文章编号:1001 - 9081(2005)02 - 0326 - 03

边界局部搜索算法与应用

吴贵芳,徐 科,徐金梧
(北京科技大学 机械工程学院, 北京 100083)
(Fancyboy100@ sohu. com)

摘 要:提出一种快速的边部检测算法,即边界局部搜索算法,并应用于冷轧带钢表面缺陷在线检测系统。它采用一种局部搜索的思想,在前一次搜索的基础上准确定位钢板边部,大大提高了算法的实现速度。对现场图像数据测试表明,边界局部搜索算法比传统的灰度梯度阈值算法的速度快 20 ~ 200 倍左右,并且检测的效果得到有效的提高。这种算法还可以用于其他图像边界的搜索。

关键词:边界局部搜索; 图像处理; 冷轧带钢; 表面检测
中图分类号: TP391.41 **文献标识码**:A

Local border search algorithm and its application

WU Gui-fang, XU Ke, XU Jin-wu
(School of Mechanical Engineering, Unversity of Science and Technology in Beiging, Beijing 100083, China)

Abstract: A fast border-detection algorithm named local border search algorithm was presented here, and applied to an on-line surface inspection system for cold rolled strips. Based on the last search result, a local search was used to position the cold rolled strip border and increased the algorithm's speed greatly. Test with images collected from real world data shows that local border search algorithm is about 20 to 200 times faster than the traditional gray level gradient threshold algorithm, and can detect borders more accurately. The algorithm can also be used in the search of other image border.

Key words: local border search; image processing; cold rolled strip; surface inspection

0 引言

表面缺陷在线检测技术在工业生产线上得到了越来越多的应用,但是在应用的过程中也出现一些问题,比较突出的问题是缺陷检测与识别算法的有效性和实时性。传统的表面缺陷在线检测系统由于受 CPU 处理速度的影响,往往采用硬件(或硬件 + 软件)实现算法的方式,虽然处理速度很快,但是算法的更新很困难。随着 CPU 的速度的不断提高,采用纯软件实现缺陷检测与识别算法已成为可能。但是用纯软件实现需要对算法进行最大可能的优化,以便降低算法实现的时间,对于一些速度很快的连续生产线,其算法的实时性要求就更为突出。如由于冷轧带钢生产线的生产速度一般都是很高的,往往在 900m/min ~ 1200 m/min 之间,有的甚至更高。本文以一套生产线上使用的冷轧带钢表面缺陷在线检测系统为例,研究了钢板的边部检测问题,提出了边界局部搜索算法,该算法与传统的钢板边部检测算法相比,效率明显提高,可以为后续的算法节省大量的时间,以满足实时处理的要求。

1 系统简介

图 1 所示的是一套安装在生产线上的冷轧带钢表面缺陷在线检测系统,摄像头的数量是由被检测钢板的宽度来决定的,所有摄像头将并排放置,每个摄像头所采集的区域要有适当的重合,以保证能把完整的钢板图像采集出来。整套系统的算法全部由软件来实现。摄像机采集的图像传送给并行计算系统,经过一系列的软件处理步骤将缺陷的信息提取出来,同时显示并打印出钢板的质量检查报表。由于摄像机的采集速度是 25 帧/s,平均处理每一帧图像只有 40ms 的时间,因此必须对每一个步骤的算法都尽可能的优化,以便减少算法实现的时间,满足系统在线检测的要求。

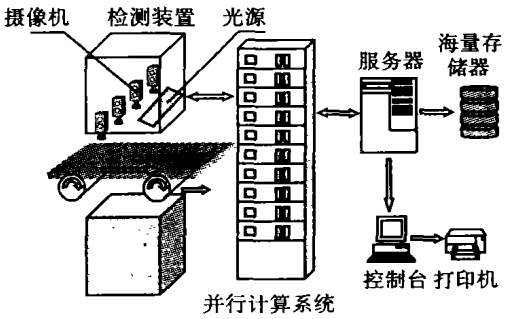
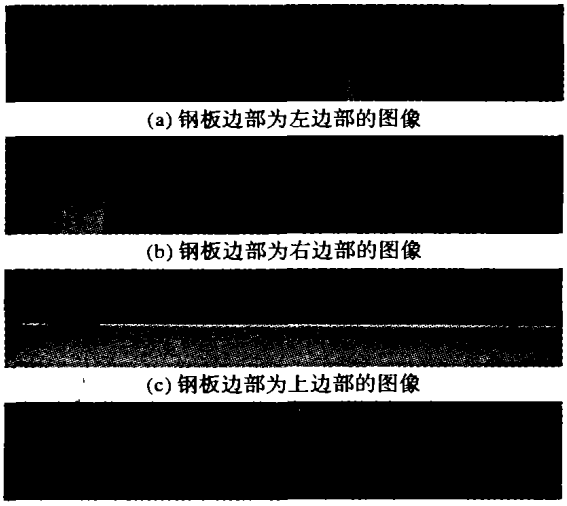


