基于 SIFT 算法的大场景视频拼接算法及优化

杨思燕[1] 贺国旗[1] 刘如意[2]
(陕西广播电视大学信息与智能技术学院 西安 710119)[1]
(西安电子科技大学计算机科学与技术学院 西安 710071)[2]

摘 要 目前大量的由独立视频设备获取的小场景视频信息难以满足大场景下信息处理的要求,而通过多设备人工查阅的方式又存在效率低下、信息冗余和碎片化等问题。文中研究了大场景视频拼接技术,利用 SIFT 算法的尺度不变特性对关键点进行特征匹配,通过仿射矩阵变形完成对图像的拼接工作。在此过程中,对传统的 SIFT 拼接算法进行进一步的优化,主要是基于距离的优化算法来完善视频拼接的效果;对 SIFT 特征点匹配、加权优化算法、关键帧提取的技术等进行并行加速,以提高拼接效率。实验结果表明,提出的优化方法能更好地提取视频中的关键信息,以实现更好的视频拼接效果。在视觉效果上,所提方法得到的拼接结果中不存在传统方法出现的两幅图像的交接线。此外,在 MATLAB 环境下分别对关键点检测和拼接部分进行了加速优化,优化后的关键点检测效率提高了约 20%,拼接部分的效率提高了将近 57%。在 C++环境下,关键点的检测效率提升了 14%,拼接部分的检测效率提升了 40%。

关键词 视频拼接,SIFT 特征点匹配,加权优化算法,关键帧,并行优化

中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.07.044

Video Stitching Algorithm Based on SIFT and Its Optimization

YANG Si-yan[1] HE Guo-qi[1] LIU Ru-yi[2]
(School of Information and Intelligence Technology, Shaanxi Radio & TV University, Xi'an 710119, China)[1]
(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)[2]

Abstract At present, a large number of video information obtained by independent video devices in small scenes cannot meet the requirements of information processing in large scenes, and the manual access by multiple devices is faced with such problems as low efficiency, information redundancy and fragmentation. This paper investigated large scene video stitching technology, carried out the feature matching of key points by using the scale-invariant feature of SIFT algorithm, and completed the image splicing by affine matrix deformation in this process. The traditional SIFT splicing algorithm is further optimized, mainly based on distance optimization algorithm to improve the effect of video splicing. In order to improve the efficiency of stitching, the techniques of key frame extraction based on SIFT feature point matching weighted optimization algorithm were accelerated in parallel. Experiments show that the proposed optimization method can extract essential information in the video more effectively, achieving better video stitching results. Visually, the intersection of the two images appeared in the results of the conventional method does not appear in the stitching results obtained by the proposed method. Moreover, the key point detection and splicing parts were accelerated and optimized respectively in the different environment. In MATLAB, the efficiency of the key point with optimized is improved by 20%, and that of splicing parts is increased by about 57%. In C++, the efficiency of the key point is improved by 14%, and that of splicing parts is increased by about 40%.

Keywords Video Stitching, SIFT feature point matching, Weighted optimization algorithm, Key frame, Parallel optimization

1 引言

视频和图像作为一种信息的表达形式,具有形象具体的特点,可以准确地传递客观世界的现实信息,人类可以非常直观地获取其中的信息[1]。但目前在很多场景下,基本都是利用单个成像设备来获取单个视角的视频信息,不利于在大场景下进行目标检测、识别以及追踪等任务。虽然可采用调整焦距或采用鱼眼镜头的方式来增加单个摄像头的覆盖范围[2],但是大场景下目标的分辨率相对较低,而鱼眼镜头在视觉效果上并不友好。因此,利用视频拼接技术[3]将由多个摄

到稿日期:2018-07-25 返修日期:2018-10-13 本文受陕西省教育科学规划课题(SGH16V022),陕西广播电视大学 2017 年度科研重点课题(17D-08-A06)资助。

杨思燕(1976—),女,硕士,副教授,主要研究方向为图形图像处理、大数据研究应用,E-mail:siyanyang@126.com(通信作者);**贺国旗**(1968—),教授,主要研究方向为网络技术、图形图像、教育技术应用;**刘如意**(1989—),讲师,主要研究方向为计算机视觉、遥感图像处理。

