

# Using Tensor Flow to Find and Grab Rover Ruckus Gold Minerals

```
to runOpMode
  Initialization
  call Telemetry . addData
    key "Init "
    text "started "
  call Telemetry . update
  set LeftMotor . Direction to Direction . REVERSE
  set UpperServo . Position to 1
  set LowerServo . Position to 0
  Init Vuforia because Tensor Flow needs it.
  call Vuforia . initialize
    cameraDirection CameraDirection . BACK
    useExtendedTracking true
    enableCameraMonitoring false
    cameraMonitorFeedback CameraMonitorFeedback . AXES
    phoneLocationOnRobot translation dx 0
    phoneLocationOnRobot translation dy 0
    phoneLocationOnRobot translation dz 0
    phoneLocationOnRobot rotation x 0
    phoneLocationOnRobot rotation y 0
    phoneLocationOnRobot rotation z 0
    useCompetitionFieldTargetLocations true
  call Telemetry . addData
    key "Vuforia "
    text "initialized "
  call Telemetry . update
  Let's use 70% minimum confidence and
  and no object tracker.
  call TensorFlowObjectDetection . initialize
    minimumConfidence 0.7
    useObjectTracker false
    enableCameraMonitoring false
  call Telemetry . addData
    key "> "
    text "Press Play to start "
```

```
call Telemetry . update
Set target ratio of object height to image
height value corresponding to the length
of the robot's neck.
set TargetHeightRatio to 0.55
call SeekGoldConverge . waitForStart
call TensorFlowObjectDetection . activate
We'll loop until gold block captured or time is up
set BlockCaptured to false
repeat while call SeekGoldConverge . opModelsActive and not BlockCaptured
do Get list of current recognitions.
set recognitions to call TensorFlowObjectDetection . getRecognitions
Report number of recognitions.
call Telemetry . addData
key " Objects Recognized "
number length of recognitions
If some objects detected...
if length of recognitions > 0
do ...let's count how many are gold.
set GoldCount to 0
Step through the gold & silver minerals detected.
for each item recognition in list recognitions
do if Recognition . Label = Label . Gold Mineral
do A gold mineral has been detected.
set GoldCount to GoldCount + 1
We can assume this is the first detected gold
because we break out of this loop below after
using the information from the first gold.
We don't need to calculate turn angle to gold
because TensorFlow has estimated it for us.
set ObjectAngle to call Recognition . estimateAngleToObject
recognition recognition
angleUnit AngleUnit . DEGREES
```

A negative angle means gold is left, else right.

call Telemetry . addData  
key " Estimated Angle "  
number ObjectAngle

if ObjectAngle > 0  
do call Telemetry . addData  
key " Direction "  
text " Right "  
else call Telemetry . addData  
key " Direction "  
text " Left "

Calculate power levels for turn toward gold.

set LeftPower to  $0.25 \times \text{ObjectAngle} \div 45$   
set RightPower to  $-0.25 \times \text{ObjectAngle} \div 45$

We'll be comparing the gold mineral height  
to the height of the video image to estimate  
how close the robot is to the gold mineral.

set ImageHeight to Recognition . ImageHeight  
recognition recognition  
set ObjectHeight to Recognition . Height  
recognition recognition

Calculate height of gold relative to image height.

Larger ratio means robot is closer to gold.

set ObjectHeightRatio to ObjectHeight  $\div$  ImageHeight  
call Telemetry . addData  
key " HeightRatio "  
number ObjectHeightRatio

Use height ratio to determine distance.

If height ratio larger than (target - tolerance)...

if ObjectHeightRatio < TargetHeightRatio - 0.05  
do ...not close enough yet.

call Telemetry . addData  
key " Distance "  
text " Not close enough "

If sum of turn powers are small

if  $|\text{LeftPower}| + |\text{RightPower}| < 0.12$

do ...don't really need to turn. Move forward.

call Telemetry . addData  
key " Action "  
text " Forward "

Go forward by setting power proportional to how far from target distance.

set LeftPower to  $0.035 + 0.5 \times (\text{TargetHeightRatio} - 0.05) \times \text{ObjectHeightRatio}$

set RightPower to LeftPower

else Else we'll turn to gold with current power levels.

call Telemetry . addData  
key " Action "  
text " Turn "

Else if height ratio more than (target+tolerance...

else if  $\text{ObjectHeightRatio} > \text{TargetHeightRatio} + 0.05$

do ...robot too close to gold.

