MedCV

发布 *1.0.0*

Zhang Hongyuan

2021 年 04 月 26 日

Contents

1 MedCV: 基于 Python 的医学可视化库 **1**
- 1.1 这就是 MedCV . . . 1
- 1.2 设计原则 . . . 1
- 1.3 当前的版本与更新 . . . 2
- 1.4 安装 . . . 2
- 1.5 技术支持 . . . 3

2 可视化接口 **5**
- 2.1 图像分类 . . . 5
- 2.2 目标检测 . . . 7
- 2.3 实例分割 . . . 9

3 工具类函数 **15**
- 3.1 图像转换 . . . 15

4 PyQt5 控件 **21**
- 4.1 QMedManger . . . 21
- 4.2 QMedLabel . . . 22

5 Utils 基础类 **25**
- 5.1 LUT . . . 25
- 5.2 参数校验 . . . 26

6 更新日志 **27**

7 常见问题 **29**

8 附录 **31**

9 Indices and tables **33**

CHAPTER 1

MedCV: 基于 Python 的医学可视化库

1.1 这就是 MedCV

MedCV 是一个高级医学图像可视化库，MedCV 由纯 Python 编写而并基于Numpy 、OpenCV 完成图像的可视化，提供PyQt5 可复用的控件。MedCV 为解决 OpenCV 对医学图像兼容而生，能够把你的结果迅速可视化，如果你有如下需求，请选择 MedCV：

- 简易和快速的医学可视化设计（MedCV 具有高度模块函数接口，极简，和可扩充特性）
- 建立医学图像交互平台

MedCV 兼容的 Python 版本：Python3+

1.2 设计原则

- **用户友好**：MedCV 是为人类而不是外星人设计的 API。用户的使用体验始终是我们考虑的首要和中心内容。MedCV 遵循减少认知困难的最佳实践：MedCV 提供一致而简洁的 API，快速地完成可视化任务。
- **模块性**：具体而言，MedCV 有可视化、工具类、GUI 控件三个独立的模块，你可以使用它们来定制自己的可视化操作或者搭建交互平台。
- **与 Python 协作**：MedCV 没有单独的模型配置文件类型，模型由 Python 代码描述，使其更紧凑和更易 debug，并提供了扩展的便利性。

1.3 当前的版本与更新

本文档是 MedCV 的中文文档，包括 MedCV 的全部内容，以及更多的例子、解释和建议

由于作者水平和研究方向所限，无法对医学图像可视化操作都非常精通，因此 MedCV 代码和文档中不可避免的会出现各种错误、疏漏和不足之处。如果您在使用过程中有任何意见、建议和疑问，欢迎发送邮件到 moyan_work@foxmail.com 与我取得联系。

您对文档的任何贡献，包括文档的翻译、查缺补漏、概念解释、发现和修改问题、贡献示例程序等，均会被记录在致谢，十分感谢您对 MedCV 的贡献！

1.4 安装

在安装 MedCV 之前，请安装以下依赖：

- Numpy（用于图像矩阵的运算）
- OpenCV（兼容调用完成可视化）
- PyQt5（定制医学图像版本的设计控件）

然后你就可以安装 MedCV 本身了。用两种方法安装 MedCV：

- **使用 PyPI 安装 MedCV（推荐）**

```
sudo pip install medcv
```

如果你使用 virtualenv 虚拟环境，你可以避免使用 sudo：

```
pip install medcv
```

- **或者：使用 Github 源码安装 MedCV**

首先，使用 `git` 来克隆 MedCV：

```
git clone https://github.com/szuboy/medcv.git
```

然后，`cd` 到 MedCV 目录并且运行安装命令：

```
cd medcv
sudo python setup.py install
```

1.5 技术支持

你可以在下列网址提问或加入 MedCV 开发讨论：

- MedCV Google group
- 你也可以在 Github issues 里提问或请求新特性（在提问之前请确保你阅读过我们的指导，我本人会经常为大家解答）

CHAPTER 2

可视化接口

2.1 图像分类

2.1.1 image_with_heatmap

```
medcv.visualize.cls.image_with_heatmap(image, heatmap, alpha=0.8, beta=0.3,␣
↪colormap=cv2.COLORMAP_JET, width=None, level=None)
```

