OpenCV Tutorial

- Popular library for Image Processing
- #include <opencv/opencv.hpp> to use all functionality available in OpenCV
- Namespace cv::

Data Types

- OpenCV uses its own types
- Names of types follow pattern CV_<bit_count><identifier><num_of_channels>

- **Example**: RGB image is `CV_8UC3`:
  8-bit unsigned char with 3 channels for RGB
- **Example**: Grayscale image is `CV_8UC1`:
  single 8-bit unsigned char for intensity
- Better to use `DataType`
- **Example**: `DataType<uint>::type == CV_8UC1`

Basic Matrix Type

- Every image is a `cv::Mat`, for "Matrix"
- `Mat image(rows, cols, DataType, Value);`
- `Mat_<T> image(rows, cols, Value);`
- Initialize with `zeros`:

```
1 cv::Mat image = cv::Mat::zeros(10, 10, CV_8UC3);
2 using Matf = cv::Mat_<float>;
3 Matf image_float = Matf::zeros(10, 10);
```

- Get type identifier with `image.type();`
- Get size with `image.rows`, `image.cols`
- **I/O**:
  - Read image with `imread`
  - Write image with `imwrite`
  - Show image with `imshow`
  - Detects I/O method from extension

cv::Mat is sort of shared pointer

It does not use `std::shared_ptr` but follows the same principle of reference counting

```
#include <opencv2/opencv.hpp>
#include <iostream>
int main() {
  using Matf = cv::Mat_<float>;
  Matf image = Matf::zeros(10, 10);
  Matf image_no_copy = image;  // Does not copy!
  image_no_copy.at<float>(5, 5) = 42.42f;
  std::cout << image.at<float>(5, 5) << std::endl;
  Matf image_copy = image.clone();  // Copies image.
  image_copy.at<float>(1, 1) = 42.42f;
  std::cout << image.at<float>(1, 1) << std::endl;
}
```

```
c++ -std=c++11 -o copy copy.cpp \
  `pkg-config --libs --cflags opencv`
```


imread

- Read image from file
- `Mat imread(const string& file, int mode=1)`
- Different modes:
  - unchanged: `cv::IMREAD_UNCHANGED < 0`
  - 1 channel: `cv::IMREAD_GREYSCALE == 0`
  - 3 channels: `cv::IMREAD_COLOR > 0`

```
#include <opencv2/imgcodecs.hpp>
#include <opencv2/opencv.hpp>
using namespace cv;
int main() {
  Mat i1 = imread("logo_opencv.png",
                  cv::IMREAD_GRAYSCALE);
  Mat_<uint8_t> i2 = imread("logo_opencv.png",
                            cv::IMREAD_GRAYSCALE);
  std::cout << (i1.type() == i2.type()) << std::endl;
  return 0;
}
```

The above slide has a type, line 4 in code snippet should be imwrite().

Writing float images to .exr files

- When storing floating point images OpenCV expects values to be in [0,1] range
- When storing arbitrary values, the values might be cut off
- Save to .exr files to avoid this
- These files will store and read values as is without losing precision

Float images I/O example

```cpp
#include <iostream>
#include <string>

#include <opencv2/opencv.hpp>
int main() {
  using Matf = cv::Mat_<float>;
  Matf image = Matf::zeros(10, 10);
  image.at<float>(5, 5) = 42.42f;
  std::string f = "test.exr";
  cv::imwrite(f, image);
  Matf copy = cv::imread(f, cv::IMREAD_UNCHANGED);
  std::cout << copy.at<float>(5, 5) << std::endl;
  return 0;
}
```

Hint: try what happens when using png images instead

8

imshow

- Display the image on screen
- Needs a window to display the image
- `void imshow(const string& window_name, const Mat& mat)`

```cpp
// clang-format off
#include <opencv2/opencv.hpp>
int main() {
  cv::Mat image = cv::imread("logo_opencv.png",
                             cv::IMREAD_COLOR);
  std::string window_name = "Window name";
  // Create a window.
  cv::namedWindow(window_name, cv::WINDOW_AUTOSIZE);
  cv::imshow(window_name, image);  // Show image.
  cv::waitKey();  // Don't close window instantly.
  return 0;
}
```

9

SIFT Descriptors

- SIFT: Scale Invariant Feature Transform
- Popular Features: illumination, rotation and translation invariant

SIFT Extraction With OpenCV

- `SiftFeatureDetector` to detect the keypoints
- `SiftDescriptorExtractor` to compute descriptors in keypoints

```
// Detect key points.
auto detector = SiftFeatureDetector::create();
vector<cv::KeyPoint> keypoints;
detector->detect(input, keypoints);

// Show the keypoints on the image.
Mat image_with_keypoints;
drawKeypoints(input, keypoints, image_with_keypoints);

// extract the SIFT descriptors
auto extractor = SiftDescriptorExtractor::create();
extractor->compute(input, keypoints, descriptors);
```


OpenCV 4 with CMake

- Install OpenCV 4 in the system see: https://gitlab.igg.uni-bonn.de/teaching/example_opencv
- Find using `find_package(OpenCV 4 REQUIRED)`

```
find_package(OpenCV 3 REQUIRED)
```

- Include `${OpenCV_INCLUDE_DIRS}`
- Link against `${OpenCV_LIBS}`

```
add_library(some_lib some_lib_file.cpp)
target_link_libraries(some_lib ${OpenCV_LIBS})
add_executable(some_program some_file.cpp)
target_link_libraries(some_program ${OpenCV_LIBS})
```