OpenCV

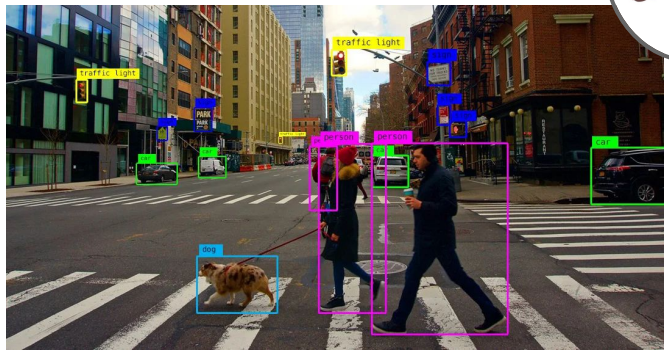
Conceptual Architecture

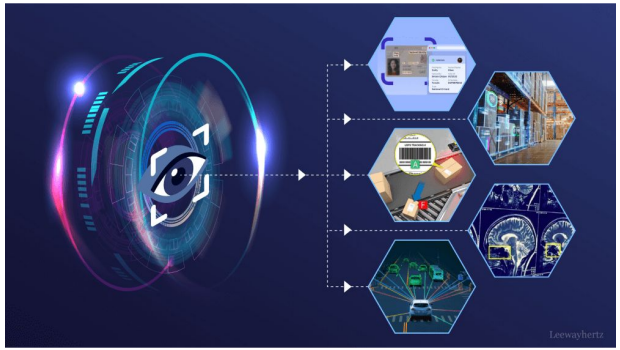
eLand

Eric Rodrigues, Laura Marin, Alp Baran Sirek, Negar Khalilazar, Danny Le

What Is Computer Vision?

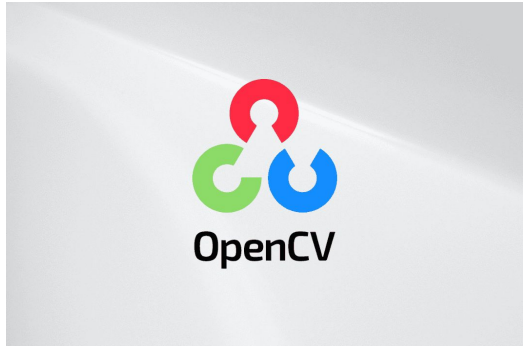
- Computer vision is a tool that enables computers to interpret visual information from the world
- We use it when cars track objects in real time, in augmented reality tools, etc
- Although the potential of computer vision is exciting, it also presents several challenges





Enter OpenCV

- OpenCV is one of the most widely used tools for addressing challenges of computer vision
- It is an open-source library designed for real-time image processing and computer vision applications
- The library, however, has expanded significantly now supporting additional languages and tools




```python
import numpy as np
import cv2
img = cv2.imread(r"C:\Users\ProjectPro\lotus.jpeg",1)
cv2.circle(img,(80,80), 55, (0,255,0), -1)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Today You Will Learn

- The Conceptual architecture of OpenCV
- How the system is organized and how its parts interact
- The illustration of the system’s structure, design and flow of control and data
- Division of Responsibilities and work flow
- Practical applications

What the System Does

Main Modules

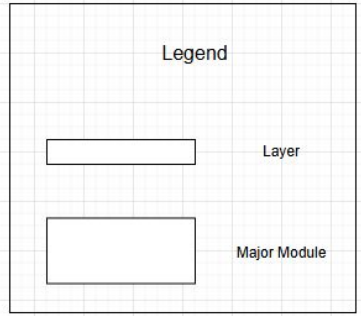
- Core functionality
- Image Processing
- Image File I/O
- Video I/O
- High-level GUI
- Video Analysis
- 3D Reconstruction
- 2D Features Framework
- Object Detection
- Deep Neural Network module
- Machine Learning
- G-API
- Computational Photography
- Images stitching

Architecture Style

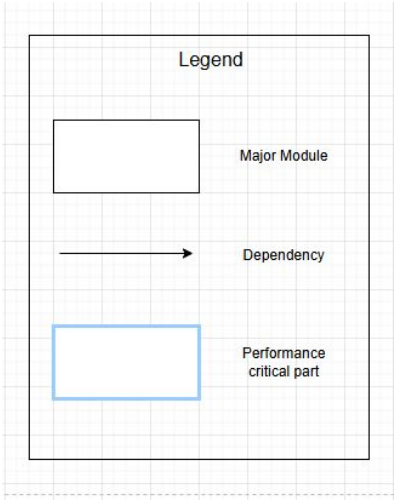
- Pipe and Filter?
