order to run V-REP with custom messages the v_repExtRosInterface plugin needs ▼11
    11▲ to be modified and recompiled within the V-REP environment.

# Definitions
* Headers
NN_XX5XX -- Node name

* Topics
TP_XX5XX_Y*Y -- Topic, node name, message type or other identifier
```

Script 109: Text file: Compilation.txt

```
Various code base handling advice.
* Since the submitted work was on read-only systems, the permissions might not be ▼2
   2▲ suitable for compilation. Use
  $ find . -type d -exec chmod 775 {} +
  to change directory permissions.
*
```

Script 110: Text file: Patching.txt

```
LSD-SLAM viewer
* Change file permission to include the exectuable flag for *.cfg files that are ▼2
    2▲ called during configuration stage
NC* Change the hashbang line to #!/usr/bin/env python2 to force for Python 2 ▼3
    3▲ version
* Correct the include directory setting in package.xml to include a folder that ▼4
    4▲ holds necessary headers keyframeGraphMsg.h, keyframeMsg.h and ▼4
    4▲ LSDSLAMViewerParamsConfig.h
* Change the declaration of position variables x, y and z type from float to double ▼5
    5▲  by from "float x, y, z" to "qreal x, y, z" in all header and source files
#* Change the declaration of position variables x, y and z type from float to ▼6
    6▲ double by from "float x, y, z" to "double x, y, z" in all header and source ▼6
    6▲ files
#$ for F in *.cpp *.h; do sed -i 's/float\ x,y,z\;/double\ x,y,z\;/g' "$F"; done
$ for F in *.cpp *.h; do sed -i 's/float\ x,y,z\;/qreal\ x,y,z\;/g' "$F"; done
$ for F in *.cpp *.h; do sed -i 's/%f\ %f\ %f\ %d/%lf\ %lf\ %lf\ %d/g' "$F"; done
* Change #include order to first position for the non free opencv header opencv2/ ▼10
    10▲ opencv.hpp

<Warnings>
* Change to correct printf formats (for example for double from %f to %lf)
```

Script 111: Text file: Instructions.txt

```
1. Follow Arch Wiki and install base and base-devel packages
2. Add French repository to pacman.conf
3. Run $ yaourt -S --noconfirm --needed ros-indigo-desktop-full
4. Move # mv /usr/bin/qmake{,.org}; and copy # cp /usr/bin/qmake-qt4 /usr/bin/qmake
5. Rerun command from #3
6. Run $ sudo rosdep init
7. Run $ rosdep update
8. Update profile with expanding ROS functions (ros-indigo, ros-jade)
9. Use either arch-ros-stacks and/or AUR (Arch User Repository) to get and compile ▼9
   9▲ custom packages, where in certain cases indigo versions will work for jade
```

Script 112: Text file: RViz.txt

```
1 Follow @ref Instructions.txt
  Install Ogre 1.9
  Modify PKGBUILD to include the custom Ogre 1.9 path
  Compile RViz
```

Script 113: Bash compatible shell script file: ubuntu-opencv-custom-repository.sh

```
#!/bin/bash
# In order to use the place recognition module, non free elements of the OpenCV package need to be installed.
# The scripts from the GitHub repository:
# https://github.com/jayrambhia/Install-OpenCV/tree/master/Ubuntu
# were not successful in compilation for both tested versions 2.4.9 and 2.4.10

# The repository suggested in Stackoverflow answer:
# http://stackoverflow.com/questions/27481849/include-nonfree-opencv-2-4-10-on-ubuntu#27493595
# did however provide the necessary libraries to compile the code base from Nived's work after a reboot.
sudo add-apt-repository --yes ppa:xqms/opencv-nonfree
sudo apt-get update
sudo apt-get install libopencv-nonfree-dev
```

Script 114: Text file: Dependencies.txt

```
1 * LSD-SLAM and CoSLAM are dependent on OpenCV non-free modules and are not present ▼1
    1▲ in official repositories, therefore an installation either from a custom ▼1
    1▲ repository or manually is required. For manual installation scripts are ▼1
    1▲ available from https://github.com/jayrambhia/Install-OpenCV, however they are ▼1
    1▲ not sufficient and of poor quality.
  * For Ubuntu 14.04, ROS full desktop installtion is dependant on QT4 framework. ▼2
    2▲ However QT5 is present in repositories and since OpenCV with non-free modules ▼2
    2▲ for SIFT and SURF feature descriptors has to be build, cmake automatically ▼2
    2▲ picks QT5, therefore creating a situation where rviz build against QT4 tries ▼2
    2▲ to use OpenCV modules linked against QT5. To properly compile OpenCV set ▼2
    2▲ cmake argument with -D WITH_QT=4 to force version 4 for QT.
  * Any changes to depenencies might trigger necessary recompilation for end ▼3
    3▲ applications like V-REP and it's ROS plugin
```

Script 115: Text file: datasets-naming-convention.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Convention
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# ROS bag datasets naming

# Syntax elements
<robot> - vehicle, robot, platform
<location> - place, area
<session> - session run ID
<run> - variations of session
<date> - preferably in "%Y%m%d" format for easier sorting

# Additional syntax
|delimiters| - whitespace discouraged, because of escape characters in shell, ▼16
    16▲ preferably single hyphen for linking elements and double hyphen for linking ▼16
    16▲  element sub-information

# Examples
Example command naming convention:
rosbag record <location>-<date>-<robot>-<session>-<run>.bag

Example 1:
rosbag play ecn-20170214-turtlebot-loop-forward.bag

Example 2:
rosbag play ecn--parkinglot-20170214-turtlebot--a-loop--01-forward--a.bag

Format of example 2 can be translated into format from example 1 using sed:
$ echo rosbag play ecn--parkinglot-20170214-turtlebot--a-loop--01-forward--a.bag |  ▼29
    29▲ sed 's/\(--[^-]*-\)/-/g;s/\(--[^-]*\.\)/./'
rosbag play ecn-20170214-turtlebot-loop-forward.bag
```

