



CoppeliaSim — Inverse Kinematics — FUNZIONANTE

Basics

Esempio "da tutorial"


Implementazione di una inverse Kinematic funzionante

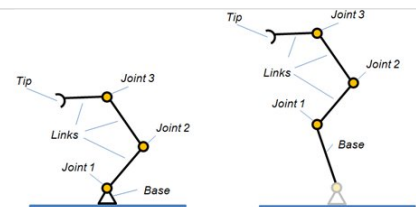
Basics

Docuemntazione ufficiale di Vrep.

Basics on IK groups and IK elements

CoppeliaSim uses IK groups and IK elements to solve inverse and forward kinematics tasks. It is important to understand how an IK task is solved in order to take full advantage of the kinematics functionality in CoppeliaSim. Make sure to

 <https://www.coppeliarobotics.com/helpFiles/en/basicsOnIkGroupsAndIkElements.htm>



Terminologia:

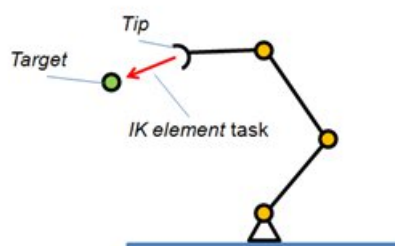
- tip: l'end effector della catena cinematica

The tip is usually the last object in the considered kinematic chain (when going from the base to the tip), and is often the end-effector. The tip object should be linked to a target object

- base: il primo oggetto della catena cinematica, la root
- target: il frame che il tip deve seguire

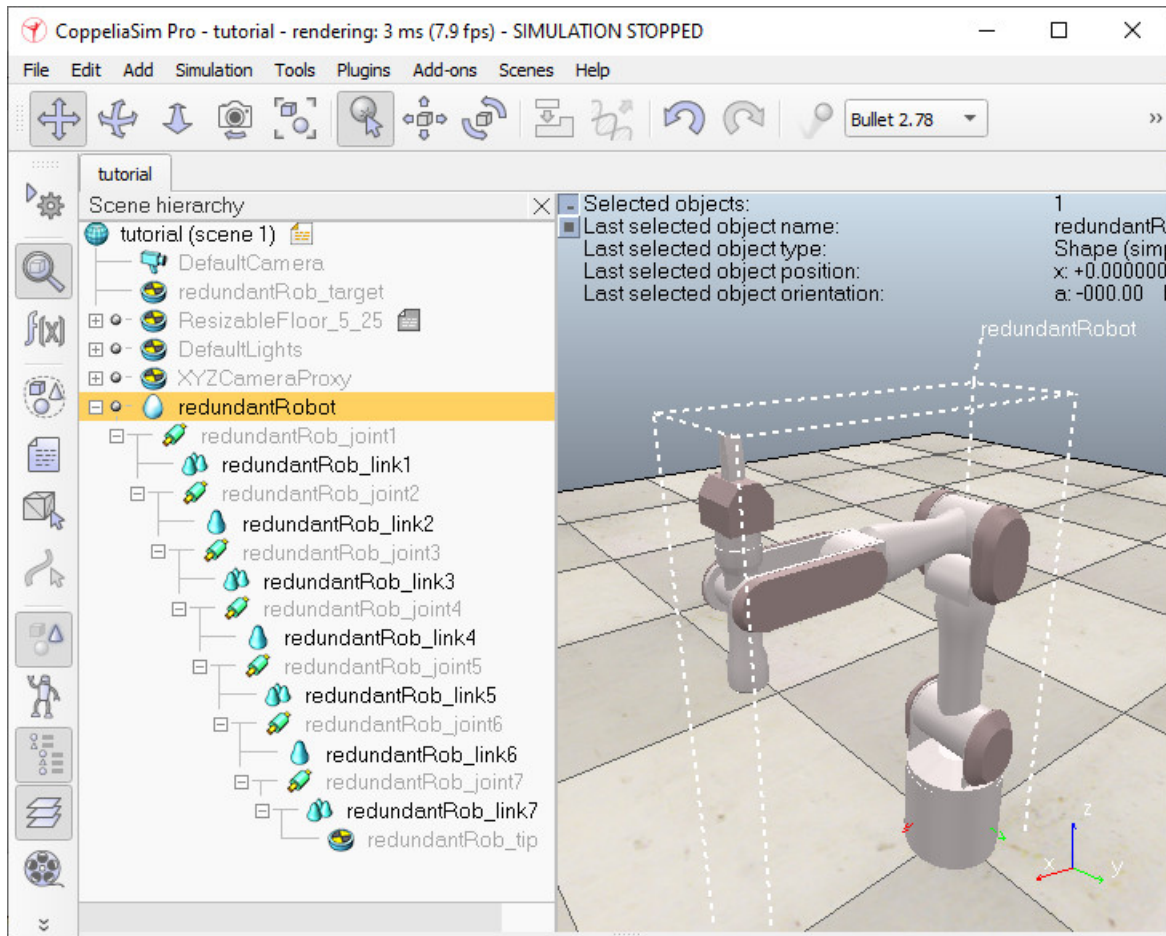
a **target**. The target represents the position and/or orientation the tip should adopt (or follow) when solving for IK. The target object should be linked to a tip object.