像设备获得的场景进行拼接以获得大场景的视频数据将是一种可行的解决方案。

目前，关于图像拼接技术已有较多的研究成果。例如，针对 As-Projective-As-Possible(APAP)算法采用均匀分块法来实现图像拼接的局部映射，没有充分考虑图像不同区域之间的差异性的问题，王元炜等提出了一种基于特征点分布的图像映射算法[4]。但该方法存在图像形变以及视差容忍度的问题。针对待拼接图片重叠区域的视差具有一定的容忍性，并且能够适应更复杂的图像拼接场景。何川等提出了一种具有直线结构保护的图像拼接算法(MISwLP)[5]，该算法通过提取图片中的直线结构并施加约束，可以得到视觉效果自然、畸变较小的图像拼接结果，但是对拼接图像中的全局直线结构有待进一步深入研究。针对来自于不同视点拍摄的具有视差的两幅图像，周雪等提出一种基于特征点匹配对平面相似性的图像拼接方法[6]。目前，基于尺度不变特征变换(SIFT)算法的图像拼接技术研究得较多[7-8]。刘文亮等则将 LBP 特征引入 SIFT 特征点描述中以进行特征点提取[9]，在描述特征点时，采用隔点计算特征点周围像素点的旋转不变均匀 LBP 特征值，并用其替代一阶梯度幅值，生成改进的 SIFT 特征点描述向量。而陆园园等针对红外图像拼接误匹配点过多、耗时过长等问题，对基于 SIFT 算法的红外图像拼接方法进行改进[10-12]。为加快图像拼接的效率，相应的加速算法被提出了。陈月等先后提出了一种基于 SIFT 特征矢量图的快速图像拼接方法[10]和一种图像局部特征自适应的快速尺度不变特征变换拼接方法[13]。赵岩等提出了一种结合投影误差校正的快速 SIFT 图像拼接方法[14]。

目前视频拼接技术的应用前景十分广阔，不论是虚拟现实技术还是智能城市系统都需要视频拼接技术作为基础来实现[15-17]。针对多视角视频图像拼接因角度因素容易产生形状失真、深度失真和运动失真等问题，刘娟等对现有视频图像拼接方法加入角度约束以进行改进[18]，并将拼接后的静止区域和运动区域融合成宽视野的视频图像。与图像拼接方法类似，视频拼接研究也存在实时性要求的问题。为此，李勇等提出一种利用小区域融合和查表映射实现实时视频拼接的方法[19]，该方法具有更好的拼接效果和更高的效率。而杨晓萍等提出了一种基于 FPGA(Field Programmable Gate Array)的车载视频拼接方法[20]。此外，常嘉义等针对图像和视频拼接方法的性能提出了一套质量评价指标[21]。

可见，目前针对图像和视频都已有较多的研究成果，而研究较为广泛的主要是基于 SIFT 算法的图像或视频拼接技术。但是，这些方法在针对分辨率较高的图像或视频进行拼接时都存在效率欠佳的问题。本文主要针对视频拼接过程中存在的时效性问题提出相应的优化加速算法，并且对拼接方法进行优化，以去除拼接效果图中存在的拼接痕迹。

2　传统的基于 SIFT 算法的视频拼接方法

David Lowe 教授于 1999 年首次发表了 SIFT 算法，并于 2004 年对其进行完善和总结[22]。SIFT 算法可以保证特征点自身不会因为其他外在因素的变化而变化，例如光照、仿射变换和噪声等因素，从而达到基于特征点完成图像拼接的目的。其主要包括两个部分：特征点检测与图像融合。

2.1　特征点检测

基于 SIFT 的图像拼接算法中，可以通过改变空间尺度的参数找到不同的特征点，这代表着不同尺度下的图像有不同的特征点。在不同尺度空间选取的特征点具有一定的特殊性，这类特征点满足视觉不变性。空间尺度 $L(x,y,\sigma)$ 的表达式为：

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{(x-\frac{m}{2})^2+(y-\frac{n}{2})^2}{2\sigma^2}} \tag{1}$$

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y) \tag{2}$$

其中，$G(x,y,\sigma)$ 为高斯函数，$I(x,y)$ 为原图像矩阵，σ 是尺度空间因子。尺度空间的极值点(特征点)就是在高斯差分金字塔中检测的特征点，可以在不同尺度下比较图像之间的差别以寻找极值，从而得到不同尺度下的特征点。但是，产生的极值点并非稳定的特征点，其中夹杂着不少伪极值点。为了保证特征点的匹配效果，还需要对现有特征点进行筛选和优化。