将关注热图在原图上进行可视化，例如：深度学习分类网络的 Grad-CAM 热图叠加在医学图像上进行可视化。

本函数内部采用 `cv2.addWdighted` 函数实现，是一种图像融合方法，对图像赋予不同的权重，以使其具有融合或透明的效果。根据以下等式进行融合：

$$dst = \alpha \cdot image + \beta \cdot heatmap + \gamma$$

参数

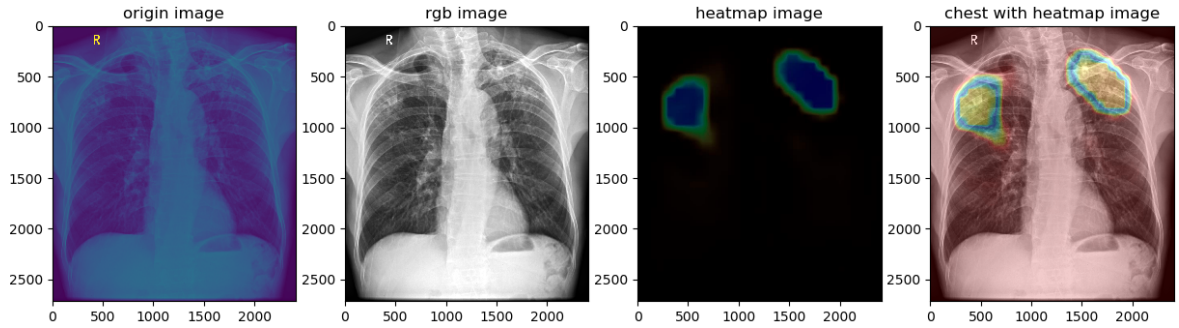
- **image** (np.ndarray)：二维样式的医学图像原图
- **heatmap** (np.ndarray)：`image` 对应的热图矩阵
- **alpha** (float)：`image` 对应的权重，取值范围为 `0 ~ 1`，默认为 `0.8`
- **beta** (float)：`heatmap` 对应的权重，取值范围为 `0 ~ 1`，默认为 `0.3`
- **colormap**：颜色图类型，默认为 `cv2.COLORMAP_JET`，详情参考 此文档

- **width** (int)：显示的窗宽，默认为 “None“，则 “width=max(image)-min(image)“
- **level** (int)：显示的窗位，默认为 “None“，则 “level=((image)+min(image))/2“

使用示例

```
import numpy as np
import matplotlib.pyplot as plt
from medcv.tools.transform import med2rgb
from medcv.data import chest_dcm, chest_heatmap
from medcv.visualize.cls import image_with_heatmap

# load chest dcm image and its heatmap
chest_dcm_image = chest_dcm()
chest_heatmap_image = chest_heatmap()

# dcm to rgb image with image window width and level
chest_rgb_image = med2rgb(chest_dcm_image, width=22135, level=12209)

# dcm with heatmap
chest_image_with_heatmap = image_with_heatmap(chest_dcm_image, chest_heatmap_image,␣
↪width=22135, level=12209)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('origin image')
plt.imshow(np.squeeze(chest_dcm_image))
plt.subplot(1, 4, 2)
plt.title('rgb image')
plt.imshow(chest_rgb_image)
plt.subplot(1, 4, 3)
plt.title('heatmap image')
plt.imshow(chest_heatmap_image)
plt.subplot(1, 4, 4)
plt.title('chest with heatmap image')
plt.imshow(chest_image_with_heatmap)
plt.show()
```

可视化



2.2 目标检测

2.2.1 image_with_bbox

```
medcv.visualize.det.imag_with_bbox(image, bbox, bbox_ids=None, colors=None,␣
↪thickness=1, width=None, level=None)
```