La Inverse Kinematics di CoppeliaSim imlementa l'inseguimento del target da parte del tip.



Esempio "da tutorial"

Supponiamo di avere questa gerarchia iniziale:



Inizializzazione della kinematic chain (versione *simple*):

```
-- handlers riferiti a basem tip e target
local simBase=sim.getObjectHandle('redundantRobot') --> base
local simTip=sim.getObjectHandle('redundantRob_tip') --> tip
local simTarget=sim.getObjectHandle('redundantRob_target') --> target

-- l'environment è unico per tutti i gruppi
ikEnv=simIK.createEnvironment()

-- creazione di una kinematic chain con un parametro
ikGroup_undamped=simIK.createIkGroup(ikEnv)
simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped,simIK.method_pseudo_inverse,0,6)
-- NOTA: 'simIK.constraint_pose' OVVERO tenta di far coincidere il frame
--       del tip con quello del target
simIK.addIkElementFromScene(ikEnv,ikGroup_undamped,simBase,simTip,simTarget,simIK.constraint_pose)

-- è possibile creare più gruppi cinematici nello stesso environment
ikGroup_damped=simIK.createIkGroup(ikEnv)
simIK.setIkGroupCalculation(ikEnv,ikGroup_damped,simIK.method_damped_least_squares,1,99)
simIK.addIkElementFromScene(ikEnv,ikGroup_damped,simBase,simTip,simTarget,simIK.constraint_pose)
```

Attuazione:

```

if simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped,true)==simIK.result_fail then
    print( "actuation --> FAILED" )
else
    print( "actuation --> OK" )
end if

-- versione con damped e undamped

```

ricorda la cleanup:

```

simIK.eraseEnvironment(ikEnv)

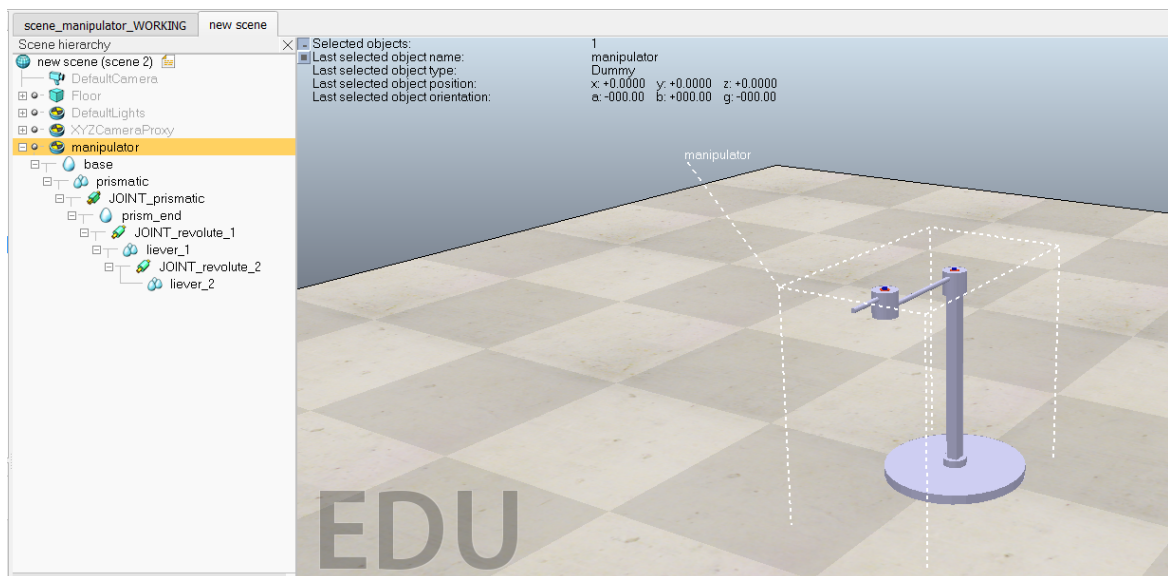
```