call Telemetry . addData  
key " Distance "  
text " Too close "

If calculated turn power levels are small...

if  $|\text{LeftPower}| + |\text{RightPower}| < 0.12$

do ...don't need to turn. Backup instead by setting

power proportional to how far past target ratio

call Telemetry . addData  
key " Action "  
text " Back up "

```
set LeftPower to (-0.05 + (-0.5 * (TargetHeightRatio + 0.05 - TargetHeightRatio)))
set RightPower to LeftPower
else
  Else use current power levels to turn to gold.
  call Telemetry . addData
    key "Action"
    text "Turn"
else
  Gold is one next length away.
  call Telemetry . addData
    key "Distance"
    text "Correct"
  If calculated turn power levels are small...
  if (absolute LeftPower + absolute RightPower < 0.12)
  do
    ...robot is centered on gold.
    call Telemetry . addData
      key "Action"
      text "Motors off, grab the gold"
    Turn motors off by setting power to 0.
    set LeftPower to 0
    set RightPower to 0
    Lower neck and open jaw.
    set LowerServo . Position to 0.5
    set UpperServo . Position to 0.5
    set BlockCaptured to true
  else
    Otherwise use current power levels to turn
    to better center on gold.
    call Telemetry . addData
      key "Action"
      text "Turn"
  call Telemetry . addData
    key "Left Power"
```

```
number LeftPower
call Telemetry . addData
  key "Right Power"
  number RightPower
Set power levels to get closer to gold object.
set Power
  LeftMotor to LeftPower
  RightMotor to RightPower
We've found a gold so we don't have
to look at rest of detected objects.
Break out of For-each-recognition.
break out of loop

If no gold objects detected...
if GoldCount = 0
do
  call Telemetry . addData
    key "Status"
    text "No Gold"
  call Telemetry . addData
    key "Action"
    text "Back up"
  Back up slowly hoping to bring gold in view.
  set Power
    LeftMotor to -0.1
    RightMotor to -0.1
else
  No objects detected
  call Telemetry . addData
    key "Status"
    text "No objects detected"
  call Telemetry . addData
    key "Action"
    text "Back up"
  Back up slowly hoping to bring objects in view.
  set Power
```

The code is written in a Scratch-like block-based language. It starts with a loop that sets the power levels for the left and right motors to 'LeftPower' and 'RightPower' respectively. It then sends a message to 'Telemetry' to add data with the key 'Right Power' and the value 'RightPower'. A comment indicates that the power levels are set to get closer to a gold object. The code then sets the 'Power' variable and assigns 'LeftPower' to the 'LeftMotor' and 'RightPower' to the 'RightMotor'. A message is sent to 'Telemetry' with the key 'Status' and the text 'No Gold'. Another message is sent to 'Telemetry' with the key 'Action' and the text 'Back up'. A comment says 'Back up slowly hoping to bring gold in view.' The code then sets the 'Power' variable and assigns -0.1 to both the 'LeftMotor' and 'RightMotor'. An 'else' block handles the case where no gold objects are detected. It sends a message to 'Telemetry' with the key 'Status' and the text 'No objects detected'. Another message is sent to 'Telemetry' with the key 'Action' and the text 'Back up'. A comment says 'Back up slowly hoping to bring objects in view.' The code then sets the 'Power' variable.

```
LeftMotor to -0.1
RightMotor to -0.1
call Telemetry . update
Gold captured, time is up or stop was requested.
set Power
LeftMotor to 0
RightMotor to 0
call TensorFlowObjectDetection . deactivate
Pause to let driver station to see last telemetry.
call SeekGoldConverge . sleep
milliseconds 2000
```

The image shows a sequence of Scratch code blocks. It starts with two 'set' blocks for 'LeftMotor' and 'RightMotor' both set to '-0.1'. This is followed by a 'call' block for 'Telemetry . update'. A blue comment block reads 'Gold captured, time is up or stop was requested.'. Then, a 'set' block for 'Power' is shown. Below it are two more 'set' blocks for 'LeftMotor' and 'RightMotor', both set to '0'. This is followed by a 'call' block for 'TensorFlowObjectDetection . deactivate'. Another blue comment block reads 'Pause to let driver station to see last telemetry.'. The final block is a 'call' block for 'SeekGoldConverge . sleep' with a sub-block for 'milliseconds' set to '2000'.