将目标检测框在原图上进行可视化，通过指定 `bbox_ids` 参数支持多个不同对象框的可视化

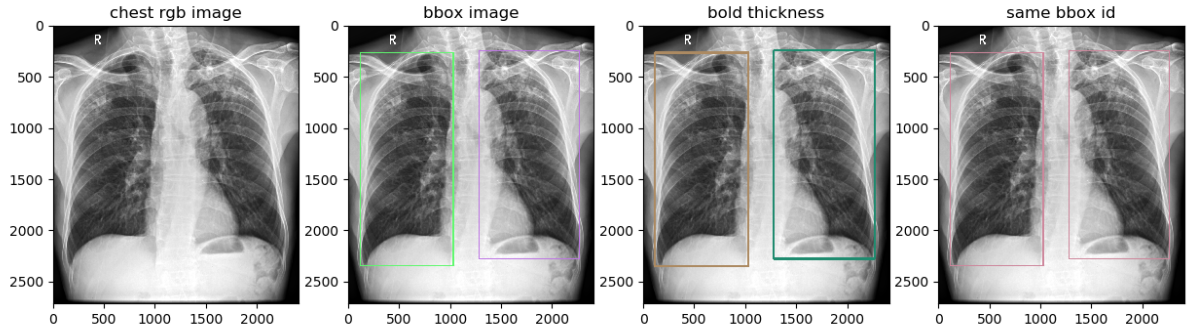
参数

- **image** (np.ndarray)：二维样式的医学图像原图
- **bbox** (list)：检测框列表，形式为 `[(x, y, w, h), (x, y, w, h), ...]`
- **bbox_ids** (list)：检测框的 id 列表，每个检测框对应一个 id，以区分不同的对象，默认为 `None`，即每个检测框对应不同的对象
- **colors** (list)：不同 id 对应 rgb 颜色列表，形式为 `[(r, g, b), (r, g, b), ...]`，如果为 `None`，颜色将会随机
- **thickness** (int)：边框的像素大小，默认为 1
- **width** (int)：显示的窗宽，默认为 “None“，则 “width=max(image)-min(image)“
- **level** (int)：显示的窗位，默认为 “None“，则 “level=((image)+min(image))/2“

使用示例

```
import matplotlib.pyplot as plt
from medcv.tools.transform import med2rgb
from medcv.data import chest_dcm, chest_bbox
from medcv.visualize.det import image_with_bbox

# load chest dcm image and its heatmap
chest_dcm_image = chest_dcm()
chest_bbox_dict = chest_bbox()

# bbox list, because chest_bbox_dict has left-lung and right-lung bbox
bbox_list = chest_bbox_dict.values()

# dcm to rgb image with image window width and level
chest_rgb_image = med2rgb(chest_dcm_image, width=22135, level=12209)

# dcm with bbox, default is different bbox id.
chest_image_with_bbox = image_with_bbox(chest_dcm_image, bbox_list, width=22135,␣
↪level=12209, thickness=10)

# dcm with bbox with thickness=20
chest_image_with_bold_bbox = image_with_bbox(chest_dcm_image, bbox_list, width=22135,␣
↪level=12209, thickness=20)

# dcm with same bbox id
chest_image_with_same_bbox = image_with_bbox(chest_dcm_image, bbox_list, bbox_ids=[0,␣
↪0], width=22135, level=12209, thickness=10)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('chest rgb image')
plt.imshow(chest_rgb_image)
plt.subplot(1, 4, 2)
plt.title('bbox image')
plt.imshow(chest_image_with_bbox)
plt.subplot(1, 4, 3)
plt.title('bold thickness')
plt.imshow(chest_image_with_bold_bbox)
plt.subplot(1, 4, 4)
plt.title('same bbox id')
plt.imshow(chest_image_with_same_bbox)
plt.show()
```

可视化



2.3 实例分割

2.3.1 imag_with_mask

```
medcv.visualize.seg.imag_with_mask(image, mask, alpha=0.5, colors=None, width=None,␣
↪level=None)
```

