```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "
\n",
    "
\n",
    "
\n",
    "
\n",
    "
\n",
    "
"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "import os\n",
    "from pathlib import Path\n",
    "os.chdir('/SSD/home/ron/group_project/retinalysis')\n",
    "from vascx import RetinaLoader\n",
    "\n",
    "# path to fundus directory (choose between sample_500 and
sample_1)\n",
    "ds_path = Path(\"/SSD/home/ron/group_project/images/sample_500/
vascx_segm/fundus/\")\n",
    "\n",
    "# loading all retina objects in variable called \"retinas\"\n",
    "retinas = RetinaLoader.from_folder(ds_path,
fundus_subfolder=\"preprocessed_rgb\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
   "outputs": [],
   "source": [
    "import re\n",
    "import pandas as pd\n",
    "import numpy as np\n",
    "import networkx as nx\n",
    "import random\n",
    "\n",
    "from vascx.features.tortuosity import Tortuosity,
```

```
TortuosityMode\n",
    "tort = Tortuosity(mode=TortuosityMode.Segments)\n",
    "from vascx.analysis.aggregators import std\n",
    "tort_glob = Tortuosity(mode=TortuosityMode.Segments,
aggregator=std) #changer std en median_std et on ^rend ce qu'on
veut"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Fonctions complémentaires"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {},
   "outputs": [],
   "source": [
    "#fonction pour trouver l'équation de la droite à partir d'une
rétine\n",
    "\n",
    "def fd_eq(retina):\n",
    "\n",
    "    fov = retina.fovea_location\n",
    "    opdi = retina.disc.center_of_mass\n",
    "\n",
    "    if fov.x == opdi.x: #no div by zero\n",
    "        return (None, None)\n",
    "\n",
    "    m = (fov.y - opdi.y) / (fov.x - opdi.x)\n",
    "    b = fov.y - m * fov.x\n",
    "    return (m, b)\n",
    "\n",
    "#fonction pour touver le barycentre d'un connected
component\n",
    "\n",
    "def bary(graph):\n",
    "    x = 0\n",
    "    y = 0\n",
    "    nodes = graph.nodes\n",
    "    for node in nodes:\n",
    "        b = nodes[node]\n",
    "        c = b[\"o\"]\n",
    "        x += c[0]\n",
    "        y += c[1]\n",
    "\n",
    "    bary_x = x/len(graph.nodes)\n",
    "    bary_y = y/len(graph.nodes)\n",
    "    return (bary_x, bary_y)\n",
    "\n",
    "#pour un objet artries.graph/veins.graph trouver un
```

```
dictionnaire des connected components up et down\n",
    "\n",
    "def fd_sep(graph, discr_equation):\n",
    "    #extract connected components\n",
    "    connected_components = [graph.subgraph(c).copy() for c in
nx.connected_components(graph)]\n",
    "    cc = {'up': [], 'down': []} #dictionnaire\n",
    "    \n",
    "    for con in connected_components: #one cluster at a time\n",
    "        barycenter = bary(con)\n",
    "\n",
    "        if discr_equation[0] * barycenter[0] +
discr_equation[1] > barycenter[1]:\n",
    "            cc['down'] = cc.get('down', []) + [con]\n",
    "        else:\n",
    "            cc['up'] = cc.get('up', []) + [con]\n",
    "        \n",
    "\n",
    "    return (cc)\n",
    "\n",
    "# confronter une lite de graphs (du dictionnaire 'up' =
liste_graphs ou 'down' = liste_graphs) au dictionnaire 'tuple_edge'
= objet segemnt pour ressortir une liste de toutes les tortuosités
de tous les egments de la liste de graphs\n",
    "\n",
    "def get_all_tort(graph_list_inside_dict, seg_edges_dict):\n",
    "    \n",
    "    list_all_tort_per_seg_hemisphere = []\n",
    "    n_counter=0\n",
    "\n",
    "    for graph in graph_list_inside_dict:\n",
    "\n",
    "        tort_of_this_seg = 0 #initialiser\n",
    "\n",
    "        if len(graph.edges) != 0:\n",
    "            for edge in graph.edges:\n",
    "                if sum(edge) != edge[0]: #exclure toutes les
edges qui ne sont que des nodes\n",
    "                    if edge in seg_edges_dict:\n",
    "                        segment = seg_edges_dict[edge]\n",
    "                        if len(segment.skeleton) >=
tort.min_numpoints:\n",
    "                            tort_of_this_seg =
tort._compute_for_segment(segment)\n",
    "                        else:\n",
    "                            tort_of_this_seg = 0\n",
    "                    else:\n",
    "                        reversed_edge = tuple(reversed(edge))
\n",
    "                        if reversed_edge in seg_edges_dict:\n",
    "                            segment =
