Overview

CR-Vision is created by Carnot Research Pvt. Ltd.

Welcome to CR-vision, a comprehensive toolkit designed to streamline and simplify your everyday computer vision tasks. This powerful package encompasses a diverse array of modules and scripts, meticulously crafted to cater to a multitude of image processing challenges.

At its core, CR-vision capitalizes on the expansive ecosystem of Python's image processing, computer vision, and deep learning libraries, including heavyweight names such as OpenCV, ImageIO, Pillow, TensorFlow, Keras, Scikit-Image, and Scikit-Video. Leveraging the robust capabilities of these frameworks, CR-vision empowers users with unparalleled flexibility and efficiency in tackling a wide spectrum of visual data tasks.

Setting itself apart, CR-vision embraces the paradigm of Reactive Extensions for Python (RxPY), facilitating the construction of sophisticated, optimized, and stream-oriented computer vision pipelines. This approach proves invaluable, particularly in orchestrating complex, large-scale computer vision applications characterized by dynamic and interrelated subsystems.

Spanning an extensive range of functionalities, CR-vision offers unparalleled support across various domains, including:

- Basic Image Processing Operations: Lay the groundwork for your visual data manipulation with an arsenal of fundamental operations.
- Filtering and Effects: Elevate your images with a plethora of filters and effects, unlocking creative possibilities.
- Image Editing: Seamlessly edit and enhance images with intuitive tools and utilities.
- Image Restoration: Combat noise and imperfections with advanced denoising and super-resolution techniques.
- Face Detection: Harness the power of facial recognition for a myriad of applications, from security to entertainment.
- Template Matching: Effortlessly identifies patterns and structures within images, facilitating robust object detection.
- Pedestrian Detection: Ensure safety and efficiency in crowded environments with precise pedestrian detection capabilities.
- Image Classification: Enable intelligent decision-making with robust image classification algorithms.
- Traffic Monitoring: Gain insights and control in traffic management scenarios through advanced visual analytics.

With CR-vision, empower yourself to navigate the complexities of visual data effortlessly, unlocking new realms of possibility in the realm of computer vision.

Explore, innovate, and redefine what's possible with CR-vision—your indispensable companion in the realm of visual data processing.

FUNCTIONS-

- Image Processing**-**

  ex_emboss.py- This file imports necessary modules for image processing such as **os**, **cv2**, **cr.vision**, and **cr.vision.io**. It applies the emboss effect from **cr.vision.core.effects** module on the loaded puppy image in both RGB and BGR formats, displaying the original image along with the embossed versions using OpenCV's DisplayManager from **cr.vision.io**

- Basics-

  ex_translate.py-This file utilizes os, cv2, and modules from cr.vision for image processing. It demonstrates image translation using the translate function from cr.vision.geom, showcasing shifting and enlarging images. Finally, it displays the original, shifted, and enlarged images using OpenCV functions.

- Animation-

  ex_sliding_window. This file utilizes **cv2** for image processing, **numpy** for array manipulation, and **imageio** for creating the animation. It also employs functions such as **imread**, **rectangle**, **imshow**, **waitKey**, **bgr_to_rgb**, and **get_writer** for image manipulation, visualization, and animation creation.

- Contours-

  ex_draw_contour.py- This file imports **numpy** for array manipulation and **cv2** for image processing. It imports modules from **cr.vision** for further image processing functionalities. The **get_one_contour** function returns a predefined contour. Loads an image and draws the contour on it using the **drawContours** function from OpenCV. Finally, it displays the image with the drawn contour and waits for a key press to exit the window.

- Faces-

  ex_faces.py**-** This file utilizes cv2 for image processing and imports modules from cr.vision for image processing functionalities such as face detection. It loads an image of a girl and uses a face cascade detector from **cr.vision.faces.CascadeDetector()** to detect faces. Detected faces are then outlined with rectangles. Finally, both the original and modified images with detected faces are displayed using OpenCV's DisplayManager.

- Motion-

    ex_rx_motion_detection.py- This file sets up an application that detects **motion using a webcam feed**. It builds a pipeline consisting of various image processing steps, including **resizing, converting to grayscale, blurring, and motion detection**.

    The application runs indefinitely, subscribing to the pipeline, processing each frame from the webcam feed, displaying the frames with detected motion, and awaiting user input to quit the application.

- Object Detection-

    ex_nms.py- imports **numpy** for array manipulation and **cv2** for image processing. Additionally, it imports modules from **cr.vision** including **io** for displaying images and **bb** for non-maximum suppression (NMS) functionality. Utilizes functions such as **blank_image** to create blank images, **rectangle** and **nms** to perform non-maximum suppression on bounding boxes.

- Pedestrians-

    ex_detect.py- utilizes the Histogram of Oriented Gradients (HOG) method for pedestrian detection, employing functions such as **HOGDescriptor** and **detectMultiScale**. Additionally, it uses **non_maximum_suppression** from **cr.vision.object_detection** to suppress overlapping bounding boxes. Finally, it displays the original image and the image with detected people using OpenCV's DisplayManager from **cr.vision.io**.

- Traffic-

    ex_vehicle_counter.py- The **TrafficCounter** class from **cr.vision.traffic** is utilized to count vehicles in a traffic video. It captures frames from the video source, processes each **frame** to **draw vehicle information**, and writes the processed frames to a new video file using **imageio**. The process continues until an exception occurs or the user presses a key to exit.

- Video-ex_webcam.py- It imports the **WebcamSequence** class from **cr.vision** to capture frames from the webcam. It then displays each frame using OpenCV's **imshow** function and waits for the escape key to be pressed using **vision.wait_for_esc_key**.

Seamlessly blending cutting-edge technologies with intuitive functionalities, our library stands at the forefront of innovation in computer vision. Dive into a world of limitless creativity as you harness the power of our comprehensive image processing tools. From basic operations to intricate effects, CR-vision empowers you to effortlessly transform visual data with unmatched precision.

Delve deeper into dynamic animation and robust motion detection with CR-vision's captivating functionalities. Craft immersive visual experiences and gain real-time insights with functions provided. Navigate complex environments with confidence using our advanced object detection capabilities. From pedestrian detection to traffic monitoring, CR-vision provides the tools you need to identify, track, and analyze objects with unrivaled accuracy and efficiency. Step into the future of computer vision with CR-vision. Explore, innovate, and redefine what's possible in visual data processing—CR-vision is your gateway to a world of endless possibilities.

Requirements:

```
numpy==1.26.4
scipy==1.13.0
matplotlib==3.8.4
scikit-image==0.23.1
opencv-contrib-python==4.9.0.80
imageio==2.34.0
ipykernel==6.29.4
nbsphinx==0.9.3
click==8.1.7
sk-video==1.1.10
rx==3.2.0
sphinxcontrib-bibtex==2.6.2
```