2.2　特征点匹配与图像融合

基于检测得到的不同尺度下的特征点，求得两幅图像或者视频帧的最优投影变换矩阵，以确定待拼接的图像或帧之间的空间关系。为了得到最终的合成图像，还需要选择合适的图像融合方法来完成图像或者视频帧的拼接，以得到一幅无缝的拼接图像。这一过程一般可分为两个步骤：1)图像合并，即利用检测的特征点及变换矩阵参数，将两幅图像拼接到同一个坐标空间内；2)利用合适的图像融合算法将两幅图像或视频帧融合成一幅图像。

但是传统的基于 SIFT 算法的图像拼接结果普遍存在效率低以及效果欠佳的问题，特别是针对具有较高分辨率的视频进行拼接时，时效性问题尤为突出。为此，本文提出了相应的方法对视频拼接过程中关键帧的选择以及算法加速性能进行优化，以提高视频拼接的效率，并改善视频拼接后的视觉效果。

3　基于 SIFT 算法的视频拼接算法的优化

3.1　基于距离加权优化算法

针对摄像设备以及光线等其他因素在图像拼接结果图中的交界处会出现明显的差异，导致在拼接后的图像中存在较为明显的拼接痕迹或断裂感的问题，考虑算法性能和通用性等因素，本文提出一种具有较低计算复杂度并能够较大提高拼接图像效果的距离加权算法。

如图 1 所示，该方法根据特征点在重叠部分中的对应位置引入一个权值系数 α，找到起始点并将其横坐标记为 P_s，计算出重叠部分区域的长度 L_{pw}，再根据不同像素点的位置得到对应的权值系数 $\alpha_{i,j}$。

$$\alpha_{i,j}=1-\frac{j-P_s}{L_{pw}} \tag{3}$$

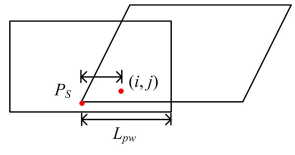

图 1　距离加权优化算法

Fig. 1　Distance weighting optimization algorithm

得到权值系数 $\alpha_{i,j}$ 后，计算图像 P 和 T 的最终重叠部分的三通道数据 $F_{i,j}$。

$$F_{i,j}=P_{i,j}\times\alpha_{i,j}+T_{i,j}\times(1-\alpha_{i,j}) \tag{4}$$

然后以最终 $F_{i,j}$ 三通道数据还原重叠部分的图形，实现拼接部分的有效优化。

3.2 基于帧间差异信息的关键帧提取

为提高视频拼接的效率，提取视频中的关键帧。利用灰度信息进行关键帧的提取，将 RGB 信息转化为灰度信息进行处理。只考虑视频的灰度信息可以极大地提升算法的效率。

定义 $r(S_m,S_n)$ 为第 m 帧与第 n 帧之间的相关函数：

$$r(S_m,S_n)=\sum_{i=1}^{i=M}S_m(i)S_n(i) \tag{5}$$

其中，M 为像素点的总个数，i 为该帧中第 i 个像素点。m 帧的自相关函数 $r(S_m)$ 的定义为：

$$r(S_m)=\sum_{i=1}^{i=M}S_m^2(i) \tag{6}$$

因此，两帧之间越相似，则互相关性越大，反之越小[23]。

然后依据相关性系数之间的差异来判定视频中的关键帧。判断帧 S_i 是否为关键帧的关键是比较 S_i 与当前关键帧 S_k 之间的相关函数与 S_k 的自相关函数之间的差值是否大于某个阈值，表达式为：

$$\Delta r=|r(S_k,S_i)-r(S_k)| \tag{7}$$

$$\frac{\Delta r}{r(S_k)}-\theta>0 \tag{8}$$

其中，Δr 为当前帧与关键帧之间的差值，θ 为自定义阈值。不同的取值会直接导致抽取的关键帧的结果不同，可以根据不同的实验环境和需求进行合理的调整。若式(8)成立，则当前帧为关键帧，此时输出关键帧。

为避免提取到冗余的关键帧，在相关性判别的过程中使用了滑动窗口机制，这同时可以提高程序的运行效率。例如：假设滑动窗口的宽度 $W=4$，而窗口内的第二帧图像的 Δr 为窗口内最大值，那么对窗口内其他图像就不再进行关键帧判断，仅对第二帧进行判断。若不是关键帧，则窗口向后滑动 4 个帧位；若是关键帧，则更新关键帧信息，窗口向后滑动 2 个帧位。程序继续重复上述步骤，直到视频中所有帧都查找完毕。