将标注的 Mask 在原图上进行可视化，例如：对比分割结果与金标准的重叠程度的可视化。

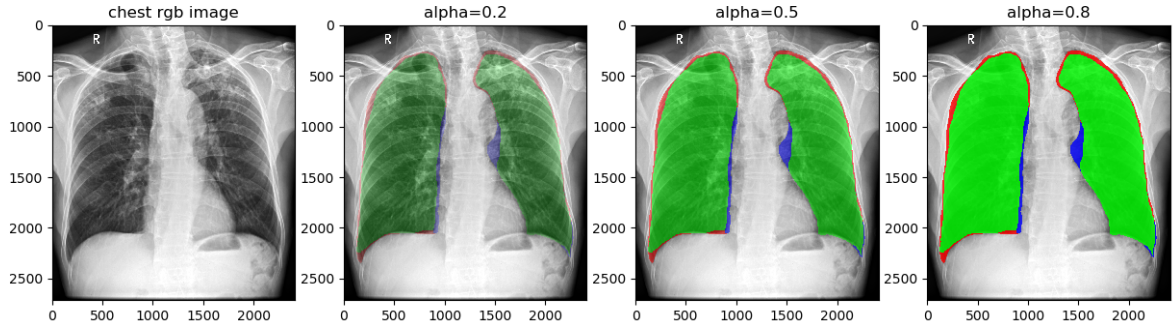
参数

- **image** (np.ndarray)：二维样式的医学图像原图
- **mask** (np.ndarray)：`image` 对应的 mask，默认 0 为背景，非零值为 ROI
- **alpha** (float)：`mask` 对应的加权值，取值范围为 `0 ~ 1`，默认为 `0.5`
- **colors** (list)：不同 ROI 对应 rgb 颜色列表，按照 ROI 的标注值进行升序索引，形式为 `[(r, g, b), (r, g, b), ...]`
- **width** (int)：显示的窗宽，默认为 `None`，则 “‘width=max(image)-min(image)“‘
- **level** (int)：显示的窗位，默认为 `None`，则 “‘level=((image)+min(image))/2“‘

使用示例

```
import numpy as np
import matplotlib.pyplot as plt
from medcv.tools.transform import med2rgb
from medcv.data import chest_dcm, chest_mask
from medcv.visualize.seg import image_with_mask

# load chest dcm image and its mask
chest_dcm_image = chest_dcm()
chest_mask_image = chest_mask()

# dcm to rgb image with image window width and level
chest_rgb_image = med2rgb(chest_dcm_image, width=22135, level=12209)

# visualize colors
colors = [(0, 255, 0), (255, 0, 0), (0, 0, 255)]

# alpha=0.2
chest_image_with_mask1 = image_with_mask(chest_dcm_image, chest_mask_image,␣
↪colors=colors, alpha=0.2, width=22135, level=12209)
# alpha=0.5 (default alpha=0.5)
chest_image_with_mask2 = image_with_mask(chest_dcm_image, chest_mask_image,␣
↪colors=colors, alpha=0.5, width=22135, level=12209)
# alpha=0.8
chest_image_with_mask3 = image_with_mask(chest_dcm_image, chest_mask_image,␣
↪colors=colors, alpha=0.8, width=22135, level=12209)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('chest rgb image')
plt.imshow(chest_rgb_image)
plt.subplot(1, 4, 2)
plt.title('alpha=0.2')
plt.imshow(chest_image_with_mask1)
plt.subplot(1, 4, 3)
plt.title('alpha=0.5')
plt.imshow(chest_image_with_mask2)
plt.subplot(1, 4, 4)
plt.title('alpha=0.8')
plt.imshow(chest_image_with_mask3)
plt.show()
```

可视化



备注说明：绿色为金标准与分割结果的重叠部分，红色 + 绿色 = 金标准区域，蓝色 + 绿色 = 分割结果区域

2.3.2 image_with_contours

```
medcv.visualize.seg.image_with_contours(image, mask, colors=None, thickness=1,␣
↪width=None, level=None)
```

本函数内部采用 `cv2.drawContours` 函数实现，将标注的 Mask 的边缘轮廓在原图上进行可视化，例如：对比分割结果与金标准的边缘信息的可视化。

参数

- **image** (np.ndarray)：二维样式的医学图像原图
- **mask** (np.ndarray)：`image` 对应的 mask，默认 0 为背景，非零值为 ROI
- **colors** (list)：不同 ROI 对应 rgb 颜色列表，按照 ROI 的标注值进行升序索引，形式为 `[(r, g, b), (r, g, b), ...]`
- **thickness** (int)：边缘轮廓线的像素大小
- **width** (int)：显示的窗宽，默认为 “None“，则 “width=max(image)-min(image)“
- **level** (int)：显示的窗位，默认为 “None“，则 “level=((image)+min(image))/2“

