

Tutorial OpenCV

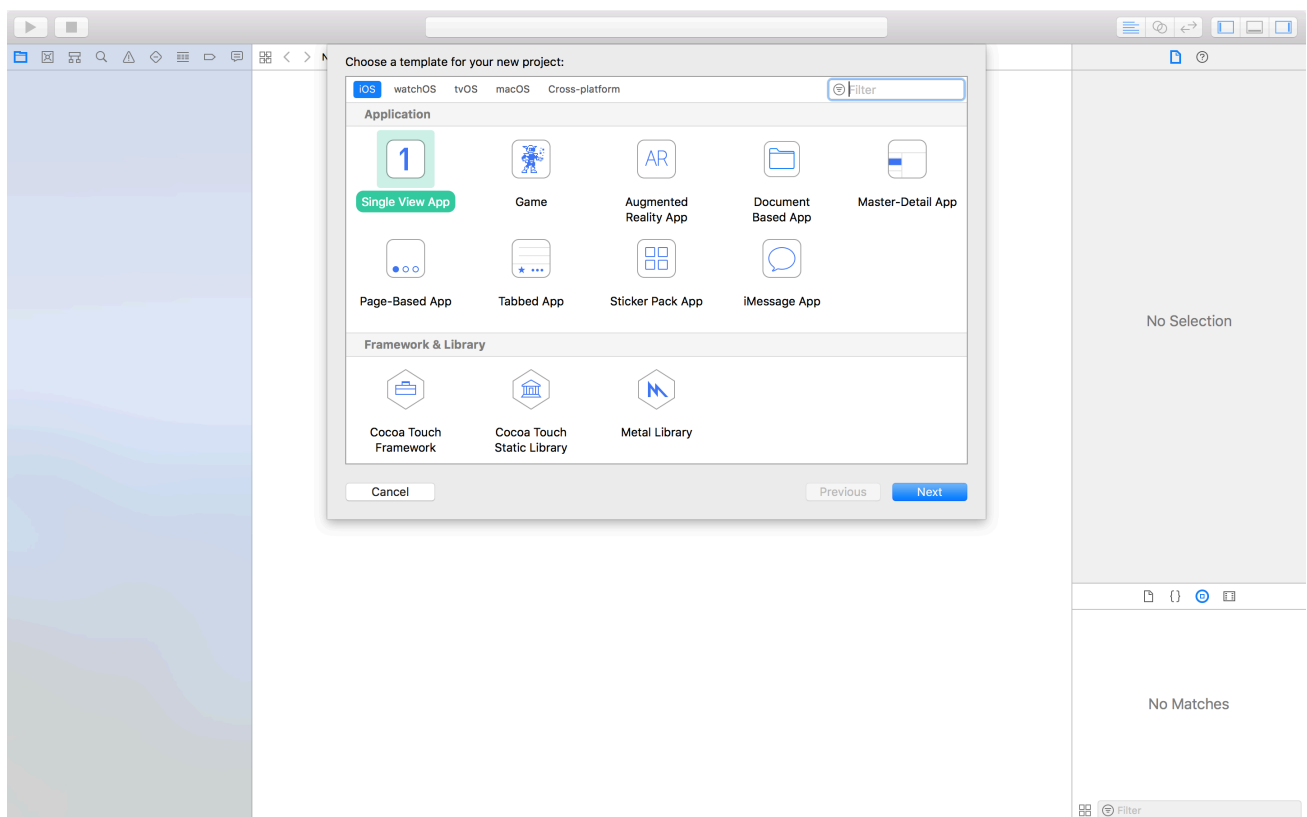
Como usar a lib do OpenCV em um projeto iOS