- Repository?
- Modular and Layered

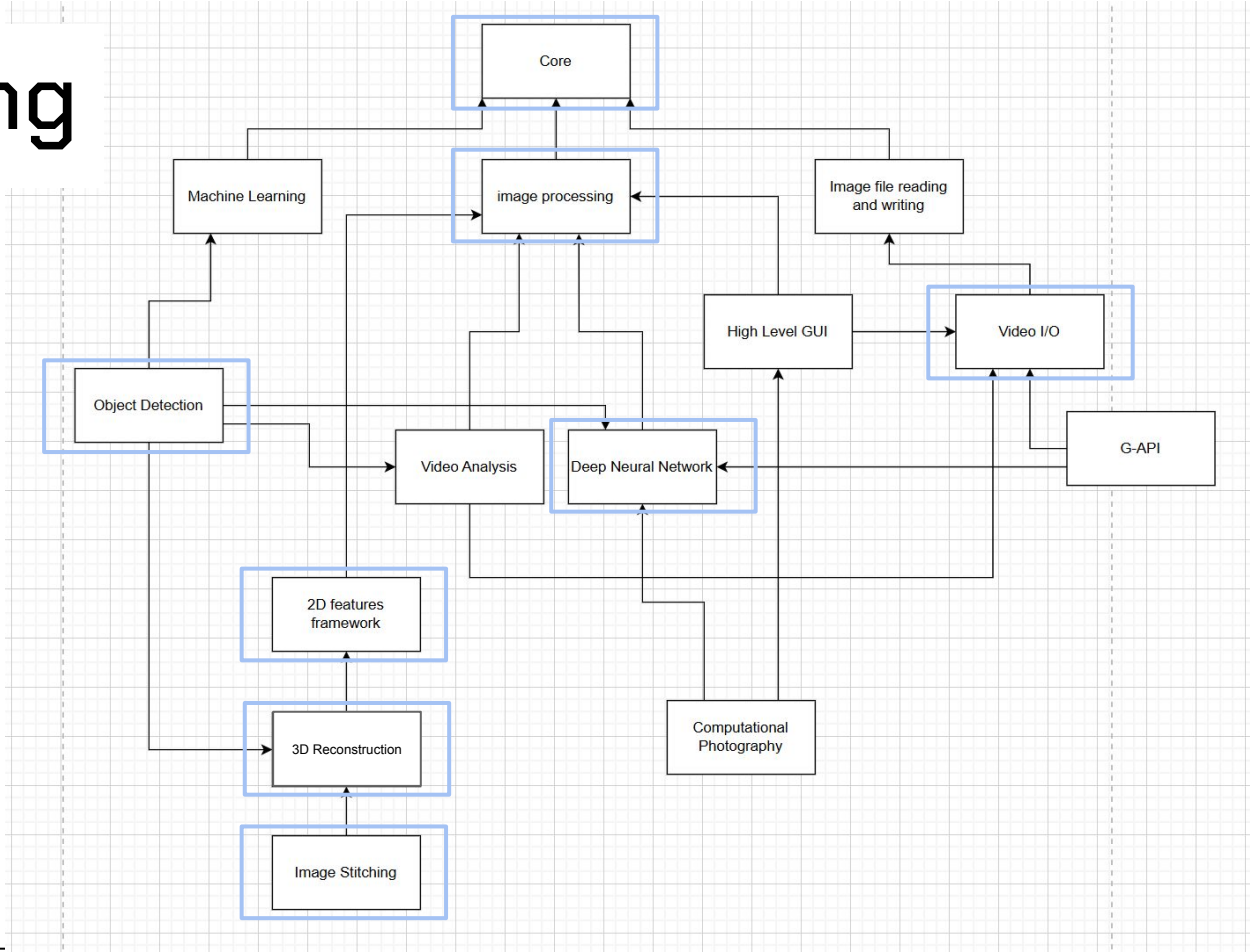
High-Level Architecture Diagram

Layer 4: Advanced Functionality				
Object Detection	Deep Neural Network	Machine Learning	Image Stitching	G-API
Layer 3: Analysis and Feature Extraction				
Video Analysis	2D Features Framework	3D Reconstruction		
Layer 2: Base Processing and I/O				
Image Processing	Video I/O	High level GUI	Computational Photography	
Layer 1: Core				
Core Functionality				