图 1 系统的总体结构图

为了对钢板进行完整检测,必须让摄像机所采集的所有区域不但可以将有钢板的部分采集出来,还必须要将钢板边部以外的一些区域也采集出来,尤其是上下表面最边上的四个摄像头。因此在算法实现过程中,首先要对所有在线采集的图像进行钢板边部的检测,钢板边部检测的主要作用是为了提取出钢

收稿日期:2004 - 07 - 26;**修订日期**:2004 - 10 - 12
基金项目:国家自然科学基金资助项目(50074010);国家 863 计划项目(2003AA331080)
作者简介:吴贵芳(1978 -),男,江西抚州人,博士研究生,主要研究方向:数字图像处理、表面检测与识别; 徐科(1972 -),男,浙江鄞县人,副研究员,主要研究方向:图像处理、模式识别中、表面检测; 徐金梧(1949 -),男,浙江宁波人,教授,博士生导师,主要研究方向:故障诊断与智能控制、数字图像处理.

板所在的区域,剔除图像中非钢板区域的图像,以避免由此造成的缺陷误检问题。如果不进行边部检测,容易把现场的一些背景及钢板的边部都作为缺陷被检测出来,并且增加了数据的处理量。钢板有四种边部类型:钢板区域在图像左部,而非钢板区域在图像右部的情况称为右边部(如图 2(a));钢板区域在图像右部,而非钢板区域在图像左部的情况称为左边部(如图 2(b));钢板区域在图像下部,而非钢板区域在图像上部的情况称为上边部(如图 2(c));钢板区域在图像上部,而非钢板区域在图像下部的情况称为下边部(如图 2(d))。另外在一些情况下还可能出现由左右与上下分别组合出来的左上边部、左下边部、右上边部以及右下边部。



(a) 钢板边部为左边部的图像

(b) 钢板边部为右边部的图像

(c) 钢板边部为上边部的图像

(d) 钢板边部为下边部的图像

图 2　钢板区域图像

2　算法原理

传统的算法大多采用灰度梯度阈值算法,它是对整幅图像进行行(或列)方向上的扫描,利用钢板边部两端的灰度值存在着较大差异,通过简单的阈值处理算法就可以得到钢板的边部。这种算法简单方便,但却存在着一些缺点,比如随着图像尺寸的增大以及图像有效区域的减小,算法的时间开销明显增加;此外,这种算法容易受一些背景奇异点的影响,检测到的边部会有错误。

根据冷轧带钢表面缺陷在线检测系统所采集图像的结果可以看出,对于钢板的边部来说,是缺陷的多发区,同时图像在没有钢板存在的区域比较暗,所以可以定制如下算法:

1) 对图像 $f(x,y)$ 以行或者列的方式进行扫描。根据不同的钢板边部类型,选择搜索的起始点与方向,如对于右边部情况,则以逐行扫描的方式从第 0 行最右边的像素开始,对横坐标递增进行;对于左边部,则以逐行扫描的方式从第 0 行最左边的像素开始,对横坐标递减进行;对于上边部,则以逐列扫描的方式从第 0 列最上边的像素开始,对纵坐标递增进行;对于下边部,则以逐列的方式从第 0 列最下面的像素开始,对纵坐标递减进行。从而使得相邻两个点的灰度值绝对值之差大于给定的某个预先设定的阈值 T。同时记下该点的纵坐标或者横坐标 $a[0]$。

2) 定义一个范围值 $TScp$,再以 $a[0]$ 这个坐标值作为下一行或者下一列的中心,分别从两边以 $TScp$ 进行区域限定,试验证明,对于钢板的边缘来说,$TScp$ 一般取 2 或者 3 就足够,因此下一行或者下一列的搜索范围就限制在 $[a[0]-TScp, a[0]+TScp]$ 之间,搜索的区间长度公为 $2TScp$。在区间内,以第一步类似的方法找到第 1 行或者第 1 列的灰度变化点坐标值,记为 $a[1]$。如果该区间中找不到符合条件的像素点,将以第 0 行或者第 0 列的值 $a[0]$ 作为当前 $a[1]$ 的值。