seg_edges_dict[reversed_edge]\n",
    "                            if len(segment.skeleton) >=
tort.min_numpoints:\n",
```

```
    "                                tort_of_this_seg = tort._compute_for_segment(segment)\n",
    "                            else:\n",
    "                                tort_of_this_seg = 0 \n",
    "                        else:\n",
    "                            tort_of_this_seg = 0\n",
    "                if tort_of_this_seg !=0:\n",
    "                    list_all_tort_per_seg_hemisphere.append(tort_of_this_seg)\n",
    "                    n_counter += 1 #that way if we weren't able to computa anything we give NA\n",
    "    if n_counter !=0:\n",
    "        return(list_all_tort_per_seg_hemisphere)\n",
    "    else:\n",
    "        return(\"NA\") \n",
    "        \n",
    "\n",
    "def from_distrib_get(list):\n",
    "    med = np.median(list)\n",
    "    sd = np.std(list)\n",
    "    return(med, sd)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Main function"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {},
   "outputs": [],
   "source": [
    "def tortle(retinalist):\n",
    "    \n",
    "    # Créer un DataFrame vide avec les colonnes spécifiées\n",
    "    columns = ['retina_id', 'global_tort_arteries_med', 'global_tort_arteries_sd', \n",
    "           'tort_up_arteries_med', 'tort_down_arteries_med', 'tort_up_arteries_sd', 'tort_down_arteries_sd',\n",
    "           'global_tort_veins_med', 'global_tort_veins_sd', 'tort_up_veins_med',\n",
    "           'tort_down_veins_med', 'tort_up_veins_sd', 'tort_down_veins_sd', 'asym_med_veins', 'asym_med_veins_random', 'asym_med_art', 'asym_med_art_random']\n",
    "    df_list = []  # Liste pour stocker les DataFrame temporaires\n",
    "\n",
    "    for retina in retinalist:\n",
    "\n",
    "        print(\"new one!\")\n",
```

```
    "        \n",
    "        discr_equation = fd_eq(retina)\n",
    "        arteries = retina.arteries\n",
    "        veins = retina.veins\n",
    "       \n",
    "        # ------------------------let's start with the
arteries-------------------------------------------------------------
---------------------------\n",
    "       \n",
    "        global_tort_arteries_med = tort.compute(arteries) #add
a try, except NA ? \n",
    "        global_tort_arteries_sd = tort_glob.compute(arteries)
#up there tort_glob is parametered to give the std \n",
    "        \n",
    "        #now let's create the dictionnary (tuple) = segment
object\n",
    "\n",
    "        seg_edges_dict = {}\n",
    "        for segment in arteries.segments:\n",
    "            mystring = str(segment)\n",
    "            numeric_values = re.findall(r'\\d+', mystring)
#expression régulière r'\\d+' : trouver toutes les séquences de
chiffres dans la chaîne\n",
    "            for i in range(len(numeric_values)):\n",
    "                numeric_values[i] = int(numeric_values[i])
#convert a list of strings to a list of integers\n",
    "                key_tuple = tuple(numeric_values)\n",
    "                seg_edges_dict[key_tuple] = segment\n",
    "        \n",
    "        #dictionnary created[0] * num_, let's extract the
connected components and attribute them to 'up' or 'down' \n",
    "\n",
    "        arteries_graph = arteries.graph\n",
    "        orientation_dict = fd_sep(arteries_graph,
discr_equation) #fd, foveau-disc separation\n",
    "\n",
    "        all_tort_up = get_all_tort(orientation_dict['up'],
seg_edges_dict)\n",
    "        \n",
    "        if all_tort_up != 'NA':\n",
    "            all_tort_up_clean = [x for x in all_tort_up if
isinstance(x, (int, float))]\n",
    "            tort_distrib = from_distrib_get(all_tort_up_clean)
\n",
    "            tort_up_arteries_med = tort_distrib[1]\n",
    "            tort_up_arteries_sd = tort_distrib[2]\n",
    "        else:\n",
    "            tort_up_arteries_med = 'NA'\n",
    "            tort_up_arteries_sd = 'NA'\n",
    "\n",
    "\n",
    "        all_tort_down = get_all_tort(orientation_dict['down'],
seg_edges_dict)\n",
    "\n",
```

```
    "        if all_tort_down != 'NA':\n",
    "            all_tort_down_clean = [x for x in all_tort_down if
isinstance(x, (int, float))]\n",
    "            tort_distrib =
from_distrib_get(all_tort_down_clean)\n",
    "            tort_down_arteries_med = tort_distrib[1]\n",
