Aim

To implement and demonstrate a real-time depth estimation system using the MiDaS (Monocular Depth Estimation) model on a local machine with an RTX 3060 GPU.

Objective

1. Utilize different versions of the MiDaS model to evaluate their performance in terms of accuracy and inference speed.
2. Implement a real-time depth estimation pipeline that captures video from a webcam, processes each frame to estimate depth, and displays the results.
3. Leverage GPU acceleration to optimize the depth estimation process.

Summary

This project implements a depth estimation system using the MiDaS model, which predicts depth from a single image. The system uses three variants of the MiDaS model (DPT_LARGE, DPT_Hybrid, MiDaS_small) and runs real-time depth estimation using a webcam. The project involves loading the MiDaS model, applying necessary transformations, predicting depth, and visualizing the results in real time.

Tools and Libraries Used

- **Python**: The programming language used for this project.
- **PyTorch**: For loading and running the MiDaS model.
- **OpenCV**: For video capture and image processing.
- **NumPy**: For numerical operations.
- **Matplotlib**: For potential plotting and visualization (though not used in the main code).

Procedure (Code Explanation)

1. **Import Libraries**: The necessary libraries (cv2, torch, matplotlib, enum, os, numpy) are imported.

CODE:

```
import cv2

import torch

import matplotlib.pyplot as plt

from enum import Enum

import os

import numpy as np
```

2. **Define Model Types**: An enumeration class ModelType is defined to specify different MiDaS models.

CODE:

```
class ModelType(Enum):

  DPT_LARGE = "DPT_Large"

  DPT_Hybrid = "DPT_Hybrid"

  MIDAS_SMALL = "MiDaS_small"
```

3. **Midas Class Initialization**: A Midas class is created to encapsulate model loading, transformation, and prediction functionalities.

   - **Constructor**: Initializes the MiDaS model based on the specified ModelType.
   - **useCUDA**: Checks for CUDA availability and sets the device to GPU or CPU.
   - **transform**: Loads the appropriate transformations based on the model type.
   - **predict**: Processes an input frame to estimate depth and returns the depth map.
   - **livePredict**: Captures video from the webcam, predicts depth for each frame, and displays the combined result.

CODE:

```
class Midas():

  def __init__(self, modelType: ModelType = ModelType.DPT_LARGE):

    self.midas = torch.hub.load("isl-org/MiDaS", modelType.value)

    self.modelType = modelType


  def useCUDA(self):

    if torch.cuda.is_available():

      print('Using CUDA')

      self.device = torch.device("cuda")

    else:

      print('Using CPU')

      self.device = torch.device("cpu")

    self.midas.to(self.device)

    self.midas.eval()
    def transform(self):
        print('Transform')
        midas_transforms = torch.hub.load("intel-isl/MiDaS", "transforms")
        if self.modelType.value == "DPT_Large" or self.modelType.value == "DPT_Hybrid":
            self.transform = midas_transforms.dpt_transform
        else:
            self.transform = midas_transforms.small_transform


    def predict(self, frame):
        img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        input_batch = self.transform(img).to(self.device)
        with torch.no_grad():
            prediction = self.midas(input_batch)
            prediction = torch.nn.functional.interpolate(
                prediction.unsqueeze(1),
                size=img.shape[:2],
                mode="bicubic",
                align_corners=False,
            ).squeeze()
        depthMap = prediction.cpu().numpy()
        depthMap = cv2.normalize(depthMap, None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX,
dtype=cv2.CV_8U)
        depthMap = cv2.applyColorMap(depthMap, cv2.COLORMAP_INFERNO)
        return depthMap


    def livePredict(self):
        print('Starting webcam (press q to quit)...')
        capObj = cv2.VideoCapture(0)
        while True:
            ret, frame = capObj.read()
            depthMap = self.predict(frame)
            combined = np.hstack((frame, depthMap))
            cv2.imshow('Combined', combined)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        capObj.release()
        cv2.destroyAllWindows()
```

4. **Run Function**: Instantiates the Midas class and runs the live prediction.

CODE:

```
def run(modelType: ModelType):
    midasObj = Midas(modelType)
    midasObj.useCUDA()
    midasObj.transform()
    midasObj.livePredict()


if __name__ == '__main__':
    run(ModelType.MIDAS_SMALL)
    # run(ModelType.DPT_LARGE)
```

Highlights

- **Model Versatility**: The code supports multiple MiDaS models, allowing a trade-off between accuracy and inference speed.
- **Real-Time Processing**: The implementation captures video in real-time and processes each frame for depth estimation.
- **GPU Acceleration**: The code checks for CUDA availability and utilizes the GPU to speed up the computations.
- **Visualization**: The depth map is colorized and displayed alongside the original frame for easy comparison.

Conclusion

This project successfully demonstrates a real-time depth estimation system using the MiDaS model. By supporting different versions of the MiDaS model, it provides flexibility in balancing accuracy and speed. The use of GPU acceleration ensures efficient processing, making it suitable for applications requiring real-time depth information.