3) 将行或者列不断地往下推移,循环 2) 的方法。即第 $i+1$ 行或者第 $i+1$ 列的中心就是上一行或者上一列的边界坐标值 $a[i]$,从而搜索区间为 $[a[i]-TScp, a[i]+TScp]$,同时得到该行或者该列的边界坐标 $a[i+1]$。如果该区间中找不到符合条件的像素点,我们将以上一行或者上一列的值 $a[i]$ 作为当前 $a[i+1]$ 的值。直到最后一行或者最后一列搜索完毕。因此可以得到如下边界方程:

横向边界点坐标:

$$(x, a[x]),\ x = 0,1,2,\cdots,Width-1 \tag{1}$$

或者纵向边界点坐标:

$$(a[y], y),\ y = 0,1,2,\cdots,Height-1 \tag{2}$$

4) 对于单纯的上边部、下边部、左边部或者右边部的提取,到 3) 已经完成了。但对于生产现场中,会出现一些特殊的边部,比如钢板的图像处理一幅图的一个角落,即图像中出现钢板的左上边部、左下边部、右上边部或者右下边部。对于这四种情况,用原来的方法处理起来很麻烦,得经过进一步的处理才可以得到比较合理的钢板区域。但这里只要分别按前面的几个步骤将两条边部全部提取出来,为了区别,在公式(2)中,将 $a[y]$ 替换为 $b[y]$,即得到公式(3)的点坐标组:

$$\begin{aligned}&(x, a[x]),\ x = 0,1,2,\cdots,Width-1\\&(b[y], y),\ y = 0,1,2,\cdots,Height-1\end{aligned} \tag{3}$$

然后利用条件:

$$\begin{aligned}&(x, b[y]),\ x = 0,1,2,\cdots,Width-1\\&(y, a[x]),\ y = 0,1,2,\cdots,Height-1\end{aligned} \tag{4}$$

计算出两条边部线的交叉点 (X,Y),最后根据摄像机所处的位置以及钢板的进出钢信息对边部进行重新组合,从而得到钢板所处在整幅图像中的位置。

下面以纵向右边部搜索为例进行更详细的实例解释。如图 3 所示,是一幅大小 756 × 141 像素大小的图片,钢板的有效区域处于图像的左边,由于图像中钢板的边部为右边部。因此,搜索就从右边开始,对于每一行来说,不断往左推进,即对横坐标递减进行。首先第 0 行的开始搜索位置是 ,然后不断地缩小横坐标,直到找到一个点的横坐标 $a[0]$,使得 $|f(a[0],0)-f(a[0]+1,0)|>T$,同时记录 $a[0]$ 的值,即为第 0 行的边部横坐标值。现在我们取 $TScp$ 为 3,因此,对于第 1 行的搜索范围就确定为 $[a[0]+3, a[0]-3]$,是一个递减区域,总共搜索区间长度为 6 个像素点。于是我们最多只要 6 次就可以在这个区间里找到一个点 $(a[1],1)$,使得 $|f(a[1],1)-f(a[1]+1,1)|>T$。如果没有找到,就让 $a[1]=a[0]$。然后以同样的原理,对以后的每一行做出相应的处理,依次可以得到 $a[2]$、$a[3]$、…、$a[139]$、$a[140]$ 的值。在 C++ 中的具体实现代码如下:

```
for (int y = 2; y < m_iSrcHeight - 2; y++)
//为了看得清楚,一般从第二行开始扫描
{
  m_pBorderValueVerz[y - 2] = m_iSrcWidth - 1;
  //将边界值初始化为最右边的坐标值
  pSrc = m_pSrcBuffer + (y + 1) * m_iSrcWidth - 1;
```

```
if (y == 2)                                    //搜索第一个边界点
{
 for (int x = m_iSrcWidth - 1; x >0 ; x--)
 {
  int iTemp = *pSrc >= *(pSrc-1) ? *pSrc - *(pSrc-
   1): *(pSrc-1) - *pSrc;
  if (iTemp > m_iThreshold)
  {
   m_pBorderValueVerz[y - 2] = x - 1;
   break;
  }
  pSrc = m_pSrcBuffer + y * m_iSrcWidth + x - 1;
 }
}
else                                           //依次搜索下面的各个边界点
{
 m_bFinded = false;
 m_pBorderValueVerz[y - 2] = m_pBorderValueVerz[y - 3];
 if (m_pBorderValueVerz[y - 2] - m_iLookScope < 0 ||
  m_pBorderValueVerz[y - 2] + m_iLookScope > m_
  iSrcWidth - 1)
  continue;
 pSrc = m_pSrcBuffer + y * m_iSrcWidth + m_
  pBorderValueVerz[y - 2] + m_iLookScope;
 for (int x = m_iLookScope; x >= -m_iLookScope; x--,
 pSrc--)
 {
  int iTemp = *pSrc >= *(pSrc-1) ? *pSrc - *(pSrc-
   1): *(pSrc-1) - *pSrc;
  if (iTemp > m_iThreshold)
  {
   m_pBorderValueVerz[y - 2] = m_pBorderValueVerz[y - 2]
     + x - 1;
   m_bFinded = true;
   break;
  }
 }
}
}
```