使用示例

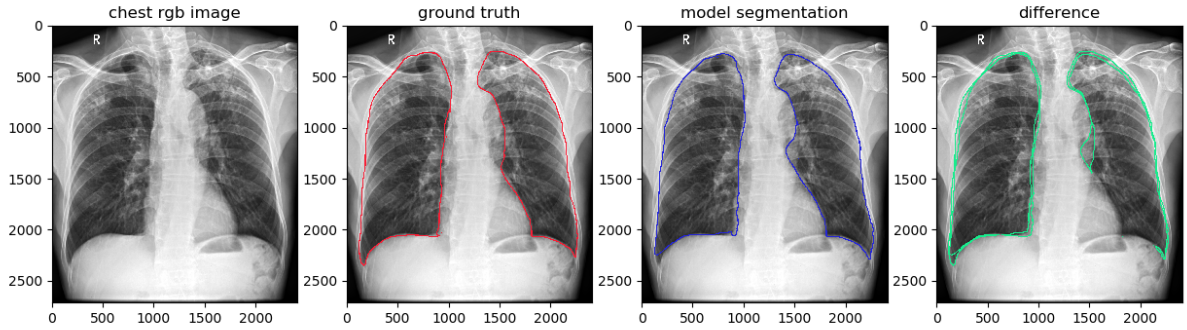
```
import numpy as np
import matplotlib.pyplot as plt
from medcv.tools.transform import med2rgb
from medcv.data import chest_dcm, chest_mask
from medcv.visualize.seg import image_with_contours

# load chest dcm image and its mask
chest_dcm_image = chest_dcm()
chest_mask_image = chest_mask()

# dcm to rgb image with image window width and level
chest_rgb_image = med2rgb(chest_dcm_image, width=22135, level=12209)

# ground truth
gt_mask = np.zeros_like(chest_mask_image)
gt_mask[np.logical_or(chest_mask_image == 1, chest_mask_image == 2)] = 1
chest_image_with_gt = image_with_contours(chest_dcm_image, gt_mask, thickness=10,␣
↪width=22135, level=12209)
# model segmentation
seg_mask = np.zeros_like(chest_mask_image)
seg_mask[np.logical_or(chest_mask_image == 1, chest_mask_image == 3)] = 1
chest_image_with_seg = image_with_contours(chest_dcm_image, seg_mask, thickness=10,␣
↪width=22135, level=12209)
# difference between ground truth and model segmentation
diff_mask = np.zeros_like(chest_mask_image)
diff_mask[np.logical_or(chest_mask_image == 2, chest_mask_image == 3)] = 1
chest_image_with_diff = image_with_contours(chest_dcm_image, diff_mask, thickness=10,␣
↪width=22135, level=12209)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('chest rgb image')
plt.imshow(chest_rgb_image)
plt.subplot(1, 4, 2)
plt.title('ground truth')
plt.imshow(chest_image_with_gt)
plt.subplot(1, 4, 3)
plt.title('model segmentation')
plt.imshow(chest_image_with_seg)
plt.subplot(1, 4, 4)
plt.title('difference')
```

(下页继续)

(续上页)

```
plt.imshow(chest_image_with_diff)
plt.show()
```

可视化



CHAPTER 3

工具类函数

3.1 图像转换

图像转换是通过建立相应的 LUT，将医学图像原图映射到一个理想的输出格式，可用于医学图像窗宽窗位的调节、医学图像转自然图像格式（映射到 0 - 255 之间）等场景。

3.1.1 med2med

```
medcv.tools.med2med(med_image, width=None, level=None, dtype=np.float64, invert=False)
```

该函数主要用于进行高效地医学图像窗宽窗位的调节，返回调窗后的医学图像，支持二维或者三维的输入。

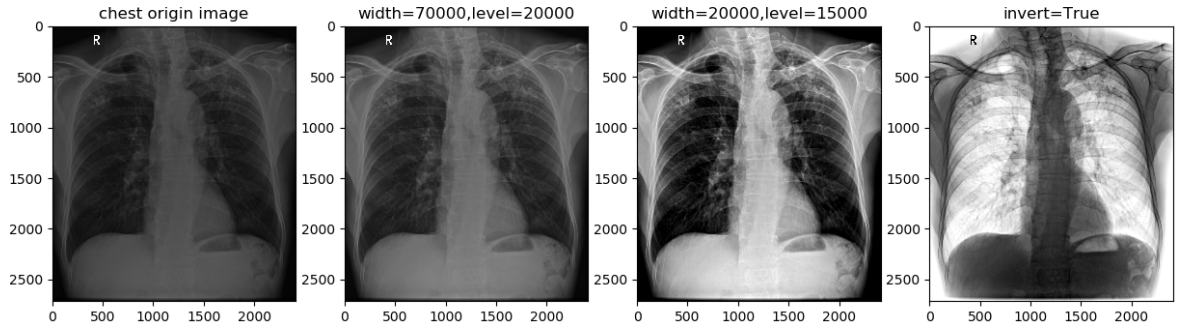
参数

- **med_image** (np.ndarray)：医学图像，二维 or 三维都可
- **width** (int)：显示的窗宽，默认为 `None`，则 `width=max(image)-min(image)`
- **level** (int)：显示的窗位，默认为 `None`，则 `level=((image)+min(image))/2`
- **dtype**：输出的数据类型，默认为 `np.float64`
- **invert**：是否进行反色操作，即为倒序映射，默认为 `False`

