**1.READ AND ROTATE THE IMAGE**

**PUBLISHER:**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

import cv2

from cv\_bridge import CvBridge

import os

class ImagePublisher:

def \_\_init\_\_(self):

rospy.init\_node('image\_publisher', anonymous=True)

self.image\_pub = rospy.Publisher('image\_topic', Image, queue\_size=10)

self.bridge = CvBridge()

self.rate = rospy.Rate(1) # Publish rate of 1 Hz

self.image\_folder = "/home/rae/Downloads"

def publish\_image(self):

while not rospy.is\_shutdown():

for root, dirs, files in os.walk(self.image\_folder):

for file in files:

if file.endswith(".jpeg") or file.endswith(".png"):

image\_path = os.path.join(root, file)

cv\_image = cv2.imread(image\_path)

if cv\_image is not None:

# Rotate image

rotated\_image = cv2.rotate(cv\_image, cv2.ROTATE\_90\_CLOCKWISE)

ros\_image = self.bridge.cv2\_to\_imgmsg(rotated\_image, "bgr8")

self.image\_pub.publish(ros\_image)

rospy.loginfo("Image published: {}".format(image\_path))

self.rate.sleep()

def main():

try:

image\_publisher = ImagePublisher()

image\_publisher.publish\_image()

except rospy.ROSInterruptException:

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

**SUBSCRIBER:**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

class ImageSubscriber:

def \_\_init\_\_(self):

rospy.init\_node('image\_subscriber', anonymous=True)

self.bridge = CvBridge()

rospy.Subscriber('image\_topic', Image, self.image\_callback)

self.cv\_window\_name = 'Subscribed Image'

def image\_callback(self, msg):

try:

cv\_image = self.bridge.imgmsg\_to\_cv2(msg, desired\_encoding="bgr8")

# Display image

cv2.imshow(self.cv\_window\_name, cv\_image)

cv2.waitKey(1)

except Exception as e:

rospy.logerr(e)

def run(self):

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

try:

rospy.init\_node('image\_subscriber', anonymous=True)

image\_subscriber = ImageSubscriber()

image\_subscriber.run()

except rospy.ROSInterruptException:

cv2.destroyAllWindows()

**2.Read an image into RoS and perform colour conversions on an image.**

**PUBLISHER**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

import os

class ImagePublisher:

def \_\_init\_\_(self):

rospy.init\_node('image\_publisher', anonymous=True)

self.image\_pub = rospy.Publisher('image\_topic', Image, queue\_size=10)

self.bridge = CvBridge()

self.rate = rospy.Rate(1) # Publish rate of 1 Hz

self.image\_folder = "/home/rae/Downloads"

def publish\_image(self):

while not rospy.is\_shutdown():

for root, dirs, files in os.walk(self.image\_folder):

for file in files:

if file.endswith(".jpeg") or file.endswith(".png"):

image\_path = os.path.join(root, file)

cv\_image = cv2.imread(image\_path)

if cv\_image is not None:

ros\_image = self.bridge.cv2\_to\_imgmsg(cv\_image, "bgr8")

self.image\_pub.publish(ros\_image)

rospy.loginfo("Image published: {}".format(image\_path))

self.rate.sleep()

def main():

try:

image\_publisher = ImagePublisher()

image\_publisher.publish\_image()

except rospy.ROSInterruptException:

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

**SUBSCRIBER:**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

class ImageSubscriber:

def \_\_init\_\_(self):

rospy.init\_node('image\_subscriber', anonymous=True)

self.bridge = CvBridge()

rospy.Subscriber('image\_topic', Image, self.image\_callback)

self.cv\_window\_name = 'Subscribed Image'

def image\_callback(self, msg):

try:

cv\_image = self.bridge.imgmsg\_to\_cv2(msg, desired\_encoding="bgr8")

# Perform color conversion

gray\_image = cv2.cvtColor(cv\_image, cv2.COLOR\_BGR2GRAY)

# Display original and converted images

cv2.imshow("Original Image", cv\_image)

cv2.imshow("Gray Image", gray\_image)

cv2.waitKey(1)

except Exception as e:

rospy.logerr(e)

def run(self):

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

try:

rospy.init\_node('image\_subscriber', anonymous=True)

image\_subscriber = ImageSubscriber()

image\_subscriber.run()

except rospy.ROSInterruptException:

cv2.destroyAllWindows()