完整方法的流程描述如下：

1)获取视频流的总帧数信息 N；

2)对关键帧进行初始化，将第一帧初始化为第一个关键帧 $S_k=S_0$，同时计算其自相关系数 $r(S_k)$；

3)判断视频流是否遍历完成，如果 $i>N$，则结束，否则，转至步骤 4)；

4)滑动窗口宽度为 W，依次计算滑动窗口内各帧图像与上一个关键帧的相关系数 $r(S_k,S_i)(i=i+1,i+2\cdots i+w-1,i+w)$，同时计算差值 $\Delta r=|r(S_k,S_i)-r(S_k)|$；

5)若 $\Delta r/r(S_k)-\theta>0$，则该帧为关键帧，更新关键帧信息 $S_k=S_i$，并输出当前关键帧信息，否则转至 6)；

6)帧索引 $i=i+1$，转至 3)。

3.3 基于聚类的关键帧提取

本文提出的另一种关键帧提取方法是基于聚类方法的。该算法对各个视频帧进行聚类处理，再根据聚类结果找到距离聚类中心最近的帧。因此，聚类算法的关键帧是在程序结束时得到的。但与其他方法相比，聚类算法又有其独特性，所得到的特征信息往往具有全局性。

在该方法中，我们先进行视频帧的色彩空间转换。与 RGB 模型相比，HSV 模型更加接近人的视觉系统，人眼能够更加直观、容易地感受颜色的变化。因此，将视频帧的色彩信息从 RGB 空间转换到 HSV 空间。

在 HSV 色彩空间中，首先根据不同色度将 H 均等划分为 12 个量级，将 S,V 均等划分为 5 个量级，再把原来在 0 ～ 255 范围的 RGB 颜色映射到 $12\times5\times5$ 的范围。然后，对于大小为 $M\times N$ 的图像，分别统计 H,S,V 分量中各个分量区域所占的比例。

$$H(i)=\frac{H\_follow(i)}{M\times N} \tag{9}$$

$$S(j)=\frac{H\_follow(j)}{M\times N} \tag{10}$$

$$V(k)=\frac{H\_follow(k)}{M\times N} \tag{11}$$

可以通过相同 HSV 区域的占比来比较两幅图像的相似度。将各个 HSV 空间中对应区域的最小百分比值相加，即为两视频帧之间的相似度信息。具体计算方法如下：

$$S_H=\sum_{i=1}^{12}\min(H(i),Shot\_H(i)) \tag{12}$$

$$S_S=\sum_{i=1}^{5}\min(S(j),Shot\_S(j)) \tag{13}$$

$$S_V=\sum_{k=1}^{5}\min(V(k),Shot\_V(k)) \tag{14}$$

由于人眼对各个 HSV 分量的敏感度不同，因此可以依据各个分量的影响程度对各分量数据进行加权。最终的加权公式为：

$$F=0.5\times S_H+0.3\times S_S+0.2\times S_V \tag{15}$$

具体计算步骤如下：

1)对程序进行初始化，创建第一个类和聚类中心。分别对当前帧进行相似度判别，维护各个聚类的中心点信息：

$$Shot\_H(i)=\frac{H_{next}+\sum_{n=1}^{Shot.length}H_n(i)}{Shot.length+1}(Shot\_H(1),\cdots,Shot\_H(12))$$

$$Shot\_S(j)=\frac{S_{next}+\sum_{n=j}^{Shot.length}S_n(j)}{Shot.length+1}(Shot\_S(1),\cdots,Shot\_S(5))$$

$$Shot\_V(k)=\frac{V_{next}+\sum_{n=1}^{Shot.length}V_n(k)}{Shot.length+1}(Shot\_V(1),\cdots,Shot\_V(5))$$

$$Shot.length=Shot.length+1 \tag{16}$$

2)计算当前帧与各个聚类质心的相似度，如果该帧与聚类质心的相似度大于某一阈值，表示该帧为该聚类的关键帧，更新该聚类质心信息，否则创建新型的聚类。

3)计算出各个聚类中信息熵最大的图像，将该图像作为该聚类的关键帧：

$$E=-\sum_{i=1}^{12}H(i)\log H(i)-\sum_{j=1}^{5}S(j)\log S(j)-\sum_{i=1}^{5}V(k)\log V(k) \tag{17}$$