Script 116: Text file: creating-squashfs-archives-convention.txt

```
### École centrale de Nantes
### Università degli studi di Genova
### EMARO+ - 2016--2017
## Master thesis - Convention
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# Compressing ROS bag datasets into SquashFS

# General information
SquashFS is a read-only filesystem that enables compression of not only files, but ▼10
    10▲ also permissions, inodes and attributes. It provides extremely efficient ▼10
    10▲ compression with large number of small size files relative to filesystem ▼10
    10▲ block. Furthermore the different compression algorithms allow for extended ▼10
    10▲ accessibility.

# Important notes
*It is possible to append files to an already created archive and to overwrite ▼13
    13▲ existing archive with the -noappend option.
*In order to be able to mount the archive, compression algorithms gzip, which is ▼14
    14▲ the default one, lz4 or xz need to be used.

# Syntax elements
<archive> - archive file holding the SquashFS according to datasets-naming- ▼17
    17▲ convention.txt
<comp> - compression algorithm
<files> - files to be included into the filesystem archive
<files-extract> - files to be extracted from the filesystem archive
<mnt> - mount destination
<username> - User name stored in ${USER}

# Examples
Example compression with default algorithm:
mksquashfs <files> <archive>

Example compression with specified algorithm
mksquashfs <files> <archive> -comp <comp>

Example 1:
mksquashfs ecn-icars-A-2017-02-09-*.bag ecn-icars-A-2017-02-09.sfs

Example mounting of created filesystem:
# mount -t squashfs <archive> <mnt>

Example 1:
# mount -t squashfs ecn-icars-A-2017-02-09.sfs /mnt

Example mounting with udevil to a /media/<username>/<archive>/ directory
$ udevil mount <archive>

Example 1:
$ udevil mount ecn-icars-A-2017-02-09.sfs

```

```
Example unmounting with from /media/<username>/<archive>/ directory
# umount /media/<username>/<archive>

Example 1:
$ udevil umount /media/ecn-icars-A-2017-02-09.sfs/

Example unmounting with udevil of <archive> from /media/<username>/<archive>/  ▼52
    52▲ directory
$ udevil umount <archive>

Example 1:
$ udevil umount /media/ecn-icars-A-2017-02-09.sfs/

Example 2:
$ udevil umount ecn-icars-A-2017-02-09.sfs

Example listing of archive contents:
$ unsquashfs -l <archive>

Example 1:
$ unsquashfs -l ecn-icars-A-2017-02-09.sfs

Example extraction of all archive contents:
$ unsquashfs <archive>

Example 1:
$ unsquashfs ecn-icars-A-2017-02-09.sfs

Example extraction of selected archive contents:
$ unsquashfs <archive> <files-extract>

Example 1:
$ unsquashfs ecn-icars-A-2017-02-09.sfs ecn-icars-A-2017-02-09-14-37-36.bag

# Usage help for mksquashfs
SYNTAX: mksquashfs source1 source2 ...  dest [options] [-e list of exclude
dirs/files]

Filesystem build options:
-comp <comp>     select <comp> compression
      Compressors available:
        gzip (default)
        lzma
        lzo
        lz4
        xz
-b <block_size>   set data block to <block_size>.  Default 128 Kbytes
      Optionally a suffix of K or M can be given to specify
      Kbytes or Mbytes respectively
-no-exports   don't make the filesystem exportable via NFS
-no-sparse    don't detect sparse files
-no-xattrs    don't store extended attributes
-xattrs     store extended attributes (default)
```

```
-noI       do not compress inode table
-noD       do not compress data blocks
-noF       do not compress fragment blocks
-noX       do not compress extended attributes
-no-fragments    do not use fragments
-always-use-fragments use fragment blocks for files larger than block size
-no-duplicates     do not perform duplicate checking
-all-root    make all files owned by root
-force-uid uid     set all file uids to uid
-force-gid gid     set all file gids to gid
-nopad       do not pad filesystem to a multiple of 4K
-keep-as-directory  if one source directory is specified, create a root
      directory containing that directory, rather than the
      contents of the directory

Filesystem filter options:
-p <pseudo-definition>  Add pseudo file definition
-pf <pseudo-file> Add list of pseudo file definitions
-sort <sort_file> sort files according to priorities in <sort_file>.  One
      file or dir with priority per line.  Priority -32768 to
      32767, default priority 0
-ef <exclude_file>  list of exclude dirs/files.  One per line
-wildcards     Allow extended shell wildcards (globbing) to be used in
      exclude dirs/files
-regex       Allow POSIX regular expressions to be used in exclude
      dirs/files

Filesystem append options:
-noappend    do not append to existing filesystem
-root-becomes <name>  when appending source files/directories, make the
      original root become a subdirectory in the new root
      called <name>, rather than adding the new source items
      to the original root

Mksquashfs runtime options:
-version     print version, licence and copyright message
-exit-on-error     treat normally ignored errors as fatal
-recover <name>   recover filesystem data using recovery file <name>
-no-recovery     don't generate a recovery file
-info      print files written to filesystem
-no-progress     don't display the progress bar
-progress    display progress bar when using the -info option
-processors <number>  Use <number> processors.  By default will use number of
      processors available
-mem <size>    Use <size> physical memory.  Currently set to 4002M
      Optionally a suffix of K, M or G can be given to specify
      Kbytes, Mbytes or Gbytes respectively

Miscellaneous options:
-root-owned    alternative name for -all-root
-noInodeCompression alternative name for -noI
-noDataCompression  alternative name for -noD
-noFragmentCompression  alternative name for -noF
```

```
-noXattrCompression alternative name for -noX

-Xhelp      print compressor options for selected compressor

Compressors available and compressor specific options:
  gzip (default)
    -Xcompression-level <compression-level>
    <compression-level> should be 1 .. 9 (default 9)
    -Xwindow-size <window-size>
    <window-size> should be 8 .. 15 (default 15)
    -Xstrategy strategy1,strategy2,...,strategyN
    Compress using strategy1,strategy2,...,strategyN in turn
    and choose the best compression.
    Available strategies: default, filtered, huffman_only,
    run_length_encoded and fixed
  lzma (no options)
  lzo
    -Xalgorithm <algorithm>
    Where <algorithm> is one of:
      lzo1x_1
      lzo1x_1_11
      lzo1x_1_12
      lzo1x_1_15
      lzo1x_999 (default)
    -Xcompression-level <compression-level>
    <compression-level> should be 1 .. 9 (default 8)
    Only applies to lzo1x_999 algorithm
  lz4
    -Xhc
    Compress using LZ4 High Compression
  xz
    -Xbcj filter1,filter2,...,filterN
    Compress using filter1,filter2,...,filterN in turn
    (in addition to no filter), and choose the best compression.
    Available filters: x86, arm, armthumb, powerpc, sparc, ia64
    -Xdict-size <dict-size>
    Use <dict-size> as the XZ dictionary size.  The dictionary size
    can be specified as a percentage of the block size, or as an
    absolute value.  The dictionary size must be less than or equal
    to the block size and 8192 bytes or larger.  It must also be
    storable in the xz header as either 2^n or as 2^n+2^(n+1).
    Example dict-sizes are 75%, 50%, 37.5%, 25%, or 32K, 16K, 8K
    etc.
```