    "            tort_down_arteries_sd = tort_distrib[2]\n",
    "        else:\n",
    "            tort_down_arteries_med = 'NA'\n",
    "            tort_down_arteries_sd = 'NA'\n",
    "        \n",
    "        #if there are no NAs we can compute the regular
asymetry\n",
    "        shopping_cart_arteries = [tort_up_arteries_med,
tort_down_arteries_med]\n",
    "        if 'NA' in shopping_cart_arteries:\n",
    "            asym_med_art = 'NA'\n",
    "        else:\n",
    "            asym_med_art = (tort_up_arteries_med -
tort_down_arteries_med)/(tort_up_arteries_med +
tort_down_arteries_med)\n",
    "        \n",
    "        #random asymetry:\n",
    "        \n",
    "        all_tort_down_clean2 = [x for x in all_tort_down if
isinstance(x, (int, float))]\n",
    "        all_tort_up_clean2 = [x for x in all_tort_up if
isinstance(x, (int, float))]\n",
    "        all_tort = all_tort_down_clean2 +
all_tort_up_clean2\n",
    "        random.shuffle(all_tort)\n",
    "        new_up, new_down = all_tort[:len(all_tort)//2],
all_tort[len(all_tort)//2:]\n",
    "        new_up_med = np.median(new_up)\n",
    "        new_down_med = np.median(new_down)\n",
    "        asym_med_art_random = (new_up_med - new_down_med)/
(new_up_med + new_down_med)\n",
    "\n",
    "        # ------------------------Now with the
veins-----------------------------------------------------------
------------------------\n",
    "       \n",
    "        global_tort_veins_med = tort.compute(veins) #add a try,
except NA ? \n",
    "        global_tort_veins_sd = tort_glob.compute(veins)\n",
    "        \n",
    "        #now let's create the dictionnary (tuple) = segment
object\n",
    "\n",
    "        seg_edges_dict = {}\n",
    "        for segment in veins.segments:\n",
    "            mystring = str(segment)\n",
    "            numeric_values = re.findall(r'\\d+', mystring)
#expression régulière r'\\d+' : trouver toutes les séquences de
```

```
chiffres dans la chaîne\n",
    "            for i in range(len(numeric_values)):\n",
    "                numeric_values[i] = int(numeric_values[i])
#convert a list of strings to a list of integers\n",
    "                key_tuple = tuple(numeric_values)\n",
    "                seg_edges_dict[key_tuple] = segment\n",
    "        \n",
    "        #dictionnary created[0] * num_, let's extract the
connected components and attribute them to 'up' or 'down' \n",
    "\n",
    "        veins_graph = veins.graph\n",
    "        orientation_dict = fd_sep(veins_graph, discr_equation)
#fd, foveau-disc separation\n",
    "\n",
    "        all_tort_up = get_all_tort(orientation_dict['up'],
seg_edges_dict)\n",
    "        \n",
    "        if all_tort_up != 'NA':\n",
    "            all_tort_up_clean = [x for x in all_tort_up if
isinstance(x, (int, float))]\n",
    "            tort_distrib = from_distrib_get(all_tort_up_clean)
\n",
    "            tort_up_veins_med = tort_distrib[1]\n",
    "            tort_up_veins_sd = tort_distrib[2]\n",
    "        else:\n",
    "            tort_up_veins_med = 'NA'\n",
    "            tort_up_veins_sd = 'NA'\n",
    "\n",
    "\n",
    "        all_tort_down = get_all_tort(orientation_dict['down'],
seg_edges_dict)\n",
    "\n",
    "        if all_tort_down != 'NA':\n",
    "            all_tort_down_clean = [x for x in all_tort_down if
isinstance(x, (int, float))]\n",
    "            tort_distrib =
from_distrib_get(all_tort_down_clean)\n",
    "            tort_down_veins_med = tort_distrib[1]\n",
    "            tort_down_veins_sd = tort_distrib[2]\n",
    "        else:\n",
    "            tort_down_veins_med = 'NA'\n",
    "            tort_down_veins_sd = 'NA'\n",
    "        \n",
    "        #if there are no NAs we can compute the regular
asymetry\n",
    "        shopping_cart_veins = [tort_up_veins_med,
tort_down_veins_med]\n",
    "        if 'NA' in shopping_cart_veins:\n",
    "            asym_med_veins = 'NA'\n",
    "        else:\n",
    "            asym_med_veins = (tort_up_veins_med -
tort_down_veins_med)/(tort_up_veins_med + tort_down_veins_med)\n",
    "        \n",
    "        #random asymetry:\n",
```

```
    "        \n",
    "        all_tort_down_clean2 = [x for x in all_tort_down if
isinstance(x, (int, float))]\n",
    "        all_tort_up_clean2 = [x for x in all_tort_up if
isinstance(x, (int, float))] #instead of 0 if x == 'NA' else x for x