3.4　并行加速

除了算法优化方法外，本文在算法实现过程中也考虑了效率的问题。并行加速是利用并行计算技术对现有程序进行加速的一种手段，可以分为时间并行和空间并行两类。本文利用了 NVIDIA 公司针对 GPU 通用计算推出的统一计算设备架构——CUDA[24]。复杂的事件和逻辑处理由 CPU 处理，而 GPU 负责接收并处理 CPU 发送的大量高度线程化的并行任务。整个程序的运行可以分为 3 个主要步骤：1）数据事件的处理和调度由 CPU 完成，CPU 将计算任务分配给 GPU 进行计算；2）GPU 完成内核函数的执行；3）数据回传，CPU 进行调度和整理。

4　实验与分析

4.1　拼接效果的实验分析

本节主要通过对 SIFT 算法的拼接效果与算法优化后的效果进行对比分析，来验证所提优化方法的有效性。

4.1.1　基于传统的 SIFT 算法的拼接效果

基于 SIFT 算法的图像拼接原理来实现视频的拼接。本文彩色视频的分辨率为 1 288×724，帧分辨率较高，能较好地表现场景中的整体和细节信息，能检测到较多的特征点，有利于提取更多的特征点，以保证图像的拼接效果。其中的两个视频帧如图 2 所示。在拼接过程中共检测到 2 252 对可匹配特征点，根据匹配特征点的欧氏距离筛选出较优的 52 对匹配特征点，对应关系如图 3 所示。

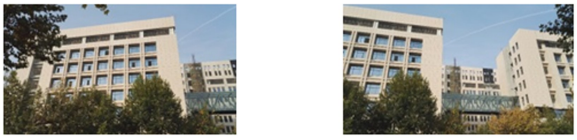

图 2　原图像

Fig. 2　Original images

图 3　SIFT 特征点匹配

Fig. 3　SIFT feature points matching

根据特征点的对应关系得到两图像之间的仿射矩阵。通过对图 2 的仿射变形，顺利得到最终的拼接结果，如图 4 所示。从图 4 中可以看出，基于传统的 SIFT 算法的图像拼接方法能够完成视频拼接，但是拼接结果中存在较为明显的拼接缝。

图 4　基于 SIFT 算法的拼接结果

Fig. 4　Image mosaic result based on SIFT

4.1.2　基于距离加权优化算法优化后的拼接效果

针对基于 SIFT 算法得到的视频拼接结果中存在较为明显的白边和拼接痕迹的问题，对于拼接处的周边像素点，本文根据其到拼接中心的欧氏距离进行加权计算，以实现对拼接痕迹的优化处理。优化后的结果如图 5 所示。

图 5　基于距离加权优化算法的拼接结果

Fig. 5　Image mosaic result based on weighting optimization

从图 5 可以看出优化后的拼接方法能够有效处理拼接痕迹，较为理想地平滑了拼接中心附近的断裂和毛糙，并且算法的复杂度较低，效率高。

4.2　基于关键帧提取的实验分析

4.2.1　基于帧间差异性信息的关键帧提取

本实验以一段大小为 22.5 MB、时长 15 s 的视频为例，对视频进行基于帧间差异最小的关键帧提取，结果如表 1 所列。已知视频帧率为 30 帧/s，因此该视频的总帧数为 450 帧。当阈值 θ 大于 0.8% 时，不能有效提取出较多的关键帧；当阈值 θ 为 0.7%时，可以有效地提取相关联的 17 个关键帧；当 θ 小于 0.7%时，算法灵敏度很高，会判断出过多的关键帧。由于在固定场景下视频具有比较强的连贯性，17 个关键帧不仅能解决信息冗余的问题，还可以保证提取的关键帧信息具有较强的连贯性。

表 1　基于帧间差法的关键帧提取实验结果

Table 1　Key frame extraction results based on inter-frame difference

阈值 θ/%	提取关键帧数	有效关键帧数
0.1	307	15
0.2	201	15
0.3	127	15
0.4	68	15
0.5	43	15
0.6	29	15
0.7	17	15
0.8	6	6
0.9	4	4
1.0	1	1

根据不同的应用场景，可以选择不同的阈值系数来实现不同的判别标准；对于不同的视频信息，采用相同的阈值也可能提取出完全不同的关键帧个数。阈值 θ 的选取标准应该视不同的实验环境来确定。

