NMS

陈烁龙

2022 年 7 月 14 日

目录

插图

表格

摘要

在进行图像处理时，当我们用特定算法提取图像中的特定对象时，往往会在局域内有多个响应对象，如图像角点的提取、图像识别物体区域、图像边缘提取等。通过 NMS 算法的处理，我们可以提取冗余的对象，得到响应最佳的对象。

关键词： NMS，非极大值抑制

1 概述

非极大值抑制 (on-Maximum Suppression，NMS)，顾名思义就是抑制不是极大值的元素，可以理解为局部最大搜索。这个局部代表的是一个邻域，邻域有两个参数可变，一是邻域的维数，二是邻域的大小。

本次实验以一维和二维 NMS 算法为例，对该算法进行了实现。

2 一维情况

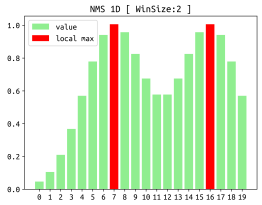
2.1 问题引出

考虑这样一种情况：我们对一张棋盘格进行角点提取，初步获取了其某个角点的信息。现在我们要对该角点的梯度方向进行计算，以供后续的优化使用。由于棋盘格网是黑白相间的，其格网点的梯度是大致垂直的，为此我们的想法是先在格网点的一定领域内获取每个像素的梯度方向，统计得到一维的条形图。而后我们便可以使用 NMS 算法来提取其两个主方向了。

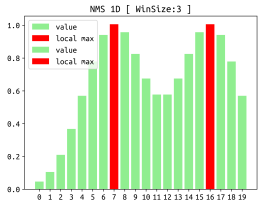
2.2 实验

本次实验通过数据模拟，获得了两组实验数据：一个基于正态分布获得的数据，一个通过均匀分布获得的数据。对两种数据都分别使用不同的窗口进行 NMS 算法，实验结果如下。

代码如下所示：

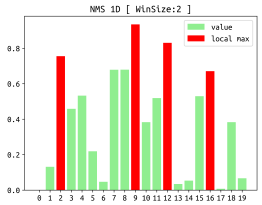


(a) 窗口半径为 2

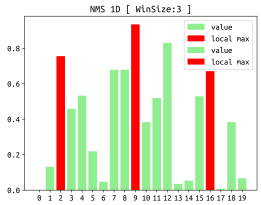


(b) 窗口半径为 3

图 1: NMS[1D]：正态分布



(a) 窗口半径为 2



(b) 窗口半径为 3

图 2: NMS[1D]：均匀分布

Listing 1: 一维 NMS

```
static std::vector<ushort> nms1d(const std::vector<float> &
    vals, const ushort hws) {
  std::vector<ushort> max;
  ushort ws = 2 * hws + 1;
  for (int i = 0; i < vals.size(); i += hws + 1) {
    float maxVal = vals[i];
    int maxIdx = i;
    for (int j = i + 1; j < std::min(i + hws + 1, int(vals.
        size())); ++j) {
      if (vals[j] > maxVal) {
        maxVal = vals[j];
        maxIdx = j;
      }
    }
    int sIdx = std::max(maxIdx - hws, 0);
    int eIdx = std::min(maxIdx + hws + 1, int(vals.size()));
    for (int k = sIdx; k < eIdx; ++k) {
      if (k == maxIdx) {
        continue;
      }
      if (vals[k] >= maxVal) {
        maxIdx = -1;
        break;
      }
    }
    if (maxIdx != -1) {
      max.push_back(maxIdx);
    }
  }
  return max;
}
```

3 一维情况

3.1 问题引出

当我们对图像进行角点检测的时候，在图像上的实际角点处的像素会响应，但不会是一个像素响应，往往是多个像素同时响应，不过响应值有高低之分。我们需要做的是筛选出局部响应最大的像素，作为后续优化的候选角点。

3.2 实验

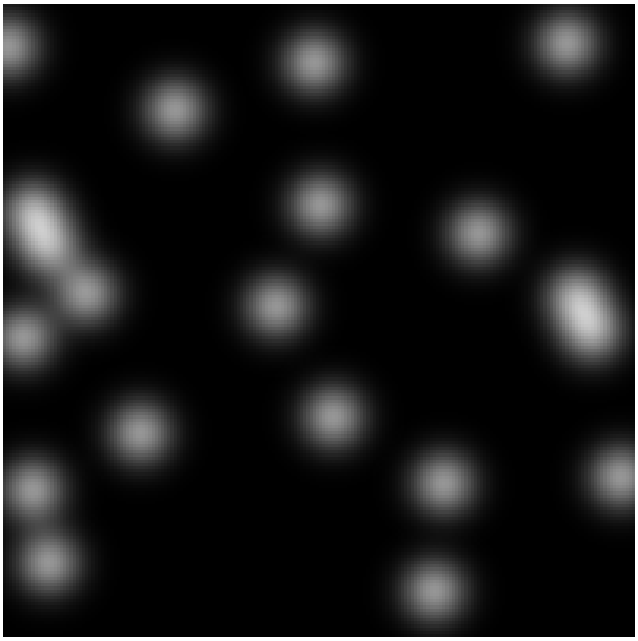
本次实验通过正太分布，随机生成了几个高值点区域。实验的目的是探测出来局部极值。代码如下所示：