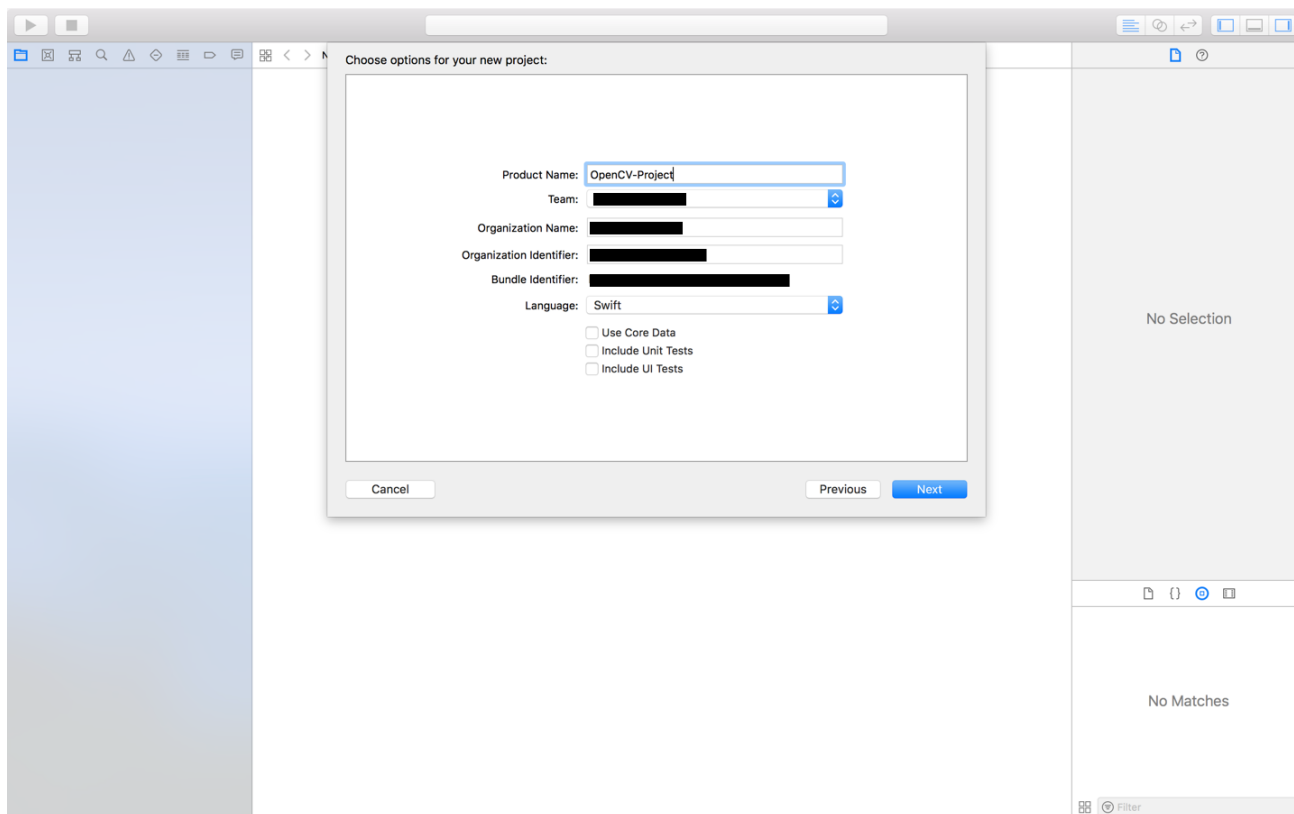
1. Introdução

Esse tutorial tem o objetivo de auxiliar na criação de uma aplicação iOS que possa utilizar métodos disponíveis na biblioteca de código aberto OpenCV, que é uma biblioteca de software de visão computacional e machine learning, muito utilizado para processamento de imagens e afins, e que é escrita em C/C++ otimizado. [1]

2. Tutorial

O primeiro a se fazer é criar um novo projeto no Xcode escolhendo a opção “Single View Application”.





A próxima etapa é a partir do uso do CocoaPods, um gerenciador de dependências para projetos em Swift e Objective-C [3], inserir a dependência do framework OpenCV em nosso novo projeto. Para isso, execute o comando abaixo na linha de comando, que será responsável por criar um arquivo Podfile.

```
pod init
```

Em seguida, abra esse novo arquivo com o nome Podfile gerado e ele irá disponibilizar o seguinte conteúdo.

```
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'OpenCV-Project' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for OpenCV-Project
end
```

Modifique esse arquivo para que ele fique da seguinte maneira, o que implica na inserção da dependência do framework OpenCV.

```
target 'OpenCV-Project' do
  use_frameworks!

  # Pods for OpenCV-Project

  # Pod for OpenCV
  pod 'OpenCV', '3.1.0.1'

end
```

