**CV Practical No.: 9**

**Aim:** **2D to 3D conversion**

**New Concept:**

**i. PIL:** PIL is a library in Python that adds image processing capabilities. It lets you open, manipulate, and save many different image file formats. PIL is usually accessed through its fork: Pillow.

**ii. img.convert:** This method is used to convert an image to a different color mode, such as:

* 'L' for grayscale,
* 'RGB' for true color,
* 'RGBA' for color with transparency,
* '1' for black and white (binary).

**iii. shift\_image:** Not a built-in function but usually a custom function in Computer Vision used to translate (shift) an image along the x and/or y axes.

**iv. depth\_data:** Refers to data representing depth (distance from the camera to each point in the scene).  
Common in applications like 3D reconstruction, stereo vision, or using depth sensors (e.g., Kinect, LiDAR). It is a 2D NumPy array where each value represents depth at a pixel.

**v. shift\_amount:** Usually refers to the number of pixels by which an image or data is to be shifted in shift\_image or similar operations. It can be a tuple like (dx, dy) for horizontal and vertical shifts.

**vi. np.zeros\_like(data):** Creates a NumPy array of zeros with the same shape and type as data. Useful when initializing an empty image or depth map.

**vii. data.shape:** This gives the dimensions of the data, typically a NumPy array. In Computer Vision:

* For grayscale image: (height, width)
* For RGB image: (height, width, 3)

**viii. Image.fromarray:** A method from PIL that creates an Image object from a NumPy array.

**ix. Image.open:** This is used to open an image file and load it as a PIL Image object.

**Theory:**

2D images contain intensity/color information at each pixel, but no information about depth (distance from the camera). Converting a 2D image into a 3D representation means estimating the depth for each pixel — essentially creating a 3D model or depth map from flat image data.

2D to 3D conversion is key in:

* Autonomous vehicles (for obstacle detection)
* AR/VR (placing virtual objects in 3D space)
* Robotics (navigation and interaction)
* 3D modeling (from photos)
* Medical imaging (3D reconstruction of organs)

**Program:**

from PIL import Image

import numpy as np

def shift\_image(img, depth\_img, shift\_amount=10):

# Ensure base image has alpha

img = img.convert("RGBA")

data = np.array(img)

# Ensure depth image is grayscale (for single value)

depth\_img = depth\_img.convert("L")

depth\_data = np.array(depth\_img)

deltas = ((depth\_data / 255.0) \* float(shift\_amount)).astype(int)

# This creates the transparent resulting image.

# For now, we're dealing with pixel data.

shifted\_data = np.zeros\_like(data)

height, width, \_ = data.shape

for y, row in enumerate(deltas):

for x, dx in enumerate(row):

if x + dx < width and x + dx >= 0:

shifted\_data[y, x + dx] = data[y, x]

# Convert the pixel data to an image.

shifted\_image = Image.fromarray(shifted\_data.astype(np.uint8))

return shifted\_image

img = Image.open("2D\_1.jpeg")

depth\_img = Image.open("2D\_2.jpeg")

shifted\_img = shift\_image(img, depth\_img, shift\_amount=10)

shifted\_img.show()

**Output:**