处理之后的结果如图 3 所示。



图 3 钢板右边部的检测结果

在整个算法的实际应用中,图像的边缘处必须得加以进一步的处理。比如上面横向右边部的情况下,当检测出来的第 0 行的 $a[0]$ 小于 3 或者大于 137 时,运用上述策略,下一行的检测区间将会出现小于 0 或者大于 140 的点坐标,这样将超出图像的边界。为了解决这个问题,可以根据图像的实际情况,一般将边部去掉三个像素点左右,即不从最边部的像素开始检测,这样就很好地避免了这种情况的发生。

3 试验结果

本文对系统采集到大量钢板图像进行了测试,图片的大小为 756 × 141 像素。如图 4 所示。

由图 4(a)可以看出,边界局部搜索算法所检测到的边部很准确,而灰度梯度阈值算法所检测的边部存在着误差(见图 4(b))。因此,边界局部搜索算法的检测效果要优于传统的灰度梯度阈值算法。从表 1 中可以看出边界局部搜索算法在提取速度上较灰度梯度算法具有明显的优越性。

表 1 右边部图像进行边部提取数据表

样图号	钢板边界与图像边界的距离	灰度梯度阈值算法所用时间 T1(ms)	边界局部搜索算法所用时间 T2(ms)	T1/T2(倍)
图 3	大于 600 像素	2.0 ~ 2.2	0.01 ~ 0.03	67 ~ 220
图 5(a)	小于 200 像素	0.65 ~ 0.75	0.01 ~ 0.03	22 ~ 75

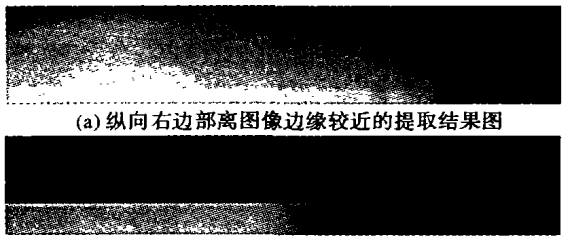


(a) 边界局部搜索算法检测的边部

(b) 灰度梯度阈值算法检测的边部

图 4 搜索算法和阈值算法检测

图 5 是对上面的检测结果,从图中可以看到检测效果非常好,得出了钢板所在图像中的精确位置。如果用灰度梯度阈值算法,也必须首先分别得出两条边部,但是这两条边部将是很杂乱的,因此要再进一步得出钢板所在边部的具体位置将会相当麻烦,而且效果会相当差。



(a) 纵向右边部离图像边缘较近的提取结果图

(b) 对右上边部离图像边缘较近的提取结果图

图 5 钢板区域提取结果图

4 结语

本文介绍了一种边部的快速检测算法,即边界局部搜索算法。该算法与传统的灰度梯度阈值算法相比,不仅实现的速度快,而且检测的效果好。从冷轧带钢表面缺陷在线检测系统的实际使用来看,它能在很大程度减少在线检测过程中图像的预处理时间,从而为后续的算法提供了更多的 CPU 计算时间,使这套在线检测系统从整体性能上得到了很大的改进。边界局部搜索算法不仅仅可以用于表面在线检测系统中钢板图像有效区域的确定问题,还可以通过调整 TSpc 参数,用来对图像中的直线或者曲线的快速提取。同时随着图像尺寸大小的增加,边界局部搜索算法的优越性更能得到充分的体现。

参考文献:

[1] 徐科, 徐金梧, 鹿守理, 等. 冷轧带钢表面自动监测系统的研究[J]. 钢铁, 2000, 35(10): 63 - 66.

[2] 吴平川, 路同浚, 王炎. 机器视觉与钢板表面缺陷的无损检测[J]. 无损检测, 2000, 22(1): 13 - 16.

[3] VENKATESWAR V, CHELLAPPA R. Extracting straight lines in aerial images[J]. IEEE Trans Pattern Anal Mach Intell, 1992, 14(1): 111 - 114.

[4] EVERDGE JR, RISEMAN EM. How easy is matching 2D line models using local search [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1997, 19(6): 56 - 579.

[5] 曹莉华. 图像边缘提取中的一种动态阈值获取法[J]. 小型微型计算机系统, 1997, 18(7): 53 - 56.