Script 117: Text file: naming-graphics-photography-convention.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Convention
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# Naming graphics -- photography

# General information

# Important notes

# Syntax elements
<filename> - filename
<tag> - file tag
<type> - type of file
<ext> - extension of the file, use either UPPERCASE or lowercase only -- DO NOT MIX

# Examples
Example compression with default algorithm
```

Script 118: Text file: _convention.txt

```
### École centrale de Nantes
### EMARO+ - 2016--2017
## Master thesis - Convention
## UGV and UAV collaboration in an autonomous infrastructure scenario
## Ernest Skrzypczyk
# General title -- subtitle convention

# General information
General information

# Important notes
* Point A
* Point B

# Syntax elements
<el1> - filename
<el2> - file tag

# Examples
Example running cmd:
cmd <el1> -c <el2>

Example 1:
cmd filename -c tag
```

Script 119: Bash compatible shell script file: vrep_system_install_3_3_2.sh

```
#!/bin/bash
## @file
## @author Olivier Kermorgant
## @author Ernest Skrzypczyk
## @brief Installs V-REP in /opt and configure the ros plugin
## @date 10.06.2017
## @licence The BSD License

## Requirements
if [[ "$(lsb_release -sd)" != "Ubuntu"* ]]; then echo "Run on Ubuntu with bash \$ ${0}"; exit 1; fi
if [[ "$SHELL" != *"/bin/bash" ]]; then echo "Run with bash \$ ${0}"; exit 1; fi

## Parameters
## Use archive version of ROS plugin, set to 0 for github version
LOCAL=0;
CDIR="$PWD";



_vrep_root="/opt/vrep"
vrep_version="3_3_2"
#_vrep_root="/opt/vrep_${_vrep_root}"
## Default ROS version
#ros_distro="indigo"
## @warning Despite the developed code being compatible with both indigo and jade versions of ROS, indigo is preferred, since indigo holds velodyne_msgs package and custom messages can be easily build.
ros_distro="jade"
version="$(lsb_release -sc)"

if [ "$version" = "xenial" ]; then ros_distro="kinetic"; fi

echo "[Installing dependencies]"
sudo apt-get install -y python-catkin-tools

echo "[Downloading V-REP]"
## Use stable rev for 3.3.2 from
## @see http://coppeliarobotics.com/files/V-REP_PRO_EDU_V3_3_2_Linux.tar.gz
if [ ! -f "V-REP_PRO_EDU_V${vrep_version}_64_Linux.tar.gz" ]; then
    wget http://coppeliarobotics.com/files/V-REP_PRO_EDU_V${vrep_version}_64_Linux.tar.gz
#    wget "http://coppeliarobotics.com/files/tmp/V-REP_PRO_EDU_V${vrep_version}_rev9_Linux.tar.gz"
fi

echo "[Extracting V-REP]"
if [ ! -d "V-REP_PRO_EDU_V${vrep_version}_64_Linux" ]; then
## Extract beta rev9 version
# tar -xzf "V-REP_PRO_EDU_V${vrep_version}_rev9_Linux.tar.gz"
  tar -xzf "V-REP_PRO_EDU_V${vrep_version}_64_Linux.tar.gz"
fi
```

```
## Extract additional archives if found (here just 7z)
if [ -f "$(ls -1 "${0#.sh}"*.7z | head -1 &>/dev/null)" ]; then
  for F in "${0#.sh}"*.7z; do echo "[Processing $F]"; 7z t "${F}" &>/dev/null && 7z x "${F}"; done
fi

## Apply initial patches
if [ -f "$(ls -1 "${0#.sh}"*.patch | head -1 &>/dev/null)" ]; then
  for F in "${0#.sh}"*.patch; do echo "[Processing $F]"; patch < "${F}"; done
fi

