计算机图形学实验报告

李明亨 PB21010375

1 实验内容

实现图像变形的 IDW 算法和 RBF 算法。

2 程序介绍

程序结构如图2所示。我们定义了 Warping 父类，在其中定义了虚函数`warping()`，输入输出分别为像素点在图像变形前后的坐标，并在 WarpingRBF 类和 WarpingIDW 类中对计算方法作出具体实现。

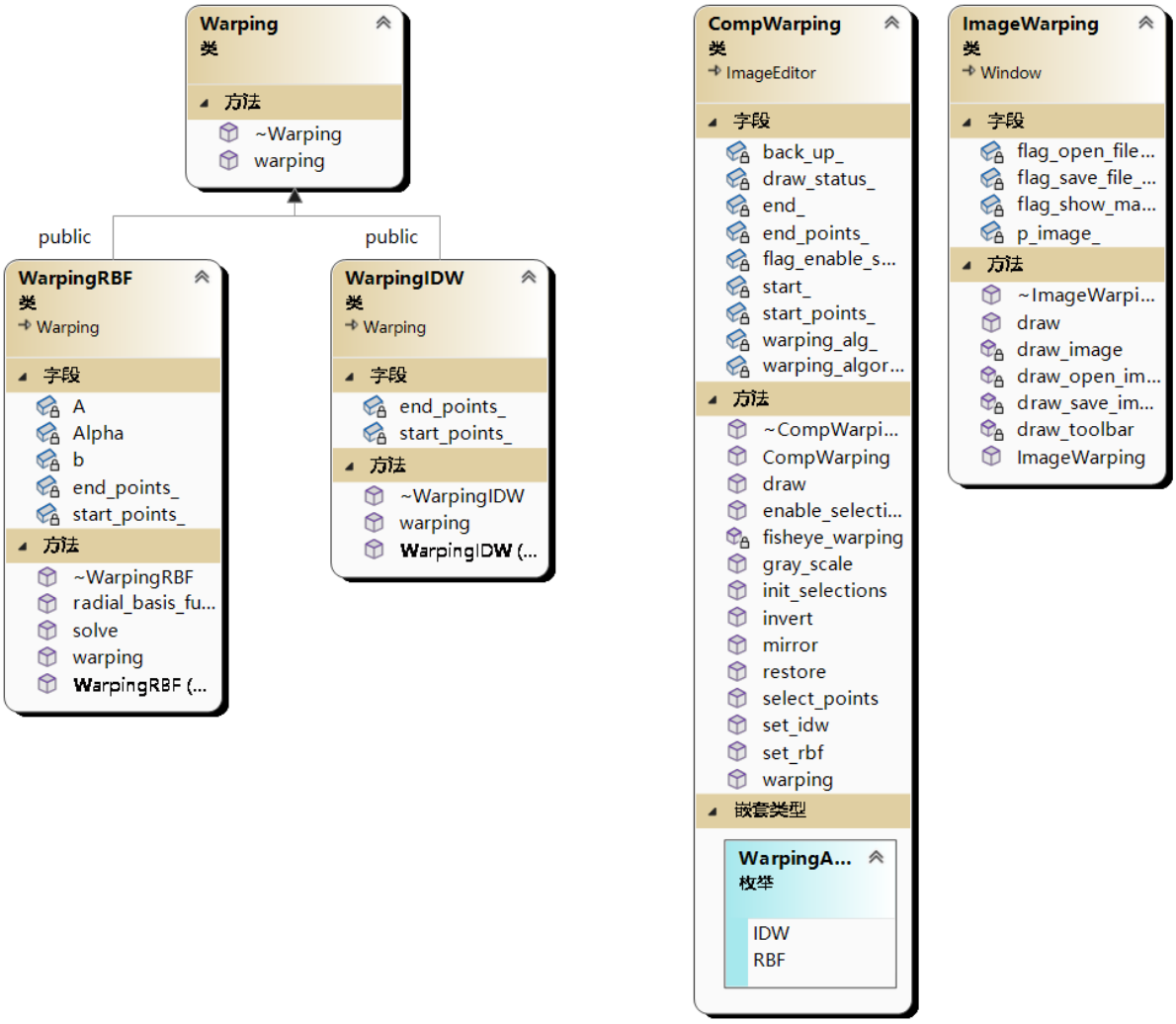


图 1

交互上，我们在 Warping 按钮下加入了子菜单，用户可以自行选择合适的算法，交互部分的代码如下：

```
if (ImGui::MenuItem("Warping") && p_image_)
{
  ImGui::OpenPopup("SubMenuWarping");
}
// HW2_TODO: You can add more interactions for IDW, RBF, etc.
if(ImGui::BeginPopup("SubMenuWarping"))
{
  if (ImGui::MenuItem("WarpingIDW") && p_image_)
  {
    p_image_->enable_selecting(false);
    p_image_->set_idw();
    p_image_->warping();
    p_image_->init_selections();
  }
  if (ImGui::MenuItem("WarpingRBF") && p_image_)
  {
    p_image_->enable_selecting(false);
    p_image_->set_rbf();
    p_image_->warping();
    p_image_->init_selections();
  }
  ImGui::EndPopup();
}
```