4.2.2　基于聚类的关键帧提取

由前文可知，在基于聚类的关键帧提取算法中，优先计算各个视频帧的相似度信息，再根据对应的相似度信息确定该视频帧的归属。这里同样涉及阈值 δ 的选择，不同的阈值 θ 选取对实现结果有明显的影响。本节同样以大小为 22.5 MB、时长 15 s 的视频流信息为例，对视频进行基于聚类的关键帧提取，结果如表 2 所列。

表 2 基于聚类的关键帧提取实验结果

Table 2 Key frame extraction results based on clustering

阈值 δ/%	提取关键帧数	有效关键帧数
95	257	15
92	184	15
89	110	15
86	51	15
83	23	15
80	15	15
77	9	9
74	5	5
71	3	3
68	1	1

在阈值 δ 大于 80% 时，算法判断出了过多的冗余关键帧，但是随着阈值的变化关键帧数量急剧减少。而阈值 δ 小于 80%时，算法判断的关键帧个数少于我们所需要的关键帧个数，随着阈值的变化，虽然关键帧数量减少，但是幅度明显放缓。

综上可知，帧间差法根据前后帧之间的相似度差别来进行关键帧的提取，因此该算法比较注重对视频信息连贯性变化的监测，提取出的关键帧也具有较强的连贯性，对视频镜头信息的变化比较敏感。而聚类算法则注重视频信息的整体性，根据聚类结果来获取关键帧。该算法对视频信息的整体概括性比较强，提取出的关键帧的内容有很强的代表性，对阈值信息的变化比较敏感。由此可见，两种算法各有优劣，应视实验环境和需求选取相关算法及阈值。

4.3 视频拼接效率的对比分析

4.3.1 基于 MATLAB 的并行加速实验

首先在 MATLAB 平台下完成视频拼接算法的实现，实验环境为 2014a 版本。测试数据如表 3 所列。从表 3 可以看出：不管是对原程序还是优化后的程序来说，关键点检测都占据了程序大部分的运行时间。就关键点检测而言，优化后的运行效率提高了大约 20%，而拼接部分的优化效率则提高了将近 57%。就并行加速而言，关键点的检测效率提升并不明显，但是对于拼接这种可以进行并行计算的算法来说，并行计算的效率具有较大的提升空间。总的来说，程序优化后运行时间明显缩短，由原来的 4.567 s 缩短至 3.215 s，幅度达到 30%。这说明优化有效提高了程序的总体运行效率，如图 6 所示。

表 3 MATLAB 原程序与优化程序的运行效率

Table 3 Efficiency of MATLAB program and optimization program

MATLAB	原程序时长/s	优化程序时长/s	优化占比
关键点检测	3.1969	2.6156	0.818168
拼接	1.3701	0.5994	0.437486

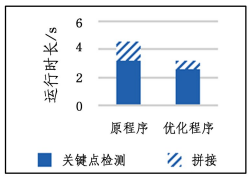

图 6 MATLAB 原程序与优化程序的运行效率比较

Fig. 6 Efficiency comparison of MATLAB program and optimization program

4.3.2 基于 C++和 CUDA 的 GPU 并行加速实验

此外，我们还利用 C++与 CUDA 联合编程实现视频拼接的 C++程序的 GPU 加速实验，实验环境为 visual studio2017、openCV2.4.9 和 CUDA，测试结果如表 4 所列。从表 4 可以看出：对于 C++程序而言，关键点检测的运行依然占据了程序运行总时间的 70%。分别对拼接和检测算法进行 GPU 加速后，程序效率有较明显的提升，其中检测部分的效率提升 14%，拼接部分提升 40%。整体的程序运行时长由 3.069 s 降至 2.423 s，总时间缩短 12%。这说明对原程序进行并行的优化算法可以有效地缩短程序运行的总时长，如图 7 所示。