Em sequência, feche o projeto criado no Xcode, salve e feche esse arquivo Podfile, e execute o seguinte comando na linha de comando. Essa execução fará com que a dependência seja instalada em seu projeto.

```
pod install
```

Basta então aguardar para que essa instalação seja finalizada. A seguinte mensagem aparecerá na saída da linha de comando após essa tarefa ser concluída.

```
Analyzing dependencies
Downloading dependencies
Using OpenCV (3.1.0.1)
Generating Pods project
Integrating client project
Sending stats
Pod installation complete! There is 1 dependency from the Podfile and 1 total pod installed.
```

Com isso, você irá perceber que o CocoaPods criou um arquivo chamado 'OpenCV-Project.xcworkspace' e uma pasta com o nome de 'Pods' no qual todas as dependências do projeto estarão armazenadas. Nesse caso, a única dependência existente será a do OpenCV.

O próximo passo então é abrir o projeto, porém agora isso será feito através do arquivo 'OpenCV-Project.xcworkspace' e não mais através do arquivo 'OpenCV-Project.xcodeproj'. Tendo feito isso, crie um novo arquivo Cocoa Touch Class, salve esse arquivo no diretório do projeto e também selecione a opção para criar um Bridging Header.


Choose a template for your new file:


ios


watchOSTVOSmacOS


Filter


Source



Cocoa Touch Class



UI Test Case Class



Unit Test Case Class



Swift File


Objective-C File



Header File



C File



C++ File



Metal File

User Interface


Storyboard


View


Empty


Launch Screen


Cancel

Previous


Next

Choose options for your new file:

Class:

Subclass of: 

☐ Also create XIB file

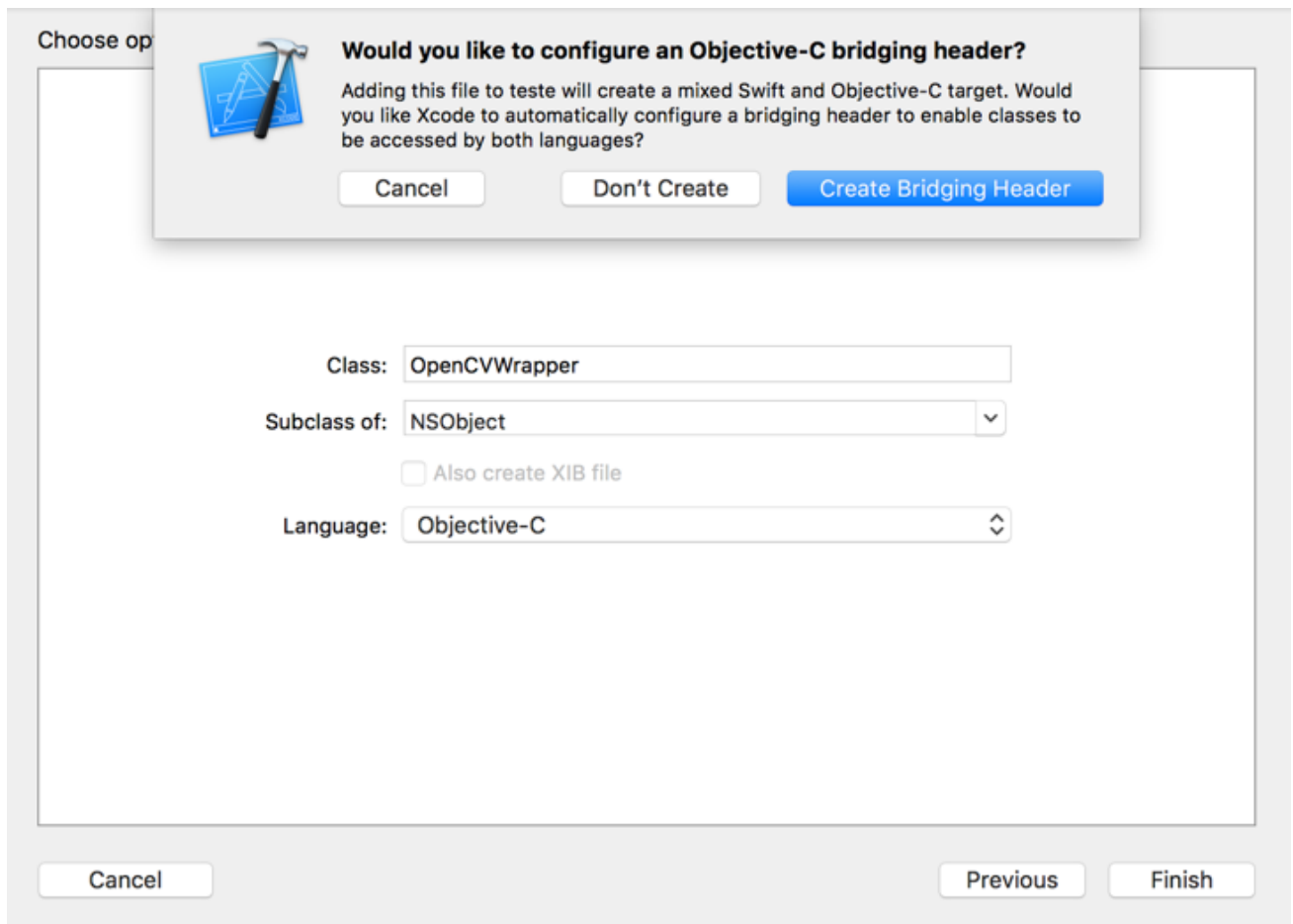
Language: 