CDIR="$PWD"
sudo rm -rf "${_vrep_root}/"
sudo cp -R "V-REP_PRO_EDU_V${vrep_version}_64_Linux" "${_vrep_root}/"
sudo chmod -R a+rwx "${_vrep_root}/"
export VREP_ROOT="${_vrep_root}/"

echo "[Configuring ROS plugin and interface]"
cd "${_vrep_root}/programming/ros_packages/"
mkdir -p catkin/src
chmod -R a+rx catkin
mv vrep_common catkin/src
mv vrep_plugin catkin/src

cd catkin/src/vrep_plugin
## Update CMakeLists for OpenCV bug
sed -i 's/include_directories/find_package(OpenCV)\ninclude_directories/' CMakeLists.txt
sed -i 's/${catkin_INCLUDE_DIRS}/${catkin_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS}/' CMakeLists.txt

## Init ROS interface
if [ $LOCAL -gt 0 ]; then
  cd "${_vrep_root}/programming/ros_packages/"
  mv v_repExtRosInterface catkin/src
else
  cd "${_vrep_root}/programming/ros_packages/catkin/src/"
  if [ -d v_repExtRosInterface ]; then rm -r v_repExtRosInterface; fi
  git clone https://github.com/fferri/v_repExtRosInterface
  export PYTHONPATH="$PWD:$PYTHONPATH"
  cd v_repExtRosInterface
  export PYTHONPATH="$PWD:$PYTHONPATH"
  #sed -i 's/COMMAND python /COMMAND python2 /g' CMakeLists.txt
# git checkout 6042debe33d79fc8e9de43c19554762382eb6115
  ## Commit from 2017-05-20 using v_repStubsGen as a submodule
  git checkout 089a3fa2a16c218bd1cf9c02a37421924aa17ad7

  mkdir -p external
  cd external
  export PYTHONPATH="$PWD:$PYTHONPATH"
  git clone https://github.com/fferri/v_repStubsGen.git
  cd v_repStubsGen
  export PYTHONPATH="$PWD:$PYTHONPATH"
```

```
  ## Commit from 2017-04-28 should work with 3.3.2
# git checkout 91cd5a71923dc6653103f292e3390fcb75423b20
  ## Commit from 2017-05-20 could work with 3.3.2
  git checkout 594dfb53a080e315908fb55de68f7b78bd135b3b

  mkdir -p external
  cd external
  export PYTHONPATH="$PWD:$PYTHONPATH"
  if [ -d pycpp ]; then rm -r pycpp; fi
  git clone https://github.com/fferri/PyCPP.git pycpp
  cd pycpp
  export PYTHONPATH="$PWD:$PYTHONPATH"
  ## Commit from 2017-06-05 could work with 3.3.2
  git checkout eb5879ada4502ac9117e02171002e633848751fc
fi

## Patch/Extract for custom messages
echo "[Patching for custom messages]"
cd "${_vrep_root}/programming/ros_packages/catkin/src/v_repExtRosInterface"
sed -i 's/include_directories/find_package(OpenCV)\ninclude_directories/'  ▼117
    117▲ CMakeLists.txt
sed -i 's/${catkin_INCLUDE_DIRS}/${catkin_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS}/'  ▼118
    118▲ CMakeLists.txt
sed -i 's/visualization_msgs/velodyne_msgs visualization_msgs/' CMakeLists.txt
sed -i 's#visualization_msgs</build_depend#velodyne_msgs</build_depend>\n  < ▼120
    120▲ build_depend>visualization_msgs</build_depend#' package.xml
sed -i 's#visualization_msgs</run_depend#velodyne_msgs</run_depend>\n  <run_depend ▼121
    121▲ >visualization_msgs</run_depend#' package.xml
sed -i 's#visualization_msgs/ImageMarker#velodyne_msgs/VelodynePacket\ ▼122
    122▲ nvelodyne_msgs/VelodyneScan\nvisualization_msgs/ImageMarker#' meta/ ▼122
    122▲ messages.txt

echo "[Compiling ROS plugins]"
echo "PYTHONPATH=$PYTHONPATH"
echo "ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH"
cd "${_vrep_root}/programming/ros_packages/catkin/"
## Extended configuration is crucial for proper custom messages build
catkin config --extend "/opt/ros/${ros_distro}" -i "/opt/ros/${ros_distro}" -- ▼129
    129▲ install
catkin clean -b --yes
sudo -E bash -c 'VREP_ROOT=/opt/vrep && catkin build --pre-clean' || { echo "[ ▼131
    131▲ Compilation error]"; exit 1; };

echo "[Linking ROS plugins]"
# copy plugin to vrep folder
if [ -f "${_vrep_root}/libv_repExtRos.so" ]; then sudo rm "${_vrep_root}/ ▼135
    135▲ libv_repExtRos.so"; fi
sudo ln -sv "${_vrep_root}/programming/ros_packages/catkin/devel/.private/ ▼136
    136▲ vrep_plugin/lib/libv_repExtRos.so" "${_vrep_root}/"

# copy extension to vrep folder
if [ -f "${_vrep_root}/libv_repExtRosInterface.so" ]; then sudo rm "${_vrep_root}/ ▼139
    139▲ libv_repExtRosInterface.so"; fi
```

```
sudo ln -sv "${_vrep_root}/programming/ros_packages/catkin/devel/.private/ ▼140
    140▲ vrep_ros_interface/lib/libv_repExtRosInterface.so" "${_vrep_root}/"

sudo chmod -R a+rx "${_vrep_root}/"

echo "[Creating V-REP launch file and shortcut]"
launch_file=/opt/ros/${ros_distro}/share/vrep_plugin/vrep.launch
if [ -f $launch_file ]; then sudo rm $launch_file; fi
sudo touch $launch_file
sudo chmod a+rw $launch_file
sudo cat>$launch_file <<EOL
<?xml version="1.0"?>
<launch>
    <!-- Launch V-REP and roscore -->
    <arg name="scene" default=""/>
    <node name="vrep" pkg="vrep_plugin" type="vrep.py" respawn="false" output=" ▼154
    154▲ screen" args="\$(arg scene)"/>