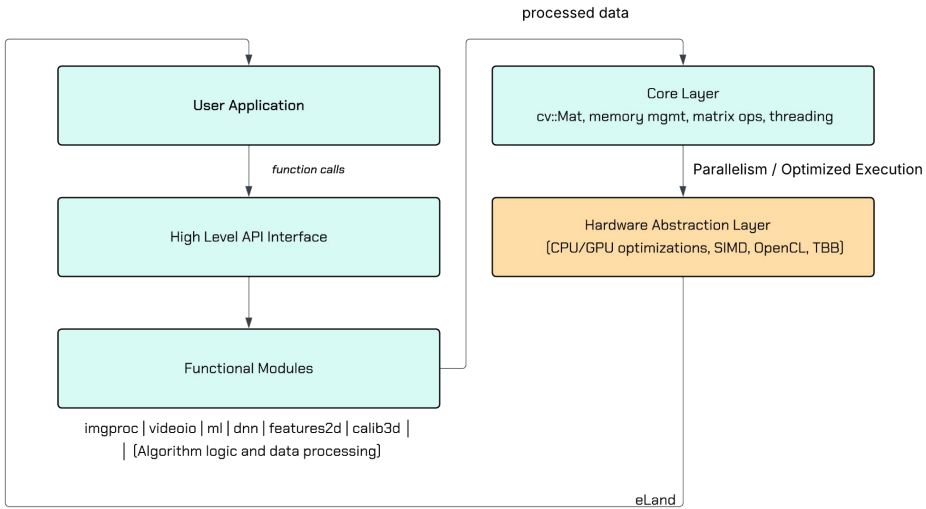
Interacting Parts





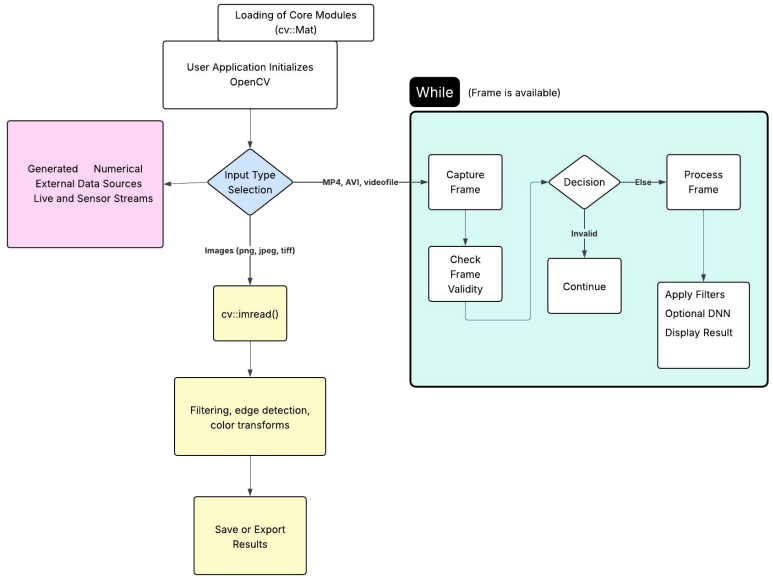
Control Flow

Conceptual-General Control Flow



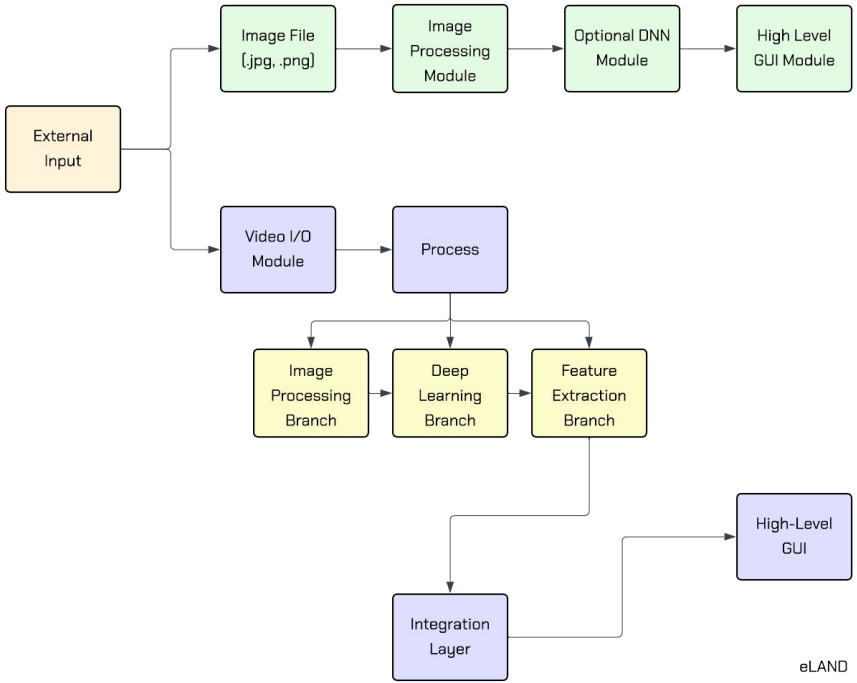
Processed data flows back upward to user

A More Specific Control Flow



Data Flow

Conceptual Data Flow Chart for Image and Video I/O



Control Flow (G-API)

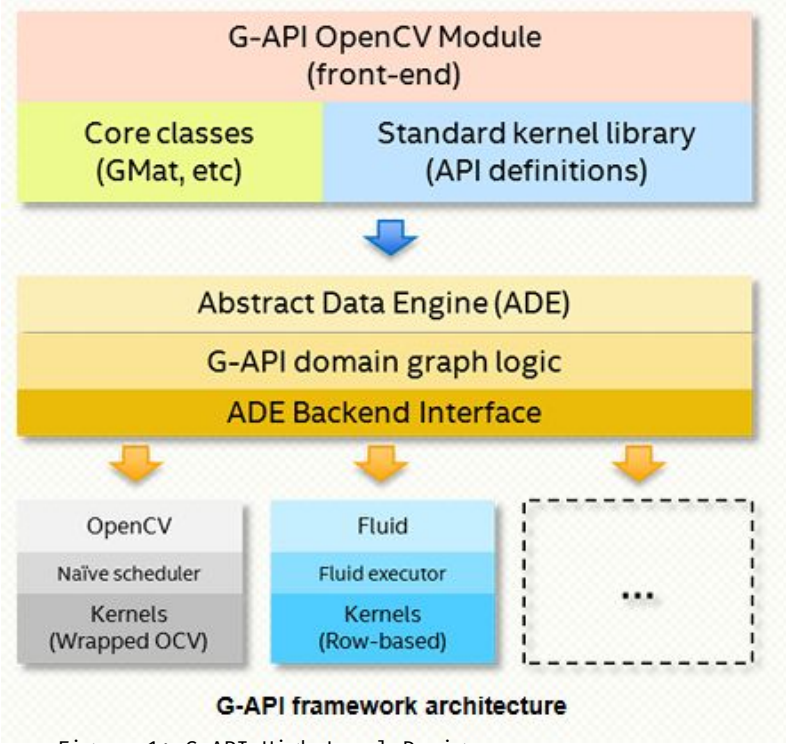


Figure 1: G-API High Level Design
https://www.ccoderun.ca/programming/doxygen/opencv/gapi_hld.html

G-API High-level design