in all_tort_up, just to be sure no other weird characters\n",
    "        all_tort = all_tort_down_clean2 +
all_tort_up_clean2\n",
    "        random.shuffle(all_tort)\n",
    "        new_up, new_down = all_tort[:len(all_tort)//2],
all_tort[len(all_tort)//2:]\n",
    "        new_up_med = np.median(new_up)\n",
    "        new_down_med = np.median(new_down)\n",
    "        asym_med_veins_random = (new_up_med - new_down_med)/
(new_up_med + new_down_med)\n",
    "\n",
    "
#-----------------------------------------------------------------
--------------------\n",
    "        #maintenant il faut append au panda\n",
    "\n",
    "        # À chaque itération, ajouter une nouvelle ligne au
DataFrame\n",
    "        new_row = {\n",
    "            'retina_id': retina.id,\n",
    "            'global_tort_arteries_med':
global_tort_arteries_med,\n",
    "            'global_tort_arteries_sd': global_tort_arteries_sd,
\n",
    "            'tort_up_arteries_med': tort_up_arteries_med,\n",
    "            'tort_down_arteries_med': tort_down_arteries_med,
\n",
    "            'tort_up_arteries_sd': tort_up_arteries_sd,\n",
    "            'tort_down_arteries_sd': tort_down_arteries_sd,\n",
    "            'global_tort_veins_med': global_tort_veins_med,\n",
    "            'global_tort_veins_sd': global_tort_veins_sd,\n",
    "            'tort_up_veins_med': tort_up_veins_med,\n",
    "            'tort_down_veins_med': tort_down_veins_med,\n",
    "            'tort_up_veins_sd': tort_up_veins_sd,\n",
    "            'tort_down_veins_sd': tort_down_veins_sd,\n",
    "            'asym_med_veins': asym_med_veins,\n",
    "            'asym_med_veins_random': asym_med_veins_random,\n",
    "            'asym_med_art': asym_med_art,\n",
    "            'asym_med_art_random': asym_med_art_random\n",
    "        }\n",
    "        temp_df = pd.DataFrame([new_row], columns=columns)  #
Utilisez les colonnes prédéfinies ici\n",
    "        df_list.append(temp_df)\n",
    "    \n",
    "    df = pd.concat(df_list, ignore_index=True)\n",
    "    return(df)\n",
    "\n",
    "\n"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
```

```
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
```

```
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
```

```
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
```

```
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
```

```
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
```

```
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
```

```
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
```

```
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
```

```
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
```

```
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
```

```
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
```

```
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
```

```
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
```

```
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
```

```
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
```

```
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
```

```
mask. \n",
       "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
      ]
     },
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "new one!\n",
       "new one!\n",
       "new one!\n",
       "new one!\n",
       "new one!\n",
       "new one!\n"
      ]
     },
     {
      "name": "stderr",
      "output_type": "stream",
      "text": [
       "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
       "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
      ]
     },
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "new one!\n",
       "new one!\n",
       "new one!\n",
       "new one!\n"
      ]
     },
     {
      "name": "stderr",
      "output_type": "stream",
      "text": [
       "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
       "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
      ]
     },
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "new one!\n",
```

```
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
```

```
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
```

```
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
```

```
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
```

```
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
```

```
