快速块匹配运动估计算法的探索与分析

——浙江大学第 13 期 srtp 研究报告

指导教师：　陆系群

组员：　金鑫、徐洋、徐超

快速块匹配运动估计算法的探索与分析

摘　要：本文介绍了块匹配运动估计算法及对现已实现的三种算法（新三步法、四步法、钻石算法）进行分析及对比。

关键词：运动估计算法、块匹配、三步法、四步法、钻石算法

Exploration and Analysis of Fast Block Matching Algorithms for Motion Estimation

Abstract: This article introduces Fast Block Matching Algorithms for Motion Estimation, and analyzes three implemented algorithms (New three-step search algorithm, Four-step search algorithm, diamond search algorithm).

Key Words: Motion Estimation Algorithms, Fast Block Matching, Three-step search algorithm, Four-step search algorithm, diamond search algorithm

第一部分 概述

1．研究背景

视频压缩(Video Compression)技术是计算机处理视频的前提。视频信号数字化后数据带宽很高，通常在 20MB/秒以上，因此计算机很难对之进行保存和处理。采用压缩技术以后通常数据带宽能降到 1-10MB/秒，这样就可以将视频信号保存在计算机中并作相应的处理。

视频图像数据有极强的相关性，也就是说有大量的冗余信息(Redundant information)。冗余信息可分为空域冗余信息和时域冗余信息。压缩技术就是将数据中的冗余信息去掉。它包括帧内图像数据压缩技术(Intra-frame image data compression)、帧间图像数据压缩技术(Interframe image data compression)和熵编码压缩技术(Entropy Encoding compression)。帧间编码可去除时域冗余信息，它包括三部分：运动补偿(Motion estimation)、运动表示(Motion representation)、运动估计(Motion compensation)。

在现有的各类快速运动估计方法中，块匹配运动估计算法(Block matching

algorithms for motion estimation)因具有算法简单、便于实现等优点得到广泛应用，它的基本思想是：将图像序列的每一帧分成许多互不重叠的块(Block)，并认为块内所有象素的位移量都相同。然后对每一当前帧(current frame)的每个块到参考帧(reference frame)某一给定特定搜索范围内，根据一定的匹配准则找出与当前块最相似的块，即匹配块(matching block)。匹配块与当前块的之间的空间位置的相对偏移量即为运动矢量(motion vector)。运动矢量和经过运动匹配后得到的预测误差(difference)共同发送到解码端。在解码端按照运动矢量指明的位置，从已经解码的邻近参考帧中找到相应的块，和预测误差相加后，就得到了块在当前帧中的位置。

运动估计算法是视频压缩领域一个非常重要的研究热点，运动估计所要花费的时间占据了整个编码过程中相当大的比重，各国学者提出了诸多快速算法以减少运动估计的计算复杂度，同时保持原有视频质量。目前，搜索精度最高的块匹配运动估计算法是全搜索法（Full search)，它对搜索范围内的每一个像素点进行匹配运算以得到一个最优的运动矢量。但它的计算复杂度太高，不适合实时应用。为此必须提出快速快匹配运动估计算法(Fast block matching algorithms for motion estimation)。目前已知的一些快速快匹配运动估计算法是三步法(Three-step search algorithm)、四步法(Four-step search algorithm)、钻石算法(diamond search algorithm)等。本项目的主要任务就是用编程实现以上三种算法，并对算法复杂度和结果质量加以分析比较。

2. 研究目的

学习有关视频编码的知识，重点学习运动估计中的快速快匹配运动估计算法，对现有的快速块匹配算法的进行比较分析。

3. 研究方法

1) 图像(视频截图)的输入与输出
2) 编程实现三步搜索块匹配运动估计算法、四步搜索块匹配运动估计算法、钻石搜索块匹配运动估计算法。
3) 简单交互界面的设计、实现，以便于对算法进行测试。
4) 对上述算法的算法效率、结果的精度进行比较。

第二部分 研究结果

1. 研究内容

1） 块匹配运动估计的基本原理：

运动估计的基本思想是将图像序列的每一帧分成许多互不重叠的宏块，并认为宏块内所有像素的位移量都相同，然后对于当前帧中的每一块到前一帧某给定搜索范围内根据一定的匹配准则找出与当前块最相似的块，即匹配块，由匹配块与当前块的相对位置计算出运动位移，所得运动位移即为当前块的运动矢量。示意图如下：