    <!-- Launch check vitals to stop simulation if a topic is not published  ▼156
    156▲ anymore -->
    <arg name="topic" default="this_is_not_a_topic"/>
    <arg name="delay" default="3"/>
    <node name="check_vitals" pkg="vrep_plugin" type="check_vitals.py" output=" ▼159
    159▲ screen" ns="vrep" args="\$(arg topic) \$(arg delay)"/>
</launch>
EOL

# shortcut for V-REP launch script
node_file=/opt/ros/${ros_distro}/share/vrep_plugin/vrep.py
if [ -f $node_file ]; then sudo rm $node_file; fi
sudo touch $node_file
sudo chmod a+rw $node_file
sudo cat>$node_file <<EOL
#!/usr/bin/env python
from subprocess import call
from sys import argv
from os import chdir
from os.path import abspath
arg = ''
if len(argv) > 1:
    arg = abspath(argv[1])
chdir('/opt/vrep')
call(['bash','vrep.sh',arg])
EOL
sudo chmod a+x $node_file


# check vitals
node_file=/opt/ros/${ros_distro}/share/vrep_plugin/check_vitals.py
if [ -f $node_file ]; then sudo rm $node_file; fi
sudo touch $node_file
sudo chmod a+rw $node_file
sudo cat>$node_file <<EOL
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
import rospy
from vrep_common.srv import simRosStopSimulation, simRosStartSimulation
import sys
from commands import getoutput

if __name__ == "__main__":

    # no topic given, I am useless
    if sys.argv[1] == 'this_is_not_a_topic':
        rospy.loginfo('check_vitals: No topic given, exit')
        sys.exit(0)

    cmd_line = 'rostopic info ' + sys.argv[1]
    msg_stop = 'check_vitals: Nobody publishing on ' + sys.argv[1] + ', stopping simulation'
    msg_run = 'check_vitals: Got publisher on ' + sys.argv[1] + ', starting simulation'

    delay = int(sys.argv[2])

    rospy.init_node('check_vitals')

    # prepare to stop the simulation every <delay> sec
    stopSim = rospy.ServiceProxy('/vrep/simRosStopSimulation', simRosStopSimulation)
    startSim = rospy.ServiceProxy('/vrep/simRosStartSimulation', simRosStartSimulation)
    startSim.wait_for_service()

    rate = rospy.Rate(1)

    alive = False
    last_seen = 0

    while not rospy.is_shutdown():

        t = rospy.Time.now().to_sec()

        # get info on this topic
        out = getoutput(cmd_line).splitlines()

        # Look for any publishers, will pass if topic does not exist (simulation not running)
        for line in out:
            if 'Publishers' in line and 'None' not in line:
                last_seen = t

        # check last seen
        if t - last_seen > delay and alive:
            # just died
```

```
            alive = False
            rospy.loginfo(msg_stop)
            stopSim()
        elif not alive and t - last_seen < delay:
            # just popped
            alive = True
            rospy.loginfo(msg_run)
            startSim()
        rate.sleep()
EOL
sudo chmod a+x $node_file

echo "[Archivize installation]"
ARCHIVE="vrep-${vrep_version}-${ros_distro}-${VREP_ROOT////_}.$(date '+%d%m%y').7z ▼250
    250▲ ";
cd "$CDIR";
if [ ! -f "${ARCHIVE}" ]; then 7z a "${ARCHIVE}" "${VREP_ROOT}" &>/dev/null; fi
```

Script 120: Bash compatible shell script file: vrep_system_install_3_4_0.sh

```
#!/bin/bash
## @file
## @author Olivier Kermorgant
## @author Ernest Skrzypczyk
## @brief Installs V-REP in /opt and configure the ros plugin
## @date 10.06.2017
## @licence The BSD License

## Requirements
if [[ "$(lsb_release -sd)" != "Ubuntu"* ]]; then echo "Run on Ubuntu with bash \$ $ ▼10
    10▲ {0}"; exit 1; fi
if [[ "$SHELL" != *"/bin/bash" ]]; then echo "Run with bash \$ ${0}"; exit 1; fi

## Parameters
## Use archive version of ROS plugin, set to 0 for github version
LOCAL=1;
CDIR="$PWD";



_vrep_root="/opt/vrep"
vrep_version="3_4_0"
#_vrep_root="/opt/vrep_${_vrep_root}"
ros_distro="jade" # Default ROS version
version="$(lsb_release -sc)"

if [ "$version" = "xenial" ]; then ros_distro="kinetic"; fi

echo "[Installing dependencies]"
sudo apt-get install -y python-catkin-tools
sudo apt-get install -y python-tempita
sudo apt-get install -y xsltproc
## Saxon
## @todo Not resolved dependencies, maybe CMakeLists.txt needs modification
sudo apt-get install -y libsaxonhe-java docbook-xsl-saxon

echo "[Downloading V-REP]"
## Use beta rev9 for 3.4.0 from
## @see http://coppeliarobotics.com/files/tmp/V-REP_PRO_EDU_V3_4_0_rev9_Linux.tar. ▼37
    37▲ gz
if [ ! -f "V-REP_PRO_EDU_V${vrep_version}_Linux.tar.gz" ]; then
#    wget http://coppeliarobotics.com/files/V-REP_PRO_EDU_V${vrep_version}_Linux. ▼39
    39▲ tar.gz
    wget "http://coppeliarobotics.com/files/tmp/V-REP_PRO_EDU_V${vrep_version} ▼40
    40▲ _rev9_Linux.tar.gz"
fi

echo "[Extracting V-REP]"
if [ ! -d "V-REP_PRO_EDU_V${vrep_version}_Linux" ]; then
## Extract beta rev9 version
  tar -xzf "V-REP_PRO_EDU_V${vrep_version}_rev9_Linux.tar.gz"
# tar -xzf "V-REP_PRO_EDU_V${vrep_version}_Linux.tar.gz"
fi
```