- Defines processing as a graph of operations instead of step-by-step calls
- API Layer: user builds pipelines using GMat, GScalar, etc.
- Graph Compiler: optimizes and schedules execution
- Backends: choose best hardware automatically (CPU, GPU, OpenCL, etc.)
- Enables parallelism and zero-copy data flow between node

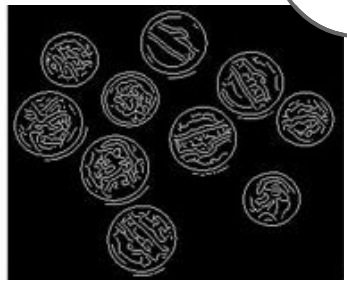
Concurrency

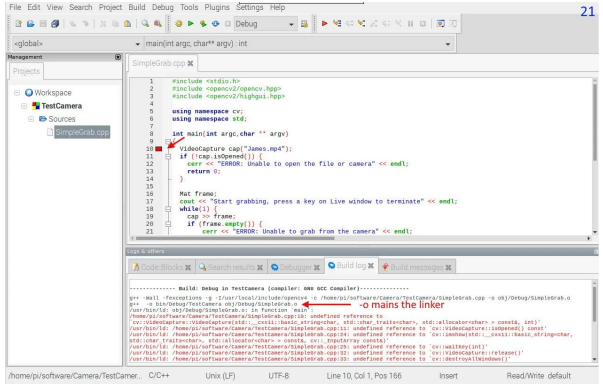
- Fully re-entrant: functions safe to call from multiple threads.