使用实例

```
import matplotlib.pyplot as plt
from medcv.data import chest_dcm
from medcv.tools.transform import med2med, med2rgb

# load chest dcm image
chest_dcm_image = chest_dcm()

# width=50000, level=20000
adjust_image1 = med2med(chest_dcm_image, width=50000, level=20000)

# width=20000, level=15000
adjust_image2 = med2med(chest_dcm_image, width=20000, level=15000)

# width=20000, level=15000, invert=True
adjust_image3 = med2med(chest_dcm_image, width=20000, level=15000, invert=True)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('chest origin image')
plt.imshow(med2rgb(chest_dcm_image))
plt.subplot(1, 4, 2)
plt.title('width=70000,level=20000')
plt.imshow(med2rgb(adjust_image1))
plt.subplot(1, 4, 3)
plt.title('width=20000,level=15000')
plt.imshow(med2rgb(adjust_image2))
plt.subplot(1, 4, 4)
plt.title('invert=True')
plt.imshow(med2rgb(adjust_image3))
plt.show()
```

可视化



3.1.2 med2grey

```
medcv.tools.med2grey(med_image, width=None, level=None, invert=False)
```

该函数功能是将二维的医学图像转为灰度图，支持窗宽窗位设置，返回转换后的灰度图，以此来兼容 OpenCV 的输入格式。

参数

- **med_image** (np.ndarray)：二维样式的医学图像
- **width** (int)：显示的窗宽，默认为 `None`，则 `width=max(image)-min(image)`
- **level** (int)：显示的窗位，默认为 `None`，则 `level=((image)+min(image))/2`
- **invert**：是否进行反色操作，即为倒序映射，默认为 `False`

使用实例

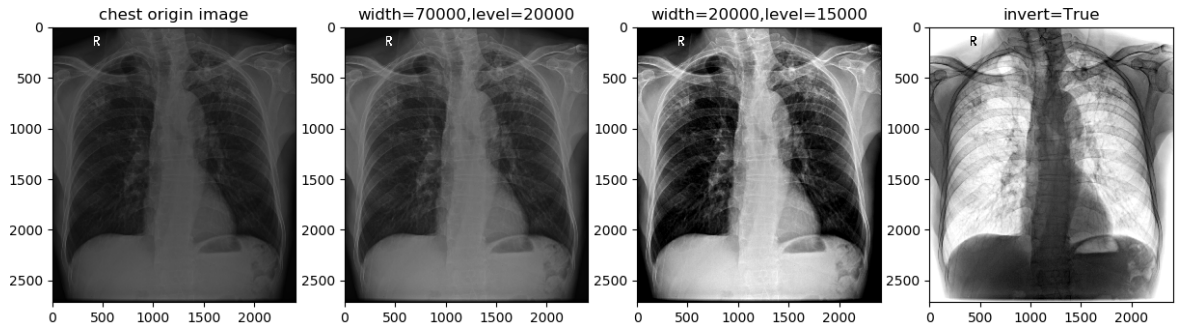
```
import matplotlib.pyplot as plt
from medcv.data import chest_dcm
from medcv.tools.transform import med2grey

# load chest dcm image
chest_dcm_image = chest_dcm()

# width=50000, level=20000
grey_image1 = med2grey(chest_dcm_image, width=50000, level=20000)

# width=20000, level=15000
```

(下页继续)

(续上页)

```
grey_image2 = med2grey(chest_dcm_image, width=20000, level=15000)

# width=20000, level=15000, invert=True
grey_image3 = med2grey(chest_dcm_image, width=20000, level=15000, invert=True)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('chest origin image')
plt.imshow(med2grey(chest_dcm_image), cmap='gray')
plt.subplot(1, 4, 2)
plt.title('width=70000,level=20000')
plt.imshow(grey_image1, cmap='gray')
plt.subplot(1, 4, 3)
plt.title('width=20000,level=15000')
plt.imshow(grey_image2, cmap='gray')
plt.subplot(1, 4, 4)
plt.title('invert=True')
plt.imshow(grey_image3, cmap='gray')
plt.show()
```