表 4 C++原程序与优化程序的运行效率

Table 4 Efficiency of C++ program and optimization program

C++	原程序时长/s	优化程序时长/s	优化占比
关键点检测	2.1483	1.8572	0.864498
拼接	0.9207	0.5658	0.614532

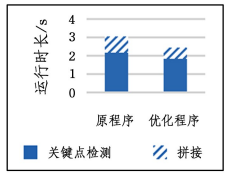

图 7 C++原程序与优化程序的运行效率比较

Fig. 7 Efficiency comparison of C++ program and optimization program

进一步对比分析两个不同平台下优化方法对视频拼接效率提升的差异。研究结果表明，C++程序优化后运行效率的提升明显优于基于 MATLAB 程序优化后运行效率的提升。虽然 C++程序的优化幅度没有 MATLAB 程序大，但是后者程序本身的运行效率较低，如图 8 所示。

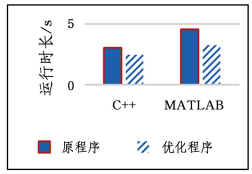

图 8 C++与 MATLAB 的运行效率比较

Fig. 8 Efficiency comparison of C++ program and MATLAB program

就优化效率而言，基于 MATLAB 平台优化后视频拼接效率提升 30%，基于 C++优化后效率提升只有 22%。但就最终程序的运行效率来说，C++程序远比 MATLAB 程序快。其原因在于：1)C++对处理视频帧信息的库函数性能比 MATLAB 优秀，openCV 是 C++库中图像领域内比较的成熟类库；而 MATLAB 中并没有相应的成熟类库；2)MATLAB 是解释型语言，C++为编译型语言，就语言本身而言，C++的运行速度优于 MATLAB，特别是在大数据量处理的情况下，C++的优势尤为明显。

结束语 针对目前视频拼接过程中对拼接效率和拼接效果等方面提出的较高要求，提出基于距离的优化算法以完善视频拼接的效果；同时提出基于帧间差异信息以及帧信息距离的关键帧提取方法等，以提高拼接效率；进一步，分别基于 MATLAB 和 C++对关键点检测和拼接部分进行了加速优化，以进一步提高拼接效率。实验表明，提出的优化方法能更

好地改善视频拼接的效果，能够很好地处理拼接痕迹的问题，并且提出的关键帧提取和并行加速方法都能够在较大程度上提高拼接效率。

参 考 文 献

[1] ZHANG J，CHEN G，JIA Z. An image stitching algorithm based on histogram matching and SIFT algorithm [J]. International Journal of Pattern Recognition and Artificial Intelligence，2017，31(4)：1754006.

[2] ZHENG G L，TANG B B. The stereo vision positioning system based on multiple fisheye cameras [J]. Computer Simulation，2016，33(7)：256-260. (in Chinese)
郑贵林，唐贝贝. 基于多鱼眼摄像头的立体视觉定位系统[J]. 计算机仿真，2016，33(7)：256-260.

[3] KAUR S，ER S K. Analysis of Image Stitching for Noisy Images using SIFT [J]. International Journal of Advanced Research in Computer Science，2017，8(5)：2078-2082.

[4] WANG Y W，YU M，JIANG H，et al. An image stitching algorithm via adaptive quadtree segmentation[J]. Journal of Ningbo University (Natural Science & Engineering)，2018，31(4)：52-58. (in Chinese)
王元炜，郁梅，姜浩，等. 一种自适应四叉树分块的图像拼接算法[J]. 宁波大学学报(理工版)，2018，31(4)：52-58.

[5] HE C，ZHOU J. Mesh-based image stitching algorithm with linear structure protection[J]. Journal of Image and Graphics，2018，23(7)：0973-0983. (in Chinese)
何川，周军. 具有直线结构保护的网格化图像拼接[J]. 中国图象图形学报，2018，23(7)：973-983.

[6] ZHOU X，CAO S，HE X J，et al. Image stitching based on the planar similarity among matching pairs of feature points [J]. Journal of University Electronic Science and Technology of China，2017，46(6)：877-882. (in Chinese)
周雪，曹爽，何香静，等. 基于特征点匹配对平面相似度的图像拼接[J]. 电子科技大学学报，2017，46(6)：877-882.

[7] WANG F B，TU P，WU C，et al. Multi-image mosaic with SIFT and vision measurement for microscale structures processed by femtosecond laser [J]. Optics and Lasers in Engineering，2018，100：124-130.

[8] FATHIMA A A，KARTHIK R，VAIDEHI V. Image stitching with combined moment invariants and sift features [J]. Procedia Computer Science，2013，19：420-427.

[9] LIU W L，WANG Z Y，QING L B，et al. Application of LBP algorithm in rock slice image stitching[J]. Computer & Digital Engineering，2016，44(2)：326-330. (in Chinese)
刘文亮，王正勇，卿粼波，等. LBP 算法在岩石薄片图像拼接中的应用[J]. 计算机与数字工程，2016，44(2)：326-330.