- Thread-safe data: cv::Mat uses atomic reference counting.
- Built-in parallelism: uses TBB / OpenMP internally.
- Custom threading: via cv::parallel_for_() and setNumThreads().
- Device-level concurrency: CPU + GPU tasks can run in parallel.

efficient, scalable real-time processing

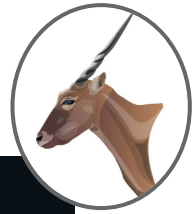
Initial OpenCV Release

- OpenCV’s early versions were primarily optimized for CPU-based image processing
- Capable of low level operations such as filtering, edge detection, and feature extraction
- Supported C and C++ ONLY



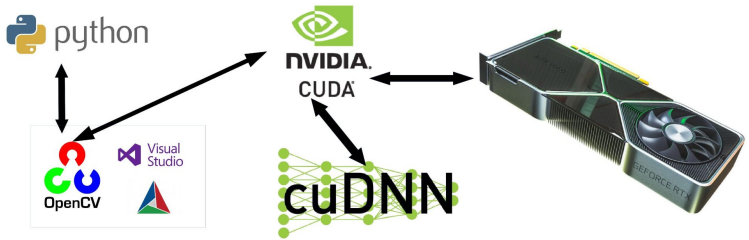


Performance Enhancements



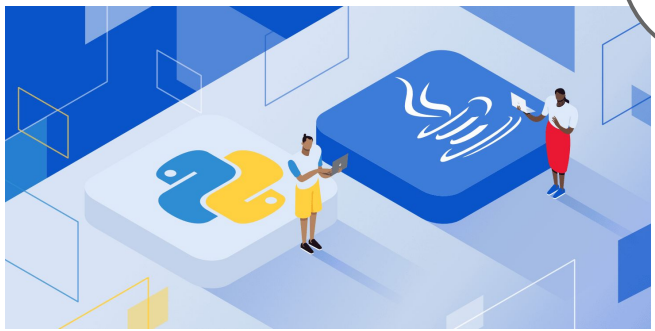
- Around version 2.0 the CUDA module was introduced to leverage NVIDIA GPUs for parallel computation

- OpenCV later integrated OpenCL support through the Transparent API (T-API), which allowed hardware-agnostic acceleration

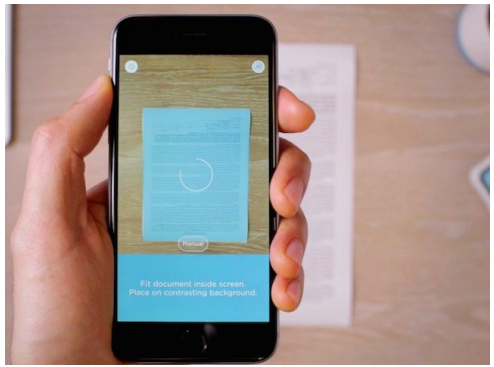


Expansion of Bindings