Cancel

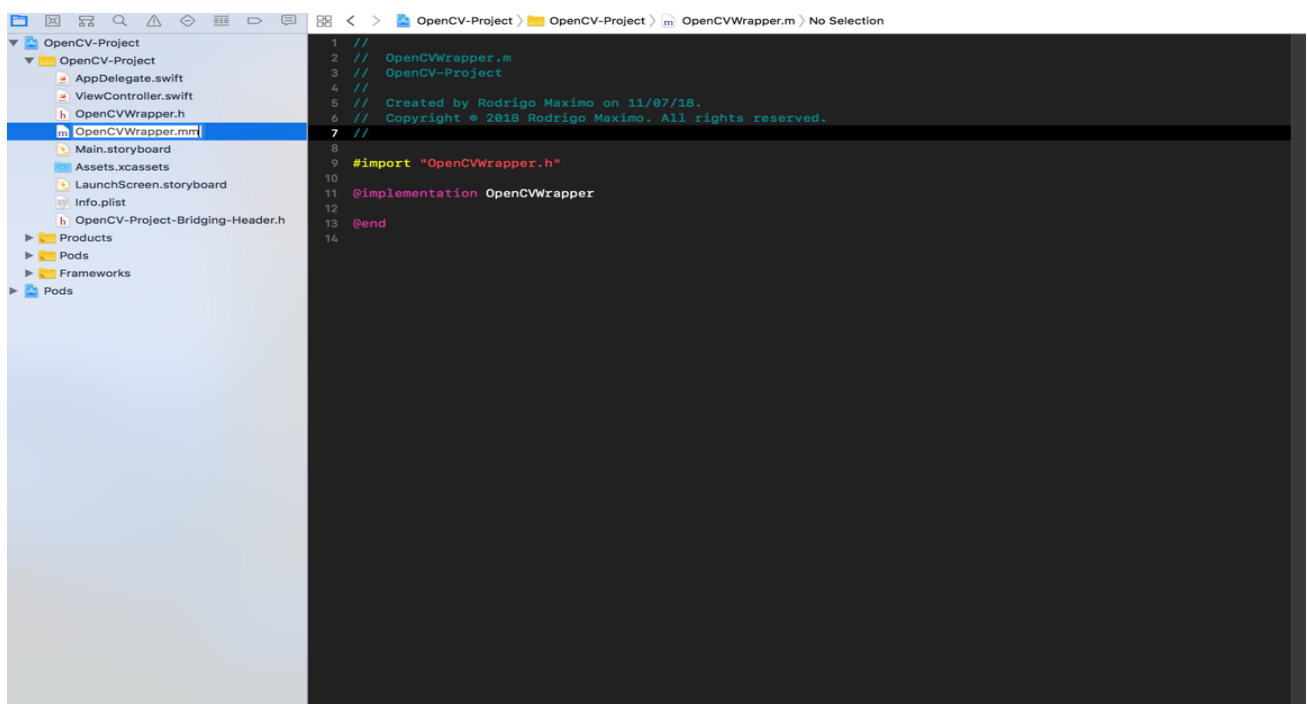
Previous

Next

4



Agora é preciso transformar esse novo arquivo criado, cuja extensão é “.m” em um arquivo de extensão “.mm”, que será reconhecido pelo Xcode como um arquivo Objective-C++, ou seja, um arquivo que compila código em Objective-C junto de código escrito em C++.