**3.Write AI and vison on an image.**

**PUBLISHER:**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

import os

class ImagePublisher:

def \_\_init\_\_(self):

rospy.init\_node('image\_publisher', anonymous=True)

self.image\_pub = rospy.Publisher('image\_topic', Image, queue\_size=10)

self.bridge = CvBridge()

self.rate = rospy.Rate(1) # Publish rate of 1 Hz

self.image\_folder = "/home/rae/Downloads"

def publish\_image(self):

while not rospy.is\_shutdown():

for root, dirs, files in os.walk(self.image\_folder):

for file in files:

if file.endswith(".jpeg") or file.endswith(".png"):

image\_path = os.path.join(root, file)

cv\_image = cv2.imread(image\_path)

if cv\_image is not None:

ros\_image = self.bridge.cv2\_to\_imgmsg(cv\_image, "bgr8")

self.image\_pub.publish(ros\_image)

rospy.loginfo("Image published: {}".format(image\_path))

self.rate.sleep()

def main():

try:

image\_publisher = ImagePublisher()

image\_publisher.publish\_image()

except rospy.ROSInterruptException:

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

**SUBSCRIBER:**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

class ImageSubscriber:

def \_\_init\_\_(self):

rospy.init\_node('image\_subscriber', anonymous=True)

self.bridge = CvBridge()

rospy.Subscriber('image\_topic', Image, self.image\_callback)

self.cv\_window\_name = 'Subscribed Image'

def image\_callback(self, msg):

try:

cv\_image = self.bridge.imgmsg\_to\_cv2(msg, desired\_encoding="bgr8")

# Add text to the image

font = cv2.FONT\_HERSHEY\_SIMPLEX

cv2.putText(cv\_image, 'AI and Vision', (20, 50), font, 1, (255, 255, 255), 2, cv2.LINE\_AA)

# Display the image

cv2.imshow(self.cv\_window\_name, cv\_image)

cv2.waitKey(1)

except Exception as e:

rospy.logerr(e)

def run(self):

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

try:

rospy.init\_node('image\_subscriber', anonymous=True)

image\_subscriber = ImageSubscriber()

image\_subscriber.run()

except rospy.ROSInterruptException:

cv2.destroyAllWindows()

**4.Find the difference between the two images.**

**PUBLSIHER**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

import os

class ImagePublisher:

def \_\_init\_\_(self):

rospy.init\_node('image\_publisher', anonymous=True)

self.image\_pub = rospy.Publisher('image\_topic', Image, queue\_size=10)

self.bridge = CvBridge()

self.rate = rospy.Rate(1) # Publish rate of 1 Hz

self.image\_folder = "/home/rae/Downloads"

def publish\_images(self):

while not rospy.is\_shutdown():

for root, dirs, files in os.walk(self.image\_folder):

for file in files:

if file.endswith(".jpeg") or file.endswith(".png"):

image\_path = os.path.join(root, file)

cv\_image = cv2.imread(image\_path)

if cv\_image is not None:

ros\_image = self.bridge.cv2\_to\_imgmsg(cv\_image, "bgr8")

self.image\_pub.publish(ros\_image)

rospy.loginfo("Image published: {}".format(image\_path))

self.rate.sleep()

def main():

try:

image\_publisher = ImagePublisher()

image\_publisher.publish\_images()

except rospy.ROSInterruptException:

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

**SUBSCRIBER:**

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

class ImageSubscriber:

def \_\_init\_\_(self):

rospy.init\_node('image\_subscriber', anonymous=True)

self.bridge = CvBridge()

self.first\_image\_received = False

self.first\_image = None

rospy.Subscriber('image\_topic', Image, self.image\_callback)

self.cv\_window\_name = 'Difference Image'

def image\_callback(self, msg):

try:

cv\_image = self.bridge.imgmsg\_to\_cv2(msg, desired\_encoding="bgr8")

if not self.first\_image\_received:

self.first\_image = cv\_image

self.first\_image\_received = True

else:

# Calculate absolute difference between two images

difference\_image = cv2.absdiff(self.first\_image, cv\_image)

# Display the difference image

cv2.imshow(self.cv\_window\_name, difference\_image)

cv2.waitKey(1)

except Exception as e:

rospy.logerr(e)

def run(self):