- OpenCV introduced bindings for additional languages such as Python, Java, and JavaScript

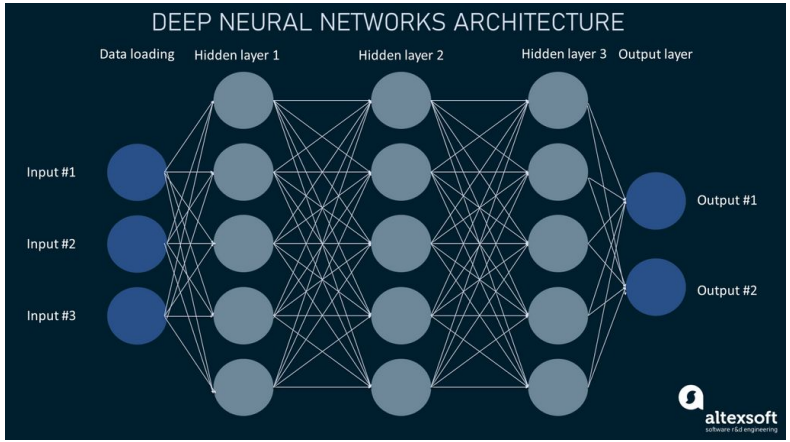


- Extended its reach to mobile platforms including Android and iOS

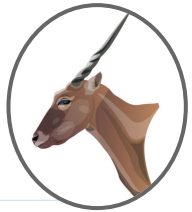


Deep Neural Network

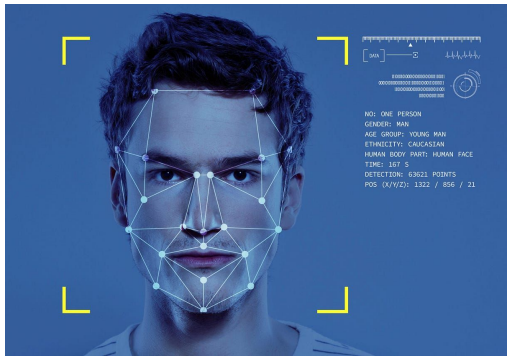
- OpenCV 3.3 (released in 2017), added the Deep Neural Network (DNN) module
- This enabled users to import and run pre-trained neural networks from frameworks
- This addition did not disrupt or modify the pre-existing modules preserving backward compatibility



Modern AI Applications

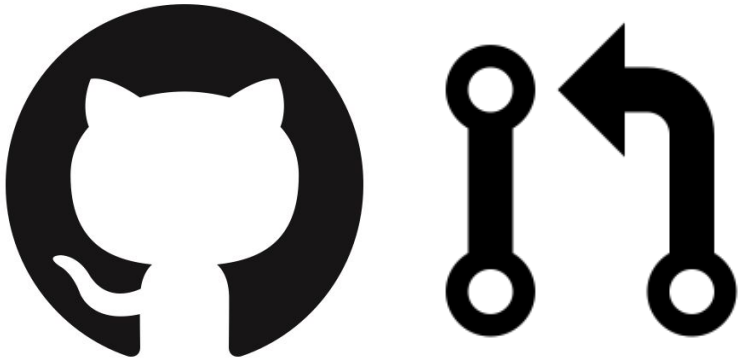


- Today, OpenCV plays a central role in the development of artificial intelligence systems
- Often used to prepare, augment, and transform data for training deep learning models
- Examples seen are autonomous driving and facial recognition



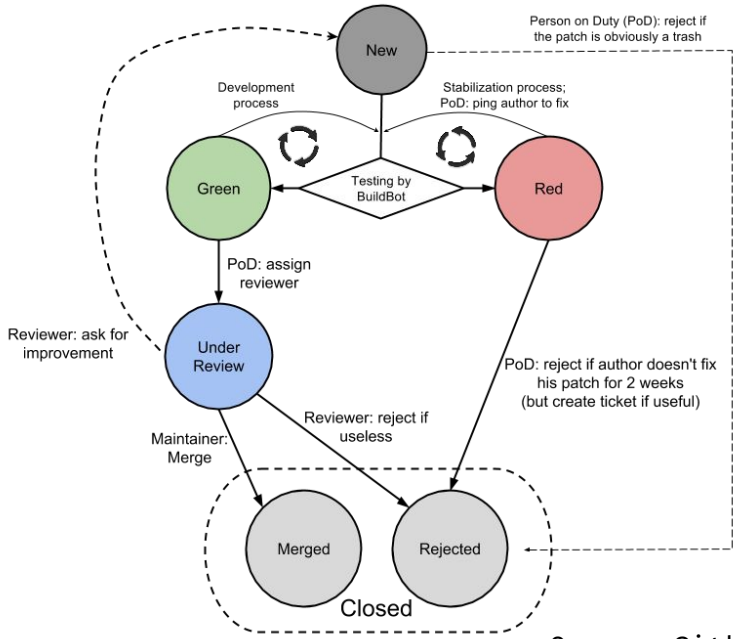
Division of Responsibilities

- Users → contributors → committers → release managers (→ technical group)
