bioinspired模块

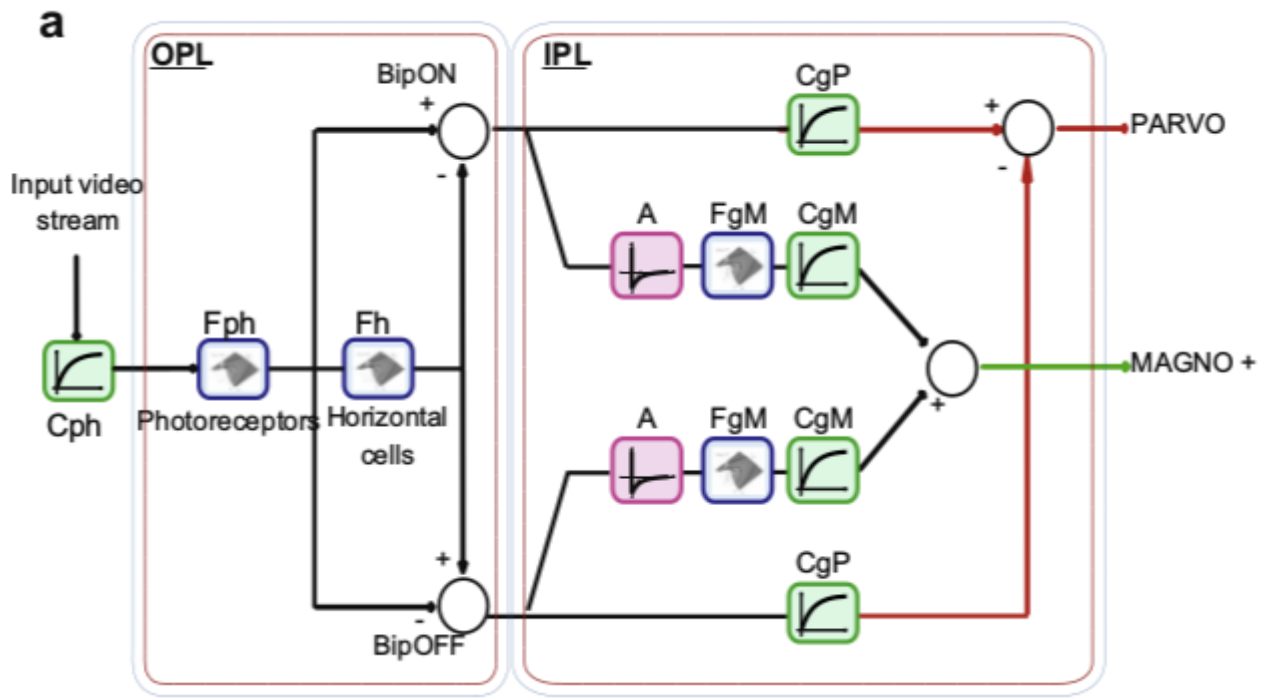
Retina

模拟人眼视觉细胞，算法中将人眼视觉细胞分为大细胞和小细胞；大细胞获取运动信息，小细胞获取细节信息。

> the Parvocellular channel (Parvo) dedicated to detail extraction and the Magnocellular channel (Magno) dedicated to motion information extraction.

算法介绍

模型



预处理

预处理模仿Michaelis–Menten（米氏方程)的方法，调整输入图像的亮度信息。

$$C(p) = \frac{R(p)}{R(p) + R_0(p)} \cdot V_{max} + R_0(p) \tag{1}$$

$$R_0(p) = V_0 \cdot L(p) + V_{max} \cdot (1 - V_0) \tag{2}$$

$C(p)$代表预处理后的输出图像，$R(p)$是当前图像的亮度信息，V_{max}代表输出像素最大值，$L(p)$代表局部亮度信息，V_0是可调参数。

$C(p)$经过F_{ph}和F_h两个传递函数。

$$F_{ph} = \frac{1}{1 + \beta_{ph} + 2\alpha_{ph} \cdot (1 - \cos(2\pi f_s)) + 2j\pi\tau_{ph} f_t} \tag{3}$$

$$F_h = \frac{1}{1 + \beta_h + 2\alpha_h \cdot (1 - \cos(2\pi f_s)) + 2j\pi\tau_h f_t} \tag{4}$$