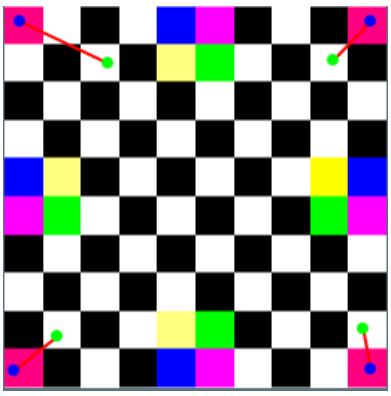
通过`set_idw()` 函数和`set_rbf()` 函数可以将信号传入 CompWarping 类中的`warping()` 函数中，具体实现是在该函数中加入如下代码：

```
switch (warping_algorithm_)
{
  case IDW:
  warping_alg_ = std::make_shared<WarpingIDW>(start_points_, end_points_);
  break;
  case RBF:
  warping_alg_ = std::make_shared<WarpingRBF>(start_points_, end_points_);
  default:
  break;
}
```

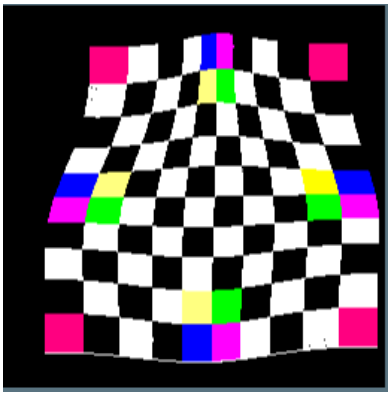
其中，`warping_algorithm` 是我们定义的枚举类型元素，用于记录图形变形算法，可以通过`set_idw()` 函数和`set_rbf()` 函数修改值。

3 实验结果

如图所示，左侧图像为所选取的锚点，右侧图像为变形后的图像。

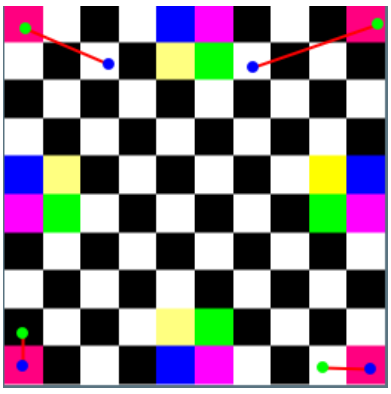


(a)

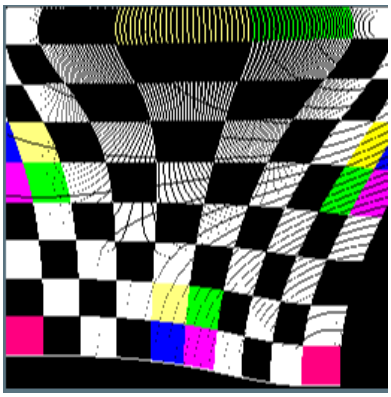


(b)

图 2: IDW 算法结果

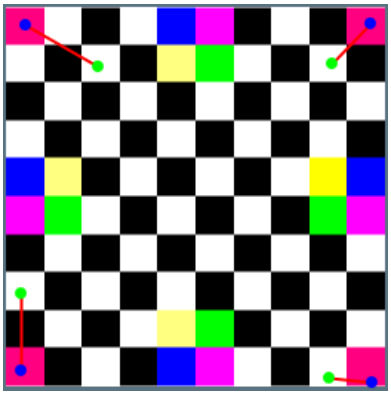


(a)

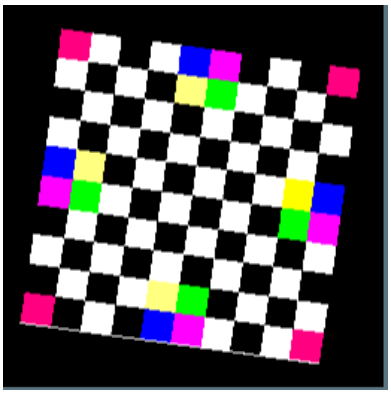


(b)

图 3: IDW 算法结果

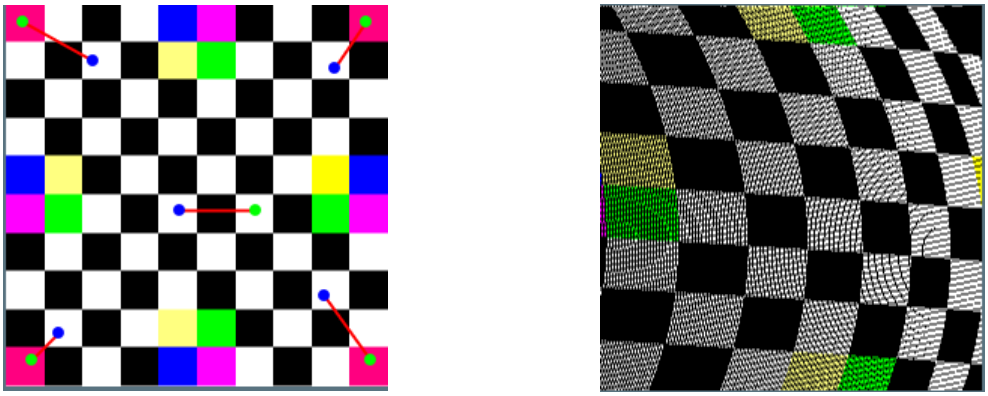


(a)



(b)

图 4: RBF 算法结果



(a) (b)

图 5: RBF 算法结果

4 结果分析

本实验中，IDW 算法由于较为简单，算法执行效率较高，而 RBF 算法更能保存图像原本的形状。另外，关于图像变形时产生的“黑缝”，这是由于我们编写的图像变形算法本质都是像素点的移动。当像素点从较小的区域变换到较大的区域时，由于像素点不足以填满整个区域，在图形映射变化较快的地方便会产生默认颜色点（在本实验中为黑点）。图形映射本身是连续的，故产生了“黑缝”。