- Changes done via pull requests, review and test
- Decisions publicly recorded and released



How Work Flows

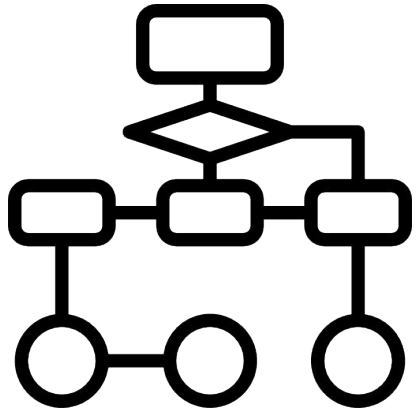
- Libraries are split by function
- QA: style guide, tests, cross-platform builds
- Decisions publicly recorded and released



*Opencv Github

Division of Responsibilities

- Each step in the pipeline has an owner
- Pull requests must define the API, add tests, document, show performance impact
- Branch policy: small fixes to current stable and big features next major

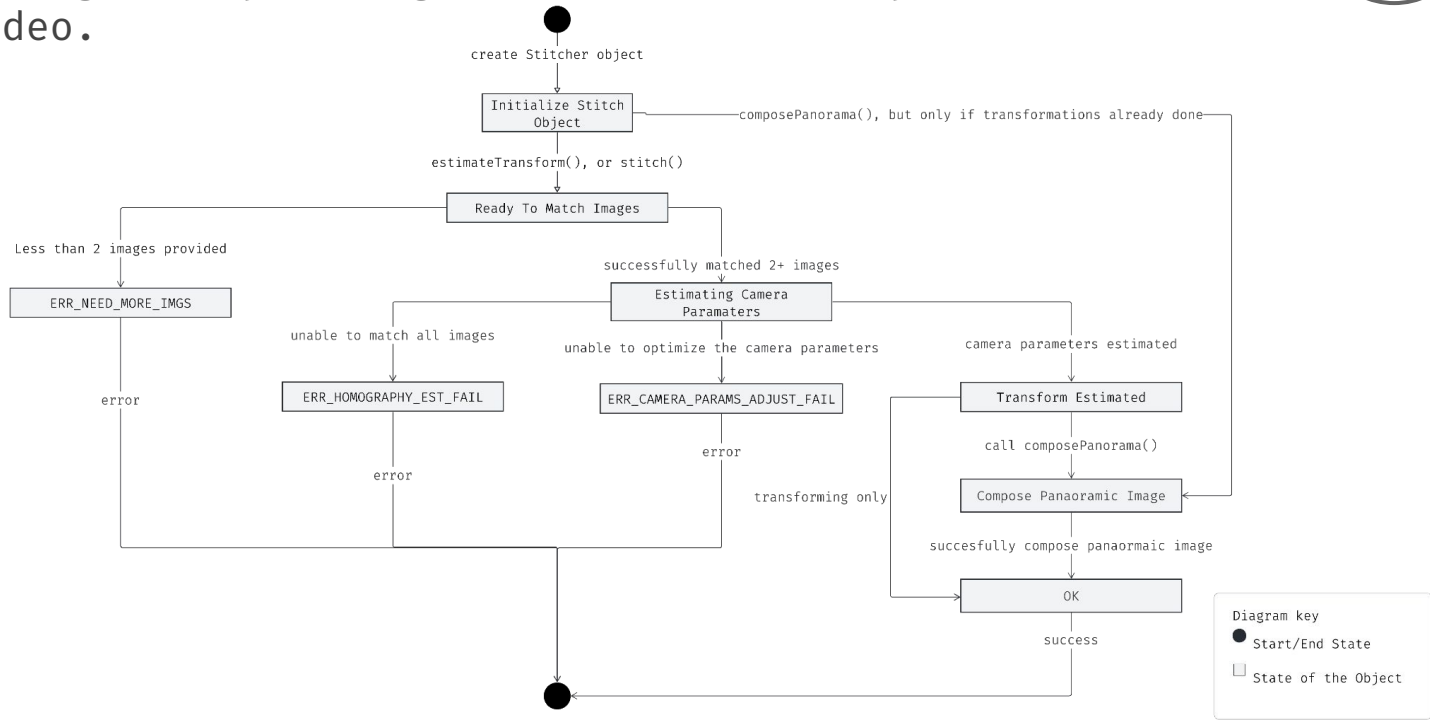


Practical Applications (1.0)

- OpenCV supports many situations where a computer needs to process and analyze visual information. One such case is the planetary rovers and drones.
  - Planetary rovers and drones require maximum situational awareness, as failure can be costly.
  - Multiple cameras require multiple video feeds to review, and can create overlapping fields of view.
  - A rotating camera cannot view all angles at once, and is extra complexity.