其中，f_s代表空域频率，f_t代表时域频率。

API

C++

```
//实例化Retina对象
//参数：
//  inputSize:输入buffer的大小
static Ptr<Retina> cv::bioinspired::Retina::create(Size inputSize);

//实例化Retina对象，与上一个方法参数列表不同
//参数：
//  inputSize:输入buffer大小
//  colorMode:true表示使用颜色处理，false表示不使用颜色处理
//  colorSamplingMethod:颜色采样方法，可选参数为：
//                      cv::bioinspired::RETINA_COLOR_RANDOM,每个像素随机使用R、G、B通道
//                      cv::bioinspired::RETINA_COLOR_DIAGONAL，
//                      cv::bioinspired::RETINA_COLOR_BAYER，标准bayer采样
//  useRetinaLogSampling:true表示使用log采样，可继续设置下两个参数；false表示不使用log采样
//  reductionFactor:输出图像衰减系数
//  samplingStrenght:log采样规模的强度
static Ptr<Retina> cv::bioinspired::Retina::create(
    Size inputSize,
    const bool colorMode,
    int colorSamplingMethod = RETINA_COLOR_BAYER,
    const bool  useRetinaLogSampling = false,
    const float reductionFactor = 1.0f,
    const float samplingStrenght = 10.0f
);

//使用parvocellular（小细胞）通道
//默认参数为true，表示激活该通道；false表示不激活该通道
virtual void activateContoursProcessing(const bool activate)=0;

//使用mangocellular（大细胞）通道
//默认参数为true，表示激活该通道；false表示不激活该通道
virtual void activateMovingContoursProcessing(const bool activate)=0;

//纠正图像的光照和背光问题，增强阴影处的细节信息
//参数：
//  inputImage:输入图像，格式可以为：CV_32F, CV_32FC1, CV_32F_C3, CV_32F_C4，第四通道不会被考虑
//  outputToneMappedImage:输出图像，格式为CV_8U, CV_8UC3
virtual void applyFastToneMapping(InputArray inputImage, OutputArray
outputToneMappedImage)=0;

//清除缓存，等效于初始化：长时间闭眼后睁眼
virtual void clearBuffers()=0;
```

```
//获取magno图像（大细胞图像）
virtual void getMagno(OutputArray retinaOutput_magno)=0;

//获取parvo图像（小细胞图像）
virtual void getParvo(OutputArray retinaOutput_parvo)=0;

//运行算法
virtual void run(InputArray inputImage)=0;

//设置图像饱和度信息，对每个通道使用sigmod函数
//参数：
//  saturateColors:默认为true激活，false表示不激活
//  colorSaturationValue:饱和度值
virtual void setColorSaturation(const bool saturateColors=true, const float
colorSaturationValue=4.0f)=0;

//读取算法的参数文件，文件格式为xml
//参数：
//  retinaParameterFile:参数文件路径
//  applyDefaultSetupOnFailure:默认为true表示文件读取错误时抛出错误，false表示不抛出错误
virtual void setup(String retinaParameterFile="", const bool
applyDefaultSetupOnFailure=true)=0;

//与上一个函数相同，第一个参数类型不同
virtual void setup(cv::FileStorage &fs, const bool applyDefaultSetupOnFailure=true)=0;

//设置参数
virtual void setup(RetinaParameters newParameters)=0;


virtual void setupIPLMagnoChannel(
    const bool normaliseOutput=true,
    const float parasolCells_beta=0.f,
    const float parasolCells_tau=0.f,
    const float parasolCells_k=7.f,
    const float amacrinCellsTemporalCutFrequency=1.2f,
    const float V0CompressionParameter=0.95f,
    const float localAdaptintegration_tau=0.f,
    const float localAdaptintegration_k=7.f
)=0;

virtual void setupOPLandIPLParvoChannel(
    const bool colorMode=true,
    const bool normaliseOutput=true,
    const float photoreceptorsLocalAdaptationSensitivity=0.7f,
    const float photoreceptorsTemporalConstant=0.5f,
    const float photoreceptorsSpatialConstant=0.53f,
    const float horizontalCellsGain=0.f,
    const float HcellsTemporalConstant=1.f,
    const float HcellsSpatialConstant=7.f,
    const float ganglionCellsSensitivity=0.7f
)=0;
```

```
//写入算法参数到文件
virtual void write(String fs)const=0;

//写入算法参数到文件，与上一个函数效果相同，参数类型不同
virtual void write(FileStorage &fs) const CV_OVERRIDE=0;

//获取原始的parvo图像（不归一化为0-255）
virtual void getParvoRAW(OutputArray retinaOutput_parvo)=0;

//获取原始的parvo图像（不归一化为0-255），与上一个函数效果相同，参数和返回值类型不同
virtual const Mat getParvoRAW() const =0

//输出使用的参数，字符创类型
virtual const String printSetup()=0;

//获取原始的magno图像（不归一化为0-255）
virtual void getMagnoRAW(OutputArray retinaOutput_magno)=0;

//获取原始的magno图像（不归一化为0-255），与上一个函数效果相同，参数和返回值类型不同
virtual const Mat getMagnoRAW() const =0;

//获取算法输出buffer大小，如果之前使用了log域转换，则输出和输入buffer大小可能不同
virtual Size getOutputSize()=0;

//获取算法当前的参数
virtual RetinaParameters getParameters()=0;

//获取输入buffer大小
virtual Size getInputSize()=0;
```