Implementazione di una inverse Kinematic funzionante

Il modello del manipolatore (vedi es. 1):

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/15c41968-e39c-4530-8fb4-551247bf094a/IK_manipulator.ttm

Partiamo da un progetto vuoto, con solo la struttura di un manipolatore all'interno.



Prima di iniziare, tutti i joint devono essere regolati correttamente:

prismatic:

Joint Dynamic Properties

Motor properties

☒ Motor enabled

Target velocity [m/s]

Maximum force [N]

☐ Lock motor when target velocity is zero

[Edit engine specific properties](#)

[Apply to selection](#)

Control properties

☒ Control loop enabled

Target position [m]

Upper velocity limit [m/s]

☒ Position control (PID)

Proportional parameter

Integral parameter

Derivative parameter

☐ Spring-damper mode

Spring constant K [N/m]

Damping coefficient C [N*s/m]

[Apply to selection](#)

revolute 1:

Joint Dynamic Properties

Motor properties

☒ Motor enabled

Target velocity [deg/s]

Maximum torque [N*m]

☐ Lock motor when target velocity is zero

[Edit engine specific properties](#)

[Apply to selection](#)

Control properties

☒ Control loop enabled

Target position [deg]

Upper velocity limit [deg/s]

☒ Position control (PID)

Proportional parameter

Integral parameter

Derivative parameter

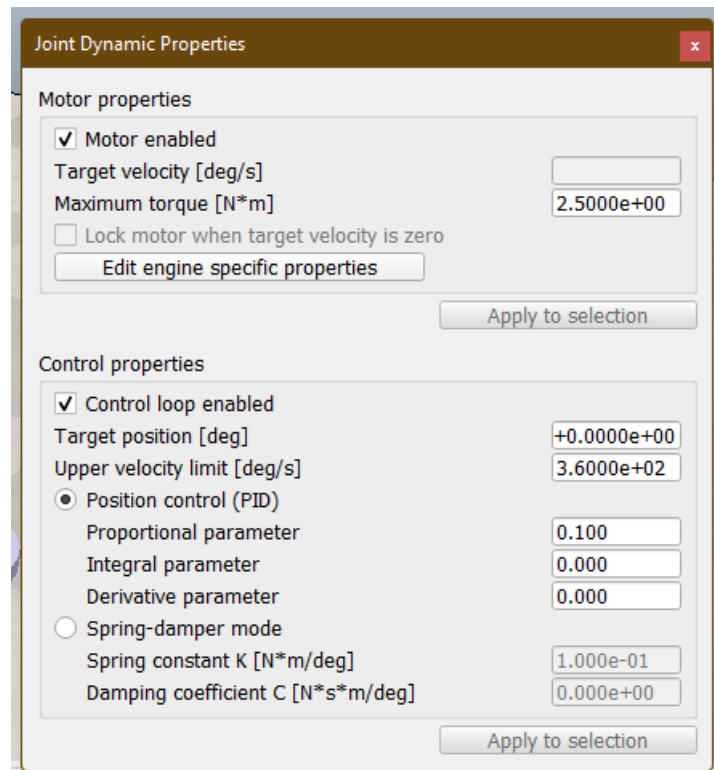
☐ Spring-damper mode

Spring constant K [N*m/deg]

Damping coefficient C [N*s*m/deg]

[Apply to selection](#)

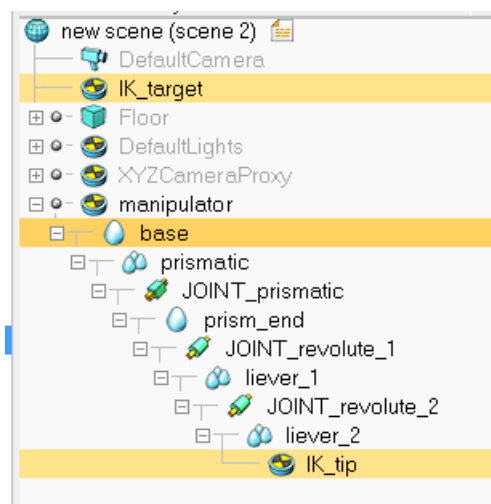
revolute 2:



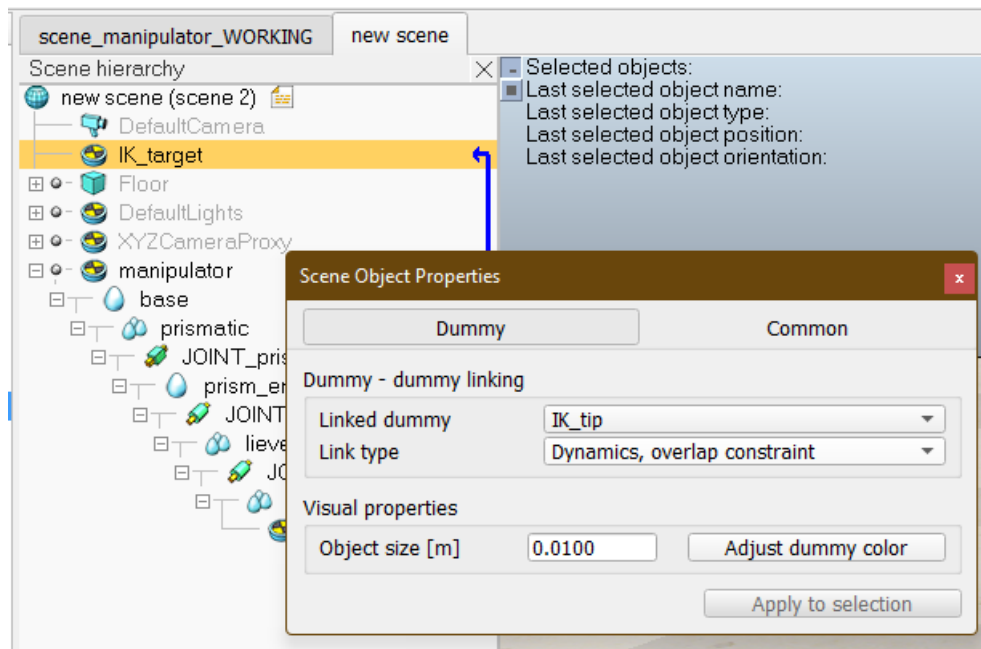
A questo punto, andiamo ad aggiungere 2 dummies:

- il *tip*, cioè il punto che dovrà allinearsi col target
- il *target*, vale a dire il punto che il manipolatore deve tentare di inseguire adattando la propria configurazione