[10] CHEN Y，ZHAO Y，WANG S G. Fast image stitching method based on SIFT feature vector image [J]. Journal of Jilin University (Science Edition)，2017，5(1)：116-122. (in Chinese)
陈月，赵岩，王世刚. 基于 SIFT 特征矢量图的快速图像拼接方法[J]. 吉林大学学报(理学版)，2017，55(1)：116-122.

[11] LU J M，ZHU Z. Real-time 4K panoramic video stitching based on GPU acceleration [J]. Computer Science，2017，44(8)：18-21. (in Chinese)
卢嘉铭，朱哲. 基于 GPU 加速的实时 4K 全景视频拼接[J]. 计算机科学，2017，44(8)：18-21.

[12] LU Y Y，ZHANG M. Improved Algorithm Based on SIFT Infrared Image Stitching Algorithm [J]. Computer Systems & Applications，2015，24(8)：165-170. (in Chinese)
陆园园，张明. 基于 SIFT 算法的红外图像拼接方法改进[J]. 计算机系统应用，2015，24(8)：165-170.

[13] CHEN Y，ZHAO Y，WANG S G. Fast image stitching method based on SIFT with adaptive local image feature[J]. Chinese Optics，2016，9(4)：415-422. (in Chinese)
陈月，赵岩，王世刚. 图像局部特征自适应的快速 SIFT 图像拼接方法[J]. 中国光学，2016，9(4)：415-422.

[14] ZHAO Y，CHEN Y，WANG S G. Corrected fast SIFT image stitching method by combining projection error [J]. Optics and Precision Engineering，2017，25(6)：1645-1651. (in Chinese)
赵岩，陈月，王世刚. 结合投影误差校正的快速 SIFT 图像拼接[J]. 光学精密工程，2017，25(6)：1645-1651.

[15] XU W，MULLIGAN J. Performance evaluation of color correction approaches for automatic multi-view image and video stitching[C]// 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE，2010：263-270.

[16] DU C，YUAN J，DONG J，et al. GPU based Parallel Optimization for Real Time Panoramic Video Stitching [J]. arXiv1810.03988，2018.

[17] JIANG W，GU J. Video stitching with spatial-temporal content-preserving warping[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. IEEE，2015：42-48.

[18] LIU J，LI S Y，LI R F. Multi-view Video Stitching of Outdoor Scenes [J]. Computer Engineering，2016，42(4)：259-265. (in Chinese)
刘娟，李实英，李仁发. 室外场景的多视角视频拼接[J]. 计算机工程，2016，42(4)：259-265.

[19] LI Y，DU B X. Real-time video splicing technology based on small region fusion [J]. Journal of Jilin University (Science Edition)，2016，54(6)：1367-1372. (in Chinese)
李勇，杜丙新. 基于小区域融合的实时视频拼接技术[J]. 吉林大学学报 (理学版)，2016，54(6)：1367-1372.

[20] YANG X P，HU Y，ZHANG K. Research on video mosaicking technology based on FPGA [J]. Journal of Jilin University (Information Science Edition)，2016，34(6)：709-715. (in Chinese)
杨晓萍，胡玉，张凯. 基于 FPGA 的视频拼接技术研究[J]. 吉林大学学报(信息科学版)，2016，34(6)：709-715.

[21] CHANG J Y，QIN R，LI Q，et al. Image quality assessment of panoramic image[J]. Computer Science，2014，41(6)：278-281. (in Chinese)
常嘉义，秦瑞，李庆，等. 全景鸟瞰拼接图像的质量评价方法[J]. 计算机科学，2014，41(6)：278-281.

[22] NOWOZIN S. Autopano-Sift，making panoramas fun [OL]. http://user. cs. tu-berlin. De/nowozin/autopano-sift.

[23] PERBET F，JOHNSON S，PHAM M T，et al. Human Body Shape Estimation Using a Multi-resolution Manifold Forest[C]// 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbus，OH，USA，2014：668-675.

[24] WANG J，WANG J D，ZENG G，et al. Fast Neighborhood Graph Search Using Cartesian Concatenation[C]// 2013 IEEE International Conference on Computer Vision (ICCV). Sydney，NSW，Australia，2013：2128-2135.