Nesse arquivo, para ter acesso aos métodos da biblioteca será preciso importá-la. Para isso, basta escrever '#import <opencv2/opencv.hpp>' no topo do arquivo 'OpenCVWrapper.mm'.

```
#import "OpenCVWrapper.h"
#import <opencv2/opencv.hpp>

using namespace cv;

@implementation OpenCVWrapper

@end
```

Além disso, o namespace inserido para o opencv (cv) irá facilitar a escrita, leitura e organização do código.

Finalmente, vamos a utilização dos métodos do OpenCV, e para demonstrar o funcionamento de todos esses passos, vamos implementar um método que dada uma imagem, retorna o negativo da mesma e também outro método que transforma uma imagem em sua escala de cinza.

Como vamos trabalhar com uma estrutura de imagem do tipo UIImage, teremos que importar também outra pasta do OpenCV, através da linha '#import <opencv2/imgcodecs/ios.h>', ela nos permitirá a conversão da estrutura UIImage para a estrutura Mat, estrutura essa que representa uma matriz, como o próprio nome sugere. Essa é a estrutura utilizada dentro do OpenCV para representar as imagens.

```
#import "OpenCVWrapper.h"
#import <opencv2/opencv.hpp>
#import <opencv2/imgcodecs/ios.h>

using namespace cv;

@implementation OpenCVWrapper

@end
```

Além disso, também iremos importar a biblioteca do UIKit dentro do arquivo 'OpenCVWrapper.h', para que seja possível a utilização e referências à estrutura UIImage. Ainda, dentro desse mesmo arquivo, vamos escrever a assinatura dos dois métodos que iremos implementar no arquivo de extensão '.mm', com o auxílio do OpenCV.

```

#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

@interface OpenCVWrapper : NSObject

-(UIImage *)grayScaleOfImage: (UIImage *) image;
-(UIImage *)negativeOfImage: (UIImage *) image;

@end

```

O primeiro método transforma uma imagem em escala de cinza, e o segundo calcula o negativo de uma imagem, caso ela esteja em escala de cinza. As suas implementações estão presentes no arquivo ‘OpenCVWrapper.mm’ e seguem abaixo.

```

-(UIImage *)negativeOfImage: (UIImage *) image {
    Mat matImage;

    // converting the uiimage to mat
    UIImageToMat(image, matImage);

    // just perform the negative if the image is in grayscale
    if (matImage.elemSize() == 1) {
        matImage = 255 - matImage;
    }

    // converting the final mat to an uiimage again
    UIImage *finalImage = MatToUIImage(matImage);

    return finalImage;
}

-(UIImage *)grayScaleOfImage: (UIImage *) image {
    Mat matImage;

    // converting the image to mat
    UIImageToMat(image, matImage);

    // just try to convert if the image is not in gray scale
    if (matImage.elemSize() != 1) {
        cvtColor(matImage, matImage, COLOR_BGR2GRAY);
    }

    // converting the final mat to an uiimage again
    UIImage *finalImage = MatToUIImage(matImage);

    return finalImage;
}

```

Agora, basta utilizá-los em uma classe em Swift. No entanto, para isso ser possível ainda é preciso fazer uma última coisa. Insira a seguinte linha naquele arquivo Bridging Header que foi criado no momento em que a Cocoa Touch Class foi criado.

```
#import "OpenCVWrapper.h"
```

Essa linha fará com que todos os métodos presentes nesse arquivo ‘OpenCVWrapper.h’, que são escritos em Objective-C, estejam visíveis para arquivos Swift. Com isso, os dois métodos implementados poderão ser chamados em uma classe Swift.

3. Referências

[1] OpenCV

Disponível em: <<https://opencv.org>>

[2] iOS — OpenCV and Swift

Disponível em: <<https://medium.com/@borisohayon/ios-opencv-and-swift-1ee3e3a5735b>>

[3] CocoaPods

Disponível em: <<https://cocoapods.org>>