```
## Extract additional archives if found (here just 7z)
if [ -f "$(ls -1 "${0#.sh}"*.7z | head -1 &>/dev/null)" ]; then
  for F in "${0#.sh}"*.7z; do echo "[Processing $F]"; 7z t "${F}" &>/dev/null &&  ▼52
    52▲ yes | 7z x "${F}"; done
fi

## Apply initial patches
if [ -f "$(ls -1 "${0#.sh}"*.patch | head -1 &>/dev/null)" ]; then
  for F in "${0#.sh}"*.patch; do echo "[Processing $F]"; patch < "${F}"; done
fi

CDIR="$PWD"
sudo rm -rf "${_vrep_root}/"
sudo cp -R "V-REP_PRO_EDU_V${vrep_version}_Linux" "${_vrep_root}/"
sudo chmod -R a+rwx "${_vrep_root}/"
export VREP_ROOT="${_vrep_root}/"

echo "[Configuring ROS plugin and interface]"
cd "${_vrep_root}/programming/ros_packages/"
mkdir -p catkin/src
chmod -R a+rx catkin

cd catkin/src/vrep_plugin
## Update CMakeLists for OpenCV bug
#sed -i 's/include_directories/find_package(OpenCV)\ninclude_directories/'  ▼73
    73▲ CMakeLists.txt
#sed -i 's/${catkin_INCLUDE_DIRS}/${catkin_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS}/'  ▼74
    74▲ CMakeLists.txt

## Init ROS interface
if [ $LOCAL -gt 0 ]; then
  cd "${_vrep_root}/programming/ros_packages/"
  mv v_repExtRosInterface catkin/src
else
  cd "${_vrep_root}/programming/ros_packages/catkin/src/"
  if [ -d v_repExtRosInterface ]; then rm -r v_repExtRosInterface; fi
  git clone https://github.com/fferri/v_repExtRosInterface
  git checkout 700a22427d1e121305d6454a13af4682e592a9cb
fi

## Patch/Extract for custom messages
echo "[Patching for custom messages]"
cd "${_vrep_root}/programming/ros_packages/catkin/src/v_repExtRosInterface"
sed -i 's/include_directories/find_package(OpenCV)\ninclude_directories/'  ▼90
    90▲ CMakeLists.txt
sed -i 's/${catkin_INCLUDE_DIRS}/${catkin_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS}/'  ▼91
    91▲ CMakeLists.txt
sed -i 's/visualization_msgs/velodyne_msgs visualization_msgs/' CMakeLists.txt
sed -i 's#visualization_msgs</build_depend#velodyne_msgs</build_depend>\n  < ▼93
    93▲ build_depend>visualization_msgs</build_depend#' package.xml
sed -i 's#visualization_msgs</run_depend#velodyne_msgs</run_depend>\n  <run_depend> ▼94
    94▲ visualization_msgs</run_depend#' package.xml
```

```
sed -i 's#visualization_msgs/ImageMarker#velodyne_msgs/VelodynePacket\ ▼95
    95▲ nvelodyne_msgs/VelodyneScan\nvisualization_msgs/ImageMarker#' meta/messages ▼95
    95▲ .txt

#Extract custom message files, maybe use a patch instead
#yes | 7z e "$MFILE"
#mv messages.txt meta/

#cd ../v_repExtRosInterface
#sed -i 's/COMMAND python /COMMAND python2 /g' CMakeLists.txt
mkdir -p external
cd external
git clone https://github.com/fferri/v_repStubsGen.git
git checkout 9c6a337b15a2d3ee856e2b56ca18156cd0673316
export PYTHONPATH="$PWD:$PYTHONPATH"

echo "[Compiling ROS plugins]"
echo "PYTHONPATH=$PYTHONPATH"
echo "ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH"
cd "${_vrep_root}/programming/ros_packages/catkin/"
## Extended configuration is crucial for proper custom messages build
catkin config --extend "/opt/ros/${ros_distro}" -i "/opt/ros/${ros_distro}" -- ▼114
    114▲ install
catkin clean -b --yes
sudo -E bash -c 'VREP_ROOT=/opt/vrep && catkin build --pre-clean' || { echo "[ ▼116
    116▲ Compilation error]"; exit 1; };

echo "[Linking ROS plugins]"
# copy plugin to vrep folder
if [ -f "${_vrep_root}/libv_repExtRos.so" ]; then sudo rm "${_vrep_root}/ ▼120
    120▲ libv_repExtRos.so"; fi
sudo ln -sv "${_vrep_root}/programming/ros_packages/catkin/devel/.private/ ▼121
    121▲ vrep_plugin/lib/libv_repExtRos.so" "${_vrep_root}/"

# copy extension to vrep folder
if [ -f "${_vrep_root}/libv_repExtRosInterface.so" ]; then sudo rm "${_vrep_root}/ ▼124
    124▲ libv_repExtRosInterface.so"; fi
sudo ln -sv "${_vrep_root}/programming/ros_packages/catkin/devel/.private/ ▼125
    125▲ vrep_ros_interface/lib/libv_repExtRosInterface.so" "${_vrep_root}/"

sudo chmod -R a+rx "${_vrep_root}/"

echo "[Creating V-REP launch file and shortcut]"
launch_file=/opt/ros/${ros_distro}/share/vrep_plugin/vrep.launch
if [ -f $launch_file ]; then sudo rm $launch_file; fi
sudo touch $launch_file
sudo chmod a+rw $launch_file
sudo cat>$launch_file <<EOL
<?xml version="1.0"?>
<launch>
    <!-- Launch V-REP and roscore -->
    <arg name="scene" default=""/>
    <node name="vrep" pkg="vrep_plugin" type="vrep.py" respawn="false" output=" ▼139
```