Listing 2: 二维 NMS

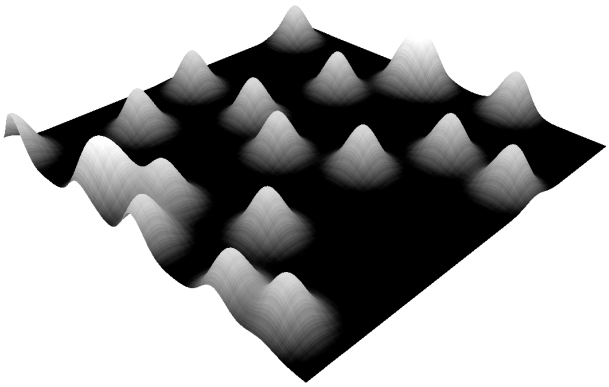
```
ushort ws = 2 * hws + 1;
for (int i = 0; i < rows; i += hws + 1) {
  for (int j = 0; j < cols; j += hws + 1) {
    // find max val and idx
    uchar maxVal;
    cv::Point2i maxIdx;
    {
      cv::Mat win = img(cv::Range(i, std::min(i + hws + 1,
          rows)), cv::Range(j, std::min(j + hws + 1, cols)
          ));
      double maxVal_t;
      int maxIdx_t[2];
      cv::minMaxIdx(win, nullptr, &maxVal_t, nullptr,
          maxIdx_t);
      maxVal = maxVal_t;
      maxIdx.y = maxIdx_t[0] + i;
      maxIdx.x = maxIdx_t[1] + j;
    }

    bool isMax = true;
    for (int r = std::max(0, maxIdx.y - hws); r < std::min
        (maxIdx.y + hws + 1, rows); ++r) {
      for (int c = std::max(0, maxIdx.x - hws); c < std::
          min(maxIdx.x + hws + 1, cols); ++c) {
        if (r == maxIdx.y && c == maxIdx.x) {
          continue;
        }
        uchar val = img.at<uchar>(r, c);
        if (val >= maxVal) {
          isMax = false;
          break;
        }
      }
      if (!isMax) {
        break;
      }
    }
    if (isMax) {
      max.push_back(maxIdx);
    }
  }
}
```

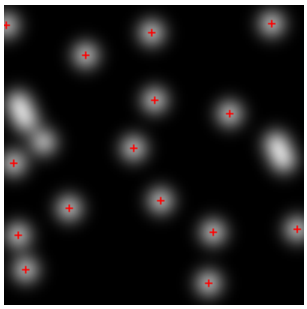
实验结果如下图所示。



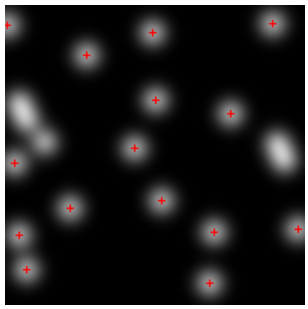
(a) 原图



(b) 三维可视化

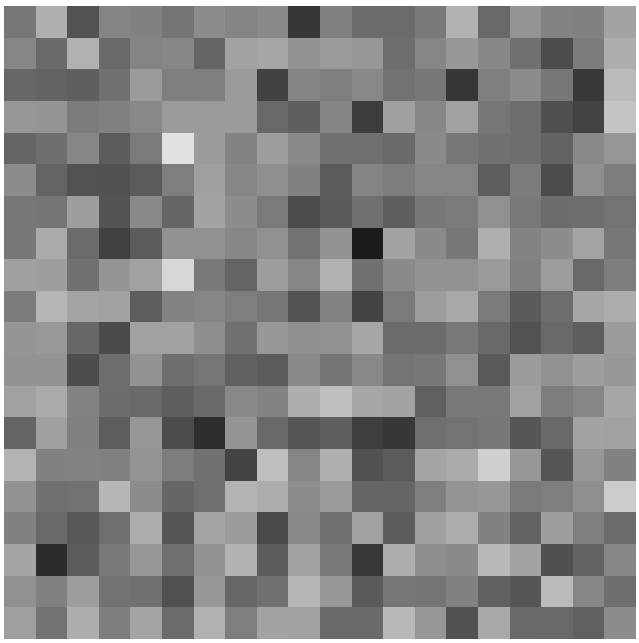


(c) 窗口半径为 2

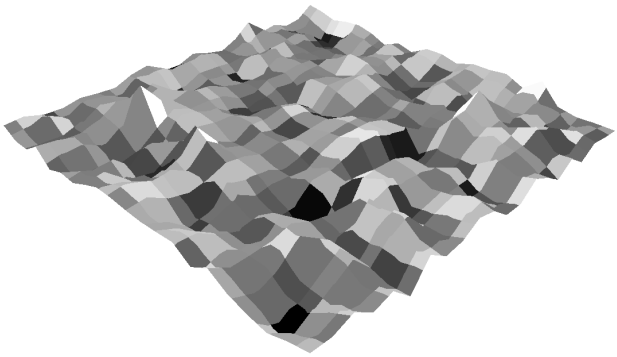


(d) 窗口半径为 3

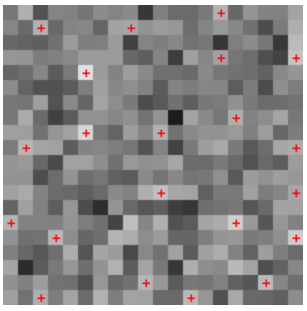
图 3: NMS[2D]：正态分布



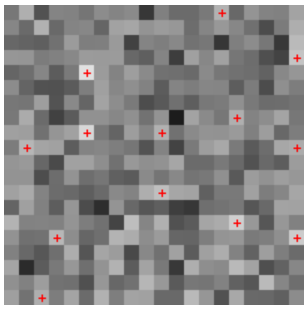
(a) 原图



(b) 三维可视化



(c) 窗口半径为 2



(d) 窗口半径为 3

图 4: NMS[2D]：均匀分布