可视化



3.1.3 med2rgb

```
medcv.tools.med2rgb(med_image, width=None, level=None, invert=False)
```

该函数功能是将二维的医学图像转为 rgb 格式的自然图像，支持窗宽窗位设置，返回转换后的 RGB 图像，以此来兼容 OpenCV 的输入格式。

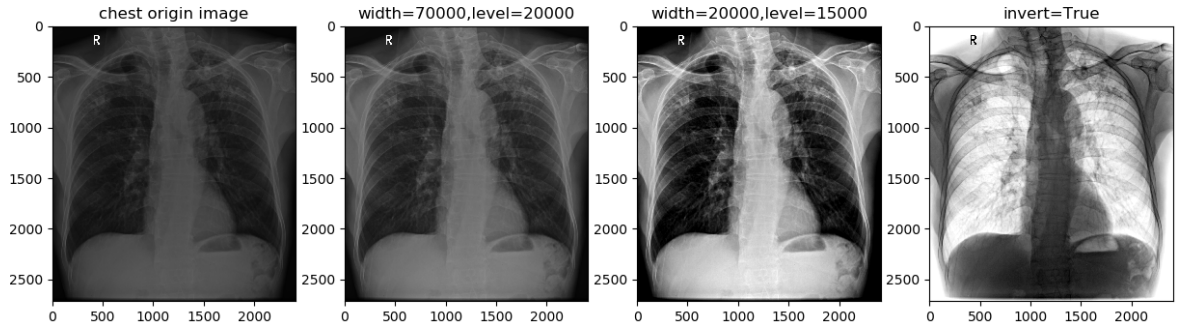
参数

- **med_image** (np.ndarray)：二维样式的医学图像
- **width** (int)：显示的窗宽，默认为 `None`，则 `width=max(image)-min(image)`
- **level** (int)：显示的窗位，默认为 `None`，则 `level=((image)+min(image))/2`
- **invert**：是否进行反色操作，即为倒序映射，默认为 `False`

使用实例

```
import matplotlib.pyplot as plt
from medcv.data import chest_dcm
from medcv.tools.transform import med2rgb

# load chest dcm image
chest_dcm_image = chest_dcm()

# width=50000, level=20000
rgb_image1 = med2rgb(chest_dcm_image, width=50000, level=20000)

# width=20000, level=15000
rgb_image2 = med2rgb(chest_dcm_image, width=20000, level=15000)

# width=20000, level=15000, invert=True
rgb_image3 = med2rgb(chest_dcm_image, width=20000, level=15000, invert=True)

# plot visualize
plt.figure(figsize=(15, 5))
plt.subplot(1, 4, 1)
plt.title('chest origin image')
plt.imshow(med2rgb(chest_dcm_image))
plt.subplot(1, 4, 2)
plt.title('width=70000,level=20000')
plt.imshow(rgb_image1)
plt.subplot(1, 4, 3)
plt.title('width=20000,level=15000')
plt.imshow(rgb_image2)
plt.subplot(1, 4, 4)
plt.title('invert=True')
plt.imshow(rgb_image3)
plt.show()
```

可视化


chest origin image
width=70000,level=20000
width=20000,level=15000
invert=True
R
0
500
1000
1500
2000
2500

CHAPTER 4

PyQt5 控件

4.1 QMedManger

```
medcv.gui.QMedManager()
```

医学图像管理者，用以管理医学图像信息以及的数据格式转换。该类同时也是其他 GUI 控件的内置管理者，完成医学图像与 PyQt5 的高效协作。

4.1.1 API 接口

设置

- `set_invert(invert)`：设置反色模式
- `set_medical_image(medical_image)`：设置二维的医学图像
- `set_image_window_width(window_width)`：设置图像窗宽
- `set_image_window_level(window_level)`：设置图像窗位
- `set_image_window(window_width, window_level)`：设置窗宽窗位

获取

- `is_invert()`：是否为反色模式
- `get_medical_image()`：获取二维的医学图像
- `get_image_window_width()`：获取图像窗宽
- `get_image_window_level()`：获取图像窗位
- `get_image_window()`：获取窗宽窗位