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
```

```
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
```

```
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
```

```
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n",
      "new one!\n"
     ]
    },
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "/SSD/home/ron/group_project/retinalysis/rtnls_enface/disc.py:
33: UserWarning: Found more than one connected component for disc
mask. \n",
      "  warnings.warn(\"Found more than one connected component for
disc mask. \")\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "new one!\n"
     ]
    }
   ],
   "source": [
```

```
    "#five_ret = retinas[:10]\n",
    "a = tortle(retinas)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 9,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "CSV file saved successfully!\n"
     ]
    }
   ],
   "source": [
    "filename_path = (\"/SSD/home/ron/group_project/notebooks/Louis/
five_hundred.csv\")\n",
    "try:\n",
    "    a.to_csv(filename_path, sep='\\t', na_rep='NaN',
index=False)  # Change 'output.csv' to the desired filename\n",
    "    print(\"CSV file saved successfully!\")\n",
    "except Exception as e:\n",
    "    print(\"Error:\", e)\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### trouver le problème dans fd_sep"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 57,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "(-0.05508136378239478, 554.2627210502066)\n",
      "{'up': [<networkx.classes.graph.Graph object at
0x7fe2572abf70>, <networkx.classes.graph.Graph object at
0x7fe2572abfd0>, <networkx.classes.graph.Graph object at
0x7fe2572abdf0>, <networkx.classes.graph.Graph object at
0x7fe2572abeb0>, <networkx.classes.graph.Graph object at
0x7fe2572aace0>, <networkx.classes.graph.Graph object at
0x7fe2572abd30>, <networkx.classes.graph.Graph object at
0x7fe2572a9cc0>, <networkx.classes.graph.Graph object at
0x7fe2572abe80>, <networkx.classes.graph.Graph object at
```

0x7fe2572a9c60>, <networkx.classes.graph.Graph object at
0x7fe2572a9c90>, <networkx.classes.graph.Graph object at
0x7fe2572abd60>, <networkx.classes.graph.Graph object at
0x7fe2572abd90>, <networkx.classes.graph.Graph object at
0x7fe2572a9ba0>, <networkx.classes.graph.Graph object at
0x7fe2572a9bd0>, <networkx.classes.graph.Graph object at
0x7fe2572a9b40>, <networkx.classes.graph.Graph object at
0x7fe2572a9b10>, <networkx.classes.graph.Graph object at
0x7fe2572abe50>, <networkx.classes.graph.Graph object at
0x7fe2572abe20>, <networkx.classes.graph.Graph object at
0x7fe2572a9ab0>, <networkx.classes.graph.Graph object at
0x7fe2572a9930>, <networkx.classes.graph.Graph object at
0x7fe2572a9960>, <networkx.classes.graph.Graph object at
0x7fe2572a9cf0>, <networkx.classes.graph.Graph object at
0x7fe2572a9ae0>, <networkx.classes.graph.Graph object at
0x7fe2572a9750>, <networkx.classes.graph.Graph object at
0x7fe2572abd00>, <networkx.classes.graph.Graph object at
0x7fe2572a97b0>, <networkx.classes.graph.Graph object at
0x7fe2572a9840>, <networkx.classes.graph.Graph object at
0x7fe2572a9a80>, <networkx.classes.graph.Graph object at
0x7fe2572a9600>, <networkx.classes.graph.Graph object at
0x7fe2572a9780>, <networkx.classes.graph.Graph object at
0x7fe2572aaa70>, <networkx.classes.graph.Graph object at
0x7fe2572a96f0>, <networkx.classes.graph.Graph object at
0x7fe2572aa980>, <networkx.classes.graph.Graph object at
0x7fe2572aa8f0>], 'down': [<networkx.classes.graph.Graph object at
0x7fe2572aad10>, <networkx.classes.graph.Graph object at
0x7fe2572aac80>, <networkx.classes.graph.Graph object at
0x7fe2572a9d50>, <networkx.classes.graph.Graph object at
0x7fe2572abdc0>, <networkx.classes.graph.Graph object at
0x7fe2572a9d20>, <networkx.classes.graph.Graph object at
0x7fe2572a9b70>, <networkx.classes.graph.Graph object at
0x7fe2572a9c00>, <networkx.classes.graph.Graph object at
0x7fe2572a99c0>, <networkx.classes.graph.Graph object at
0x7fe2572a9c30>, <networkx.classes.graph.Graph object at
0x7fe2572a9a20>, <networkx.classes.graph.Graph object at
0x7fe2572a9900>, <networkx.classes.graph.Graph object at
0x7fe2572a98d0>, <networkx.classes.graph.Graph object at
0x7fe2572a98a0>, <networkx.classes.graph.Graph object at
0x7fe2572a99f0>, <networkx.classes.graph.Graph object at
0x7fe2572a97e0>, <networkx.classes.graph.Graph object at
0x7fe2572a9990>, <networkx.classes.graph.Graph object at
0x7fe2572a9870>, <networkx.classes.graph.Graph object at