```
    139▲ screen" args="\$(arg scene)"/>
140
     <!-- Launch check vitals to stop simulation if a topic is not published  ▼141
    141▲ anymore -->
    <arg name="topic" default="this_is_not_a_topic"/>
    <arg name="delay" default="3"/>
    <node name="check_vitals" pkg="vrep_plugin" type="check_vitals.py" output=" ▼144
    144▲ screen" ns="vrep" args="\$(arg topic) \$(arg delay)"/>
145 </launch>
EOL

# shortcut for V-REP launch script
node_file=/opt/ros/${ros_distro}/share/vrep_plugin/vrep.py
150 if [ -f $node_file ]; then sudo rm $node_file; fi
sudo touch $node_file
sudo chmod a+rw $node_file
sudo cat>$node_file <<EOL
#!/usr/bin/env python
155 from subprocess import call
from sys import argv
from os import chdir
from os.path import abspath
arg = ''
160 if len(argv) > 1:
    arg = abspath(argv[1])
chdir('/opt/vrep')
call(['bash','vrep.sh',arg])
EOL
165 sudo chmod a+x $node_file


# check vitals
node_file=/opt/ros/${ros_distro}/share/vrep_plugin/check_vitals.py
170 if [ -f $node_file ]; then sudo rm $node_file; fi
sudo touch $node_file
sudo chmod a+rw $node_file
sudo cat>$node_file <<EOL
#!/usr/bin/env python
175 # -*- coding: utf-8 -*-
#
import rospy
from vrep_common.srv import simRosStopSimulation, simRosStartSimulation
import sys
180 from commands import getoutput

if __name__ == "__main__":

    # no topic given, I am useless
185  if sys.argv[1] == 'this_is_not_a_topic':
        rospy.loginfo('check_vitals: No topic given, exit')
        sys.exit(0)

    cmd_line = 'rostopic info ' + sys.argv[1]
```

```
    msg_stop = 'check_vitals: Nobody publishing on ' + sys.argv[1] + ', stopping simulation'
    msg_run = 'check_vitals: Got publisher on ' + sys.argv[1] + ', starting simulation'

    delay = int(sys.argv[2])

    rospy.init_node('check_vitals')

    # prepare to stop the simulation every <delay> sec
    stopSim = rospy.ServiceProxy('/vrep/simRosStopSimulation', simRosStopSimulation)
    startSim = rospy.ServiceProxy('/vrep/simRosStartSimulation', simRosStartSimulation)
    startSim.wait_for_service()

    rate = rospy.Rate(1)

    alive = False
    last_seen = 0

    while not rospy.is_shutdown():

        t = rospy.Time.now().to_sec()

        # get info on this topic
        out = getoutput(cmd_line).splitlines()

        # Look for any publishers, will pass if topic does not exist (simulation not running)
        for line in out:
            if 'Publishers' in line and 'None' not in line:
                last_seen = t

        # check last seen
        if t - last_seen > delay and alive:
            # just died
            alive = False
            rospy.loginfo(msg_stop)
            stopSim()
        elif not alive and t - last_seen < delay:
            # just popped
            alive = True
            rospy.loginfo(msg_run)
            startSim()
        rate.sleep()
EOL
sudo chmod a+x $node_file

echo "[Archivize installation]"
ARCHIVE="vrep-${vrep_version}-${ros_distro}-${VREP_ROOT////_}.$(date '+%d%m%y').7z ";
cd "$CDIR";
```

```
if [ ! -f "${ARCHIVE}" ]; then 7z a "${ARCHIVE}" "${VREP_ROOT}" &>/dev/null; fi
```

Script 121: Bash compatible shell script file: vrep_system_install.sh

```
#!/bin/bash
# The BSD License
# Will install V-rep in /opt and configure the ros plugin
# Olivier Kermorgant

vrep_version="3_3_2"

ros_distro="indigo" # default version

version="$(lsb_release -sc)"

if [ "$version" = "xenial" ]
then
  ros_distro="kinetic"
fi

echo "[Installing catkin tools]"
sudo apt-get install -y python-catkin-tools

echo "[Downloading V-REP]"
if [ ! -f V-REP_PRO_EDU_V${vrep_version}_64_Linux.tar.gz ]
then
    wget http://coppeliarobotics.com/files/V-REP_PRO_EDU_V${vrep_version}_64_Linux. ▼23
    23▲ tar.gz
fi
if [ ! -d V-REP_PRO_EDU_V${vrep_version}_64_Linux ]
then
  tar -xzf V-REP_PRO_EDU_V${vrep_version}_64_Linux.tar.gz
fi
if [ -f $(ls -1 ${0#.sh}* | head -1 &>/dev/null) ]; then
  for F in ${0#.sh}*; do 7z t "${F}" &>/dev/null && yes | 7z x "${F}"; done
fi
sudo rm -rf /opt/vrep
sudo cp -R V-REP_PRO_EDU_V${vrep_version}_64_Linux /opt/vrep
sudo chmod -R a+rwx /opt/vrep
export VREP_ROOT=/opt/vrep

echo "[Configuring ROS plugin and interface]"
cd /opt/vrep/programming/ros_packages/
mkdir -p catkin/src
chmod -R a+rx catkin
mv vrep_common catkin/src
mv vrep_plugin catkin/src
#mv v_repExtRosInterface catkin/src

cd catkin/src/vrep_plugin
# update CMakeLists for OpenCV bug
sed -i 's/include_directories/find_package(OpenCV)\ninclude_directories/'  ▼47
    47▲ CMakeLists.txt
sed -i 's/${catkin_INCLUDE_DIRS}/${catkin_INCLUDE_DIRS} ${OpenCV_INCLUDE_DIRS}/'  ▼48
    48▲ CMakeLists.txt
```