其他

- `pixmap()`：返回 `pixmap` 的格式图像
- `update_lut_array()`：更新映射的 LUT

4.2 QMedLabel

```
medcv.gui.QMedLabel()
```

该类继承于 PyQt5 的 `QLabel` 类，封装为医学图像的显示面板，可快速完成医学图像的显示。

4.2.1 API 接口

设置

- `setInvert(invert)`：设置反色模式
- `setMedicalImage(medicalImage)`：设置二维的医学图像
- `setImageWindowWidth(windowWidth)`：设置图像窗宽
- `setImageWindowLevel(windowLevel)`：设置图像窗位
- `setImageWindow(windowWidth, windowLevel)`：设置窗宽窗位

获取

- `isInvert()`：是否为反色模式
- `getMedicalImage()`：获取二维的医学图像
- `getImageWindowWidth()`：获取图像窗宽
- `getImageWindowLevel()`：获取图像窗位

- getImageWindow()：获取窗宽窗位

4.2.2 使用示例

```
import sys
from medcv.gui import QMedLabel
from medcv.data import chest_dcm
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QSlider


application = QApplication(sys.argv)

main_widget = QWidget()
layout = QVBoxLayout(main_widget)

med_label = QMedLabel()
med_label.setMaximumSize(600, 600)
med_label.setScaledContents(True)
med_label.setMedicalImage(chest_dcm())


width_slider = QSlider(Qt.Horizontal)
level_slider = QSlider(Qt.Horizontal)
width_slider.setRange(1, 50000)
width_slider.valueChanged.connect(med_label.setImageWindowWidth)
level_slider.setRange(0, 50000)
level_slider.valueChanged.connect(med_label.setImageWindowLevel)

layout.addWidget(med_label)
layout.addWidget(width_slider)
layout.addWidget(level_slider)

main_widget.show()

sys.exit(application.exec_())
```

CHAPTER 5

Utils 基础类

5.1 LUT

查找表（Lookup Table, LUT）是一个数组，通过索引 LUT 取代复杂的操作，将输入数据转化为更理想的输出格式，从而节省大量的处理时间。尤其是在交互界面时，交互的流畅性直接决定着用户的使用体验。

针对医学图像的处理，常见的大小为 512×512，更有是三维的图像格式。通过建立 LUT，高效地完成图像调窗操作，以及图像格式的转换，为 GUI 控件的交互的流畅性提供了保障。

5.1.1 映射函数

医学图像通常有很高的动态范围（比如：CT 的动态范围一般为-1024~1024），可以通过映射函数以完成图像的显示。依据DICOM 协议 中，主要定义了三个映射函数：`LINEAR`、`LINEAR_EXACT`、`SIGMOID`（MedCV 内部默认使用 `LINEAR`），每个函数都与窗口中心（窗位）和窗口宽度（窗宽）双变量相关，具体请参考源码

5.1.2 LUT 建立

```
medcv.utils.generate_lut_array(image, width, center, y_min=0, y_max=255, dtype=np.
↪uint8, invert=False, mode=medcv.LINEAR_MODE)
```

该函数根据传入的图像数据生成对应的 LUT，返回值为：`offset` 和 `lut_array`，具体请参考源码

参数

- **image** (np.ndarray)：医学图像格式，支持二维 or 三维输入
- **width** (int)：图像窗宽
- **center** (int)：图像窗位
- **y_min** (number)：映射函数对应的最小值，默认为 `0`
- **y_max** (number)：映射函数对应的最大值，默认为 `255`
- **dtype**：`lut_array` 数据类型，默认为 `np.uint8`
- **invert** (bool)：是否为反色模式，默认为 `False`
- **mode**：映射函数选择，可选 `LINEAR`、`LINEAR_EXACT` 和 `SIGMOID`，默认为 `LINEAR`

5.2 参数校验

```
medcv.utils.expected_type(*type_args, **type_kwargs)
```

基于 Python 函数闭包特性和装饰器机制，结合 `inspect` 库 `signature` 方法，优雅且简洁地完成函数形参类型的检查与校验，具体请参考源码

CHAPTER 6

更新日志

- 2021.04.22
  - 发布正式版 v1.0
  - 确定 MedCV 的框架结构

CHAPTER 7

常见问题

待更新…

CHAPTER 8

附录

待补充…

CHAPTER 9

Indices and tables

- genindex
- modindex
- search