0x7fe2572a9a50>, <networkx.classes.graph.Graph object at
0x7fe2572a9630>, <networkx.classes.graph.Graph object at
0x7fe2572a9720>, <networkx.classes.graph.Graph object at
0x7fe2572aab30>, <networkx.classes.graph.Graph object at
0x7fe2572aaaa0>, <networkx.classes.graph.Graph object at
0x7fe2572a9810>, <networkx.classes.graph.Graph object at
0x7fe2572a96c0>, <networkx.classes.graph.Graph object at
0x7fe2572aaad0>, <networkx.classes.graph.Graph object at
0x7fe2572a9660>, <networkx.classes.graph.Graph object at
0x7fe2572aa920>, <networkx.classes.graph.Graph object at
0x7fe2572aaa10>]}\n",

```
      "[<networkx.classes.graph.Graph object at 0x7fe2572abf70>,
<networkx.classes.graph.Graph object at 0x7fe2572abfd0>,
<networkx.classes.graph.Graph object at 0x7fe2572abdf0>,
<networkx.classes.graph.Graph object at 0x7fe2572abeb0>,
<networkx.classes.graph.Graph object at 0x7fe2572aace0>,
<networkx.classes.graph.Graph object at 0x7fe2572abd30>,
<networkx.classes.graph.Graph object at 0x7fe2572a9cc0>,
<networkx.classes.graph.Graph object at 0x7fe2572abe80>,
<networkx.classes.graph.Graph object at 0x7fe2572a9c60>,
<networkx.classes.graph.Graph object at 0x7fe2572a9c90>,
<networkx.classes.graph.Graph object at 0x7fe2572abd60>,
<networkx.classes.graph.Graph object at 0x7fe2572abd90>,
<networkx.classes.graph.Graph object at 0x7fe2572a9ba0>,
<networkx.classes.graph.Graph object at 0x7fe2572a9bd0>,
<networkx.classes.graph.Graph object at 0x7fe2572a9b40>,
<networkx.classes.graph.Graph object at 0x7fe2572a9b10>,
<networkx.classes.graph.Graph object at 0x7fe2572abe50>,
<networkx.classes.graph.Graph object at 0x7fe2572abe20>,
<networkx.classes.graph.Graph object at 0x7fe2572a9ab0>,
<networkx.classes.graph.Graph object at 0x7fe2572a9930>,
<networkx.classes.graph.Graph object at 0x7fe2572a9960>,
<networkx.classes.graph.Graph object at 0x7fe2572a9cf0>,
<networkx.classes.graph.Graph object at 0x7fe2572a9ae0>,
<networkx.classes.graph.Graph object at 0x7fe2572a9750>,
<networkx.classes.graph.Graph object at 0x7fe2572abd00>,
<networkx.classes.graph.Graph object at 0x7fe2572a97b0>,
<networkx.classes.graph.Graph object at 0x7fe2572a9840>,
<networkx.classes.graph.Graph object at 0x7fe2572a9a80>,
<networkx.classes.graph.Graph object at 0x7fe2572a9600>,
<networkx.classes.graph.Graph object at 0x7fe2572a9780>,
<networkx.classes.graph.Graph object at 0x7fe2572aaa70>,
<networkx.classes.graph.Graph object at 0x7fe2572a96f0>,
<networkx.classes.graph.Graph object at 0x7fe2572aa980>,
<networkx.classes.graph.Graph object at 0x7fe2572aa8f0>]\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "{'up': [<networkx.classes.graph.Graph at 0x7fe271da79a0>,
\n",
       "  <networkx.classes.graph.Graph at 0x7fe26a393550>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe26a393ac0>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe26a392ce0>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe26a393220>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe26a3922f0>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe26de77730>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d4640>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d6b90>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d49d0>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d74c0>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d7640>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d6170>,\n",
       "  <networkx.classes.graph.Graph at 0x7fe2686d7bb0>,\n",
```

```
        "  <networkx.classes.graph.Graph at 0x7fe2686d5660>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d7b20>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5360>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6020>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d7760>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d65c0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6aa0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6e30>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5240>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6fb0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d68c0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5420>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5540>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2722ff820>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268411d80>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268410f10>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2684137f0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412440>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412530>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268413040>],\n",