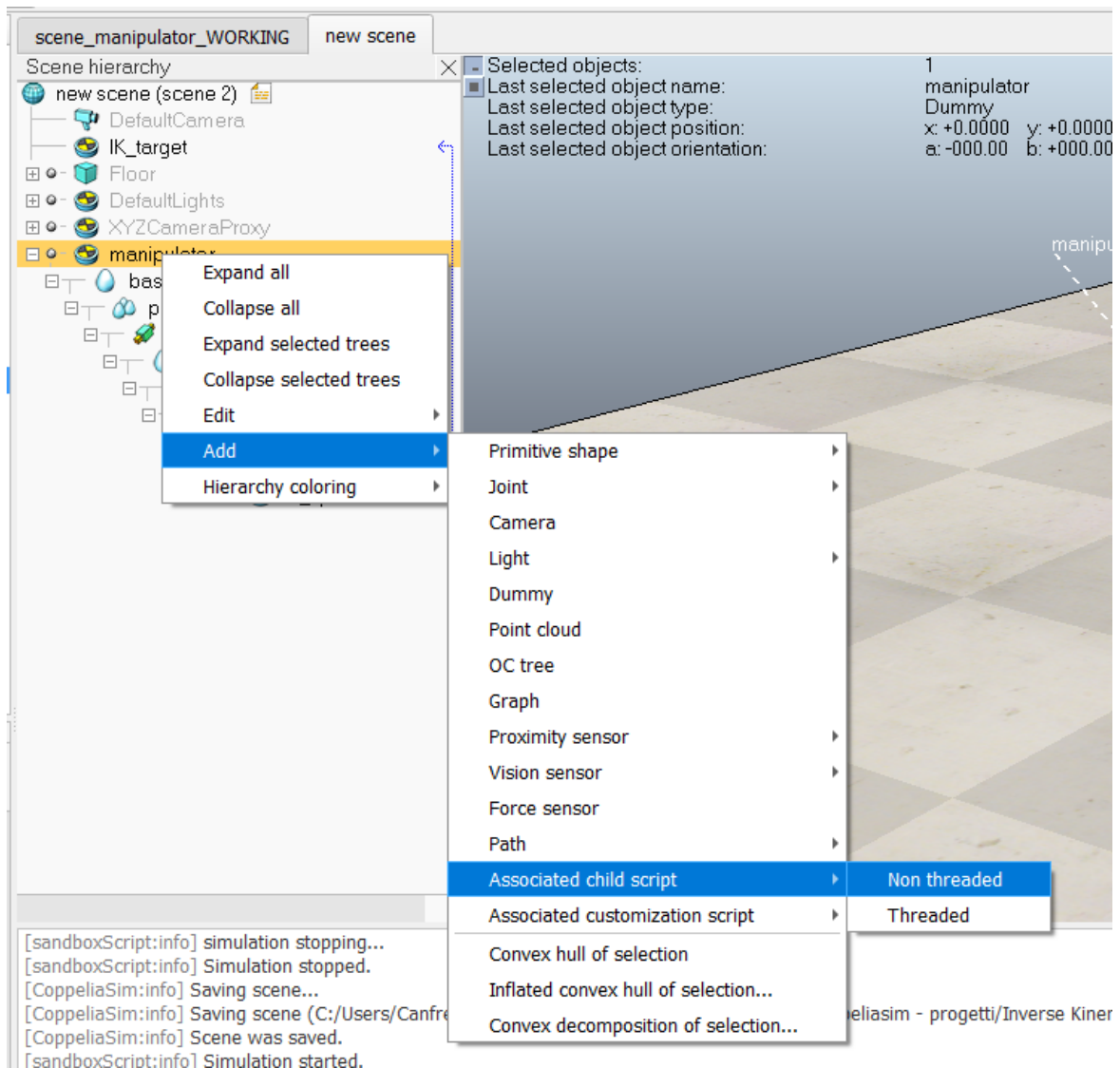
La base è direttamente un pezzo del manipolatore.



Ricorda di linkare tip e target, altrimenti l'inseguimento non funzionerà:



A questo punto, siamo pronti per inserire il codice. Nel tutorial si trova del codice migliore; ne propongo una versione semplificata.



```
function ik_init()
    local simBase=sim.getObjectHandle('manipulator') --> base
    local simTip=sim.getObjectHandle('IK_tip') --> tip
    local simTarget=sim.getObjectHandle('IK_target') --> target

    ikEnv=simIK.createEnvironment()

    -- ricerca di una soluzione forte
    ikGroup_undamped=simIK.createIkGroup(ikEnv)
    simIK.setIkGroupCalculation(ikEnv, ikGroup_undamped, simIK.method_pseudo_inverse, 0, 6)
    simIK.addIkElementFromScene(ikEnv, ikGroup_undamped, simBase, simTip, simTarget, simIK.constraint_pose)
    -- ricerca di una soluzione pi? debole
    ikGroup_damped=simIK.createIkGroup(ikEnv)
    simIK.setIkGroupCalculation(ikEnv, ikGroup_damped, simIK.method_damped_least_squares, 1, 99)
    simIK.addIkElementFromScene(ikEnv, ikGroup_damped, simBase, simTip, simTarget, simIK.constraint_pose)
end

function ik_actuate()
    if simIK.applyIkEnvironmentToScene(ikEnv, ikGroup_undamped, true)==simIK.result_fail then
        simIK.applyIkEnvironmentToScene(ikEnv, ikGroup_damped)
    end
end

function sysCall_init()
    ik_init()
end
```

```
function sysCall_actuation()
    ik_actuate()
end

function sysCall_cleanup()
    simIK.eraseEnvironment(ikEnv)
end
```

Il risultato finale:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/df93c384-d003-420b-89cb-f1e15f6d801c/scene_manipulator_2.ttt