Practical Applications (1.1)

- Stitcher class (homography model): Transforming multiple images/videos into one panoramic image/video.



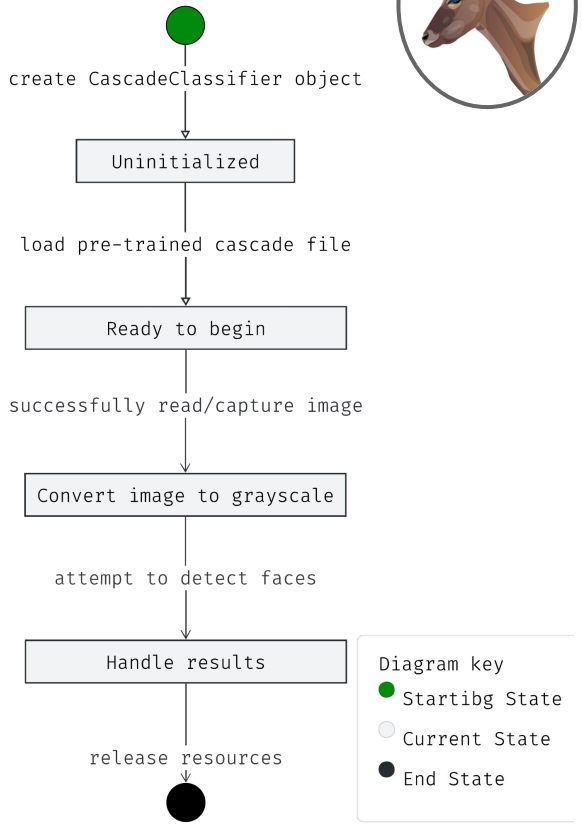
Practical Applications (1.2)

Practical Applications (2.0)

- Another case is facial recognition.
  - Easy authentication
  - Observing animals
  - Human–Robot interaction
  - Personal and Private security
  - Automated safety precautions
  - Forensic Analysis

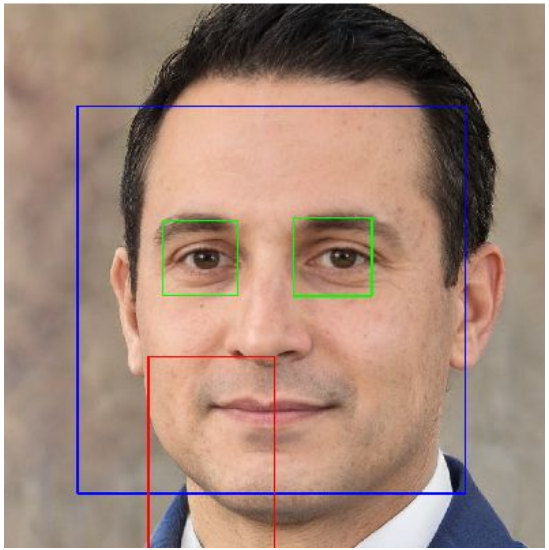
Practical Applications (2.1)

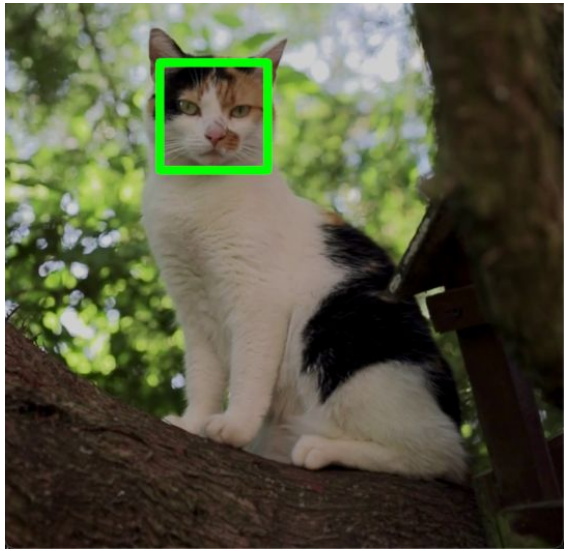
- CascadeClassifier class:
  scans images with multi-stage facial detection to efficiently reject non-facial regions early and focusing computational effort only on promising areas



Practical Applications (2.2)

- Capable of distinguishing people from animals





Lessons Learned

- Begin the analysis earlier
- Start Creating Diagrams earlier
- Better documentation of our thought process

Questions?