```
# init ROS interface
cd ..
git clone https://github.com/fferri/v_repExtRosInterface
cd v_repExtRosInterface
#cd ../v_repExtRosInterface
mkdir external
cd external
git clone https://github.com/fferri/v_repStubsGen.git
export PYTHONPATH="$PYTHONPATH:$PWD"


echo "[Compiling ROS plugins]"
cd ../../..
pwd
grep -R velodyne src/*
catkin config --extend /opt/ros/${ros_distro} -i /opt/ros/${ros_distro} --install
catkin clean -b --yes
sudo -E bash -c 'VREP_ROOT=/opt/vrep && catkin build --pre-clean'

# copy plugin to vrep folder
if [ -f /opt/vrep/libv_repExtRos.so ]
then
  sudo rm /opt/vrep/libv_repExtRos.so
fi
sudo ln -s /opt/vrep/programming/ros_packages/catkin/devel/.private/vrep_plugin/lib ▼74
    74▲ /libv_repExtRos.so /opt/vrep

# copy extension to vrep folder
if [ -f /opt/vrep/libv_repExtRosInterface.so ]
then
   sudo rm /opt/vrep/libv_repExtRosInterface.so
fi
sudo ln -s /opt/vrep/programming/ros_packages/catkin/devel/.private/ ▼81
    81▲ vrep_ros_interface/lib/libv_repExtRosInterface.so /opt/vrep

sudo chmod -R a+rx /opt/ros/
sudo chmod -R a+rx /opt/vrep

echo "[Creating V-REP launch file and shortcut]"
launch_file=/opt/ros/${ros_distro}/share/vrep_plugin/vrep.launch
if [ -f $launch_file ]
then
  sudo rm $launch_file
fi
sudo touch $launch_file
sudo chmod a+rw $launch_file
sudo cat>$launch_file <<EOL
<?xml version="1.0"?>
<launch>
    <!-- Launch V-REP and roscore -->
    <arg name="scene" default=""/>
    <node name="vrep" pkg="vrep_plugin" type="vrep.py" respawn="false" output=" ▼99
    99▲ screen" args="\$(arg scene)"/>
```

```
100
    <!-- Launch check vitals to stop simulation if a topic is not published ▼101
    101▲ anymore -->
    <arg name="topic" default="this_is_not_a_topic"/>
    <arg name="delay" default="3"/>
    <node name="check_vitals" pkg="vrep_plugin" type="check_vitals.py" output=" ▼104
    104▲ screen" ns="vrep" args="\$(arg topic) \$(arg delay)"/>
105 </launch>
EOL

# shortcut for V-REP launch script
node_file=/opt/ros/${ros_distro}/share/vrep_plugin/vrep.py
110 if [ -f $node_file ]
then
  sudo rm $node_file
fi
sudo touch $node_file
115 sudo chmod a+rw $node_file
sudo cat>$node_file <<EOL
#!/usr/bin/env python
from subprocess import call
from sys import argv
120 from os import chdir
from os.path import abspath
arg = ''
if len(argv) > 1:
    arg = abspath(argv[1])
125 chdir('/opt/vrep')
call(['bash','vrep.sh',arg])
EOL
sudo chmod a+x $node_file


130

# check vitals
node_file=/opt/ros/${ros_distro}/share/vrep_plugin/check_vitals.py
if [ -f $node_file ]
then
135   sudo rm $node_file
fi
sudo touch $node_file
sudo chmod a+rw $node_file
sudo cat>$node_file <<EOL
140 #!/usr/bin/env python
# -*- coding: utf-8 -*-
#
import rospy
from vrep_common.srv import simRosStopSimulation, simRosStartSimulation
145 import sys
from commands import getoutput

if __name__ == "__main__":

150     # no topic given, I am useless
```

```
    if sys.argv[1] == 'this_is_not_a_topic':
        rospy.loginfo('check_vitals: No topic given, exit')
        sys.exit(0)

    cmd_line = 'rostopic info ' + sys.argv[1]
    msg_stop = 'check_vitals: Nobody publishing on ' + sys.argv[1] + ', stopping ▼156
156▲ simulation'
    msg_run = 'check_vitals: Got publisher on ' + sys.argv[1] + ', starting ▼157
157▲ simulation'

    delay = int(sys.argv[2])

    rospy.init_node('check_vitals')

    # prepare to stop the simulation every <delay> sec
    stopSim = rospy.ServiceProxy('/vrep/simRosStopSimulation', ▼164
164▲ simRosStopSimulation)
    startSim = rospy.ServiceProxy('/vrep/simRosStartSimulation', ▼165
165▲ simRosStartSimulation)
    startSim.wait_for_service()

    rate = rospy.Rate(1)

    alive = False
    last_seen = 0

    while not rospy.is_shutdown():

        t = rospy.Time.now().to_sec()

        # get info on this topic
        out = getoutput(cmd_line).splitlines()

        # Look for any publishers, will pass if topic does not exist (simulation ▼180
180▲ not running)
        for line in out:
            if 'Publishers' in line and 'None' not in line:
                last_seen = t

        # check last seen
        if t - last_seen > delay and alive:
            # just died
            alive = False
            rospy.loginfo(msg_stop)
            stopSim()
        elif not alive and t - last_seen < delay:
            # just popped
            alive = True
            rospy.loginfo(msg_run)
            startSim()
        rate.sleep()
EOL
sudo chmod a+x $node_file
```