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

try:

rospy.init\_node('image\_subscriber', anonymous=True)

image\_subscriber = ImageSubscriber()

image\_subscriber.run()

except rospy.ROSInterruptException:

cv2.destroyAllWindows()

**5. Write a python program in RoS to sort a given set of numbers.**

**PUBLISHER**

#!/usr/bin/env python3

import rospy

from std\_msgs.msg import Int32MultiArray

import random

def publish\_numbers():

rospy.init\_node('sorting\_publisher', anonymous=True)

numbers\_pub = rospy.Publisher('numbers\_to\_sort', Int32MultiArray, queue\_size=10)

rate = rospy.Rate(1) # 1 Hz

while not rospy.is\_shutdown():

# Generate a list of random numbers

numbers = [random.randint(0, 100) for \_ in range(10)]

# Publish the numbers

numbers\_pub.publish(Int32MultiArray(data=numbers))

rate.sleep()

if \_\_name\_\_ == '\_\_main\_\_':

try:

publish\_numbers()

except rospy.ROSInterruptException:

pass

**SUBSCRIBER**

#!/usr/bin/env python3

import rospy

from std\_msgs.msg import Int32MultiArray

def sort\_numbers(numbers):

sorted\_numbers = sorted(numbers)

return sorted\_numbers

def numbers\_callback(msg):

numbers = msg.data

sorted\_numbers = sort\_numbers(numbers)

rospy.loginfo("Received numbers: {}".format(numbers))

rospy.loginfo("Sorted numbers: {}".format(sorted\_numbers))

def subscribe\_to\_numbers():

rospy.init\_node('sorting\_subscriber', anonymous=True)

rospy.Subscriber('numbers\_to\_sort', Int32MultiArray, numbers\_callback)

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

try:

subscribe\_to\_numbers()

except rospy.ROSInterruptException:

pass

**6.Stream the video from USB camera in RoS and write your name on the stream.**

**PUBLISHER:**

#!/usr/bin/env python3

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

class VideoPublisher:

def \_\_init\_\_(self):

rospy.init\_node('video\_publisher', anonymous=True)

self.bridge = CvBridge()

self.cap = cv2.VideoCapture(0) # Use camera index 0 for the first USB camera

self.image\_pub = rospy.Publisher('usb\_camera/image\_raw', Image, queue\_size=10)

def publish\_video(self):

while not rospy.is\_shutdown():

ret, frame = self.cap.read()

if ret:

frame\_with\_text = self.add\_text\_to\_frame(frame)

ros\_image = self.bridge.cv2\_to\_imgmsg(frame\_with\_text, "bgr8")

self.image\_pub.publish(ros\_image)

def add\_text\_to\_frame(self, frame):

# Add your name as text to the frame

name = "Your Name"

font = cv2.FONT\_HERSHEY\_SIMPLEX

cv2.putText(frame, name, (10, 30), font, 1, (0, 255, 255), 2, cv2.LINE\_AA)

return frame

def main():

try:

video\_publisher = VideoPublisher()

video\_publisher.publish\_video()

except rospy.ROSInterruptException:

pass

if \_\_name\_\_ == '\_\_main\_\_':

main()

**SUBSCRIBER:**

#!/usr/bin/env python3

import rospy

from sensor\_msgs.msg import Image

from cv\_bridge import CvBridge

import cv2

class VideoSubscriber:

def \_\_init\_\_(self):

rospy.init\_node('video\_subscriber', anonymous=True)

self.bridge = CvBridge()

rospy.Subscriber('usb\_camera/image\_raw', Image, self.video\_callback)

self.cv\_window\_name = 'Video Stream'

def video\_callback(self, msg):

try:

cv\_image = self.bridge.imgmsg\_to\_cv2(msg, desired\_encoding="bgr8")

# Display the video stream

cv2.imshow(self.cv\_window\_name, cv\_image)

cv2.waitKey(1)

except Exception as e:

rospy.logerr(e)

def run(self):

rospy.spin()

if \_\_name\_\_ == '\_\_main\_\_':

try:

video\_subscriber = VideoSubscriber()

video\_subscriber.run()

except rospy.ROSInterruptException:

cv2.destroyAllWindows()