        " 'down': [<networkx.classes.graph.Graph at 0x7fe271da7fd0>,
\n",
        "  <networkx.classes.graph.Graph at 0x7fe2722fff70>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe26a393d30>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe26a3927d0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d4c10>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d63b0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6290>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d7550>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d63e0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6710>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d4e80>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5390>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5ff0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d5c60>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d4130>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d4460>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2686d6980>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2684124a0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412890>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2684118a0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268411390>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412830>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412800>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe2684138b0>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412350>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268411450>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268413100>,\n",
        "  <networkx.classes.graph.Graph at 0x7fe268412b00>]}"
       ]
      },
      "execution_count": 57,
      "metadata": {},
      "output_type": "execute_result"
```

```
    }
   ],
   "source": [
    "one = retinas[7] #peu importe la rétine le dictionnaire 'up' a
toujours un tout petit nombre de segments\n",
    "discr_eq = fd_eq(one)\n",
    "print(discr_eq)\n",
    "a = fd_sep(one.arteries.graph, discr_eq)\n",
    "print(a)\n",
    "#one.arteries.plot_graph()\n",
    "#one.arteries.plot_segments()\n",
    "a['up']\n",
    "print(a['up'])\n",
    "\n",
    "\n",
    "def fd_eq(retina):\n",
    "\n",
    "    fov = retina.fovea_location\n",
    "    opdi = retina.disc.center_of_mass\n",
    "\n",
    "    if fov.x == opdi.x: #no div by zero\n",
    "        return (None, None)\n",
    "\n",
    "    m = (fov.y - opdi.y) / (fov.x - opdi.x)\n",
    "    b = fov.y - m * fov.x\n",
    "    return (m, b)\n",
    "\n",
    "def bary(graph):\n",
    "    import networkx as nx\n",
    "    x = 0\n",
    "    y = 0\n",
    "    nodes = graph.nodes\n",
    "    for node in nodes:\n",
    "        b = nodes[node]\n",
    "        c = b[\"o\"]\n",
    "        x += c[0]\n",
    "        y += c[1]\n",
    "\n",
    "    bary_x = x/len(graph.nodes)\n",
    "    bary_y = y/len(graph.nodes)\n",
    "    return (bary_x, bary_y)\n",
    "\n",
    "    "#pour un objet artries.graph/veins.graph trouver un
dictionnaire des connected components up et down\n",
    "\n",
    "def fd_sep_new(graph, discr_equation):\n",
    "    #extract connected components\n",
    "    connected_components = [graph.subgraph(c).copy() for c in
nx.connected_components(graph)]\n",
    "    cc = {'up': [], 'down': []} #dictionnaire\n",
    "    \n",
    "    for con in connected_components: #one cluster at a time\n",
    "        barycenter = bary(con)\n",
    "\n",
```

```
    "        if (discr_equation[0] * barycenter[0] +
discr_equation[1]) < barycenter[1]:\n",
    "            cc['up'] = cc.get('up', []) + [con]\n",
    "        else:\n",
    "            cc['down'] = cc.get('down', []) + [con]\n",
    "        \n",
    "\n",
    "    return (cc)\n",
    "\n",
    "fd_sep_new(one.arteries.graph, fd_eq(one))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "#### Export as .CSV file"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "retinalysis",
   "language": "python",
   "name": "retinalysis"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.10.13"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```