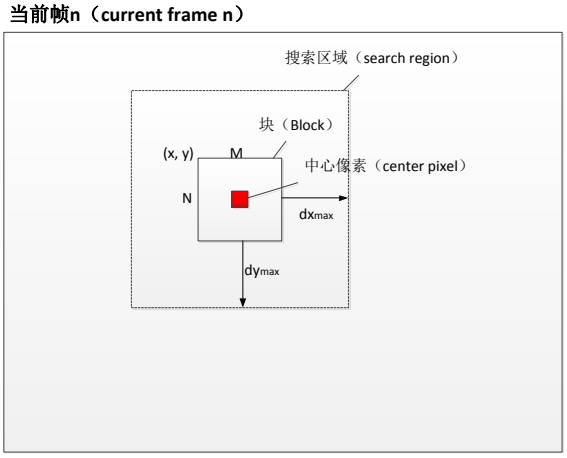


图 1

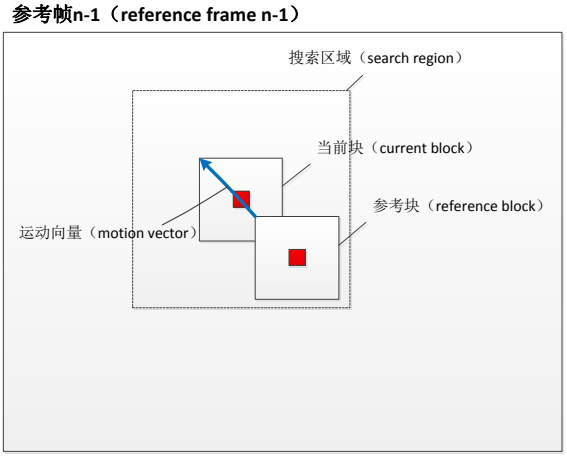


图 2

宏块大小为 M×N，一般取 M=N=16、8 或 4。搜索范围一般由最大偏移矢量来决定，设可能的最大偏移矢量为(dx_{max},dy_{max})则搜索范围为(M+2*dx_{max})*(N+2*dy_{max})。常用搜索最佳匹配准则有绝对误差和(Sum of Absolute Difference，SAD)、平均绝对误差(Mean Absolute Difference, MAD)、方差和(Sum of Sequence Error, SSE)、均方差(Mean Sequence Error, MSE)、绝对变化误差和(Sum of Absolute Transformed Difference , SATD)和归化互相关函数(NCCF)。我们在实现程序中用了绝对误差和(SAD)作为匹配准则，其公式如下：

$$\mathrm{SAD(i,j)} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}|f_n(x,y) - f_{n-1}(x+i, y+j)|$$

其中，(i , j)为位移矢量分别在水平、垂直坐标上的分量，f_n, f_{n-1} 分别为当前帧和参考帧的像素值，M x N 为宏块的大小。

2） 新三步法：

step1：新三步法首先建一个以起始点为中心，包括最大搜索范围的窗口。计算如图 3(a)所示 8 个步长等于或者略大于最大搜索范围的一半的点，8 个相邻点和起始点的匹配误差。

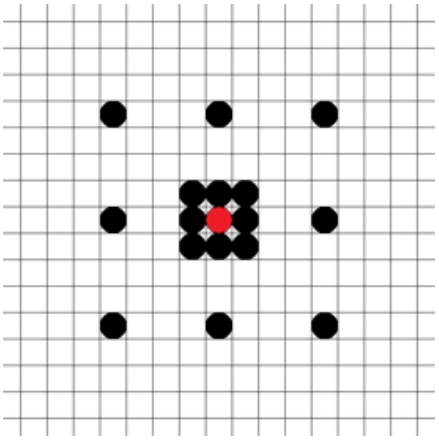
A. 如果匹配误差最小的点是起始点，以起始点作为搜索结果，搜索结束。

B. 如果匹配误差最小的点是起始点周围相邻的点，以这个点为中心，找出如图 3(b)所示的 3 个黑色点或如 3(c)所示的 5 个黑色点的匹配误差。此时的匹配误差最小的点即为最终搜索结果。

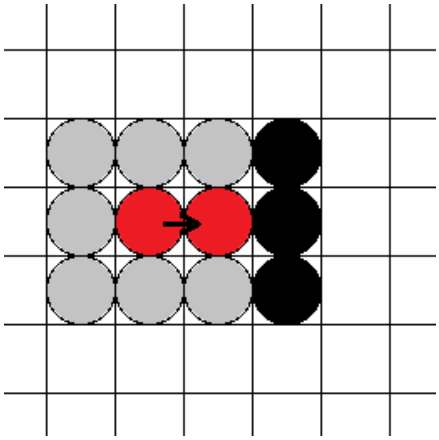
C. 否则，跳到第二步。

step2：以第一步中匹配误差最小的点为中心，步长为原来一半，计算如图 3(d)所示的 9 个点的匹配误差。

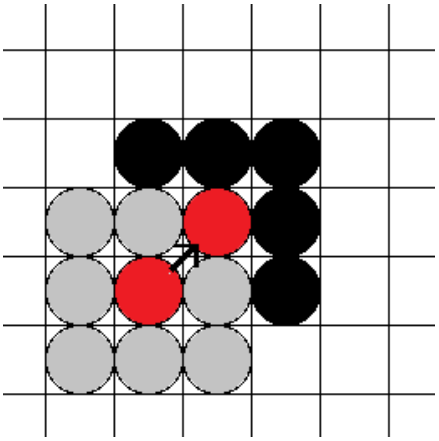
step3：重复第二步直到步长为 1。从如图 3(e)所示的 9 个点中找出最小匹配误差点，即为最终搜索结果。



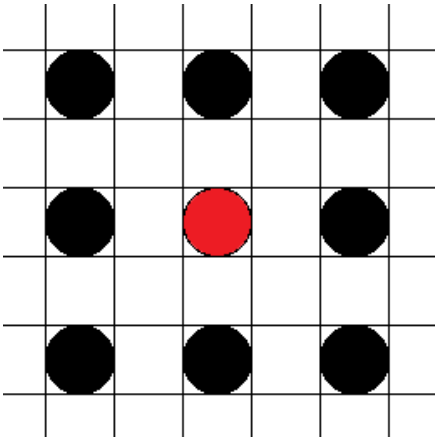
(a)



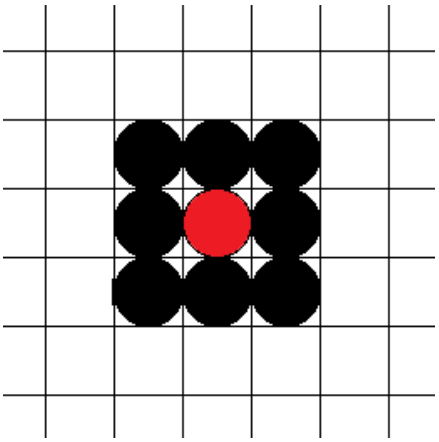
(b)



(c)



(d)



(e)

图 3

图 4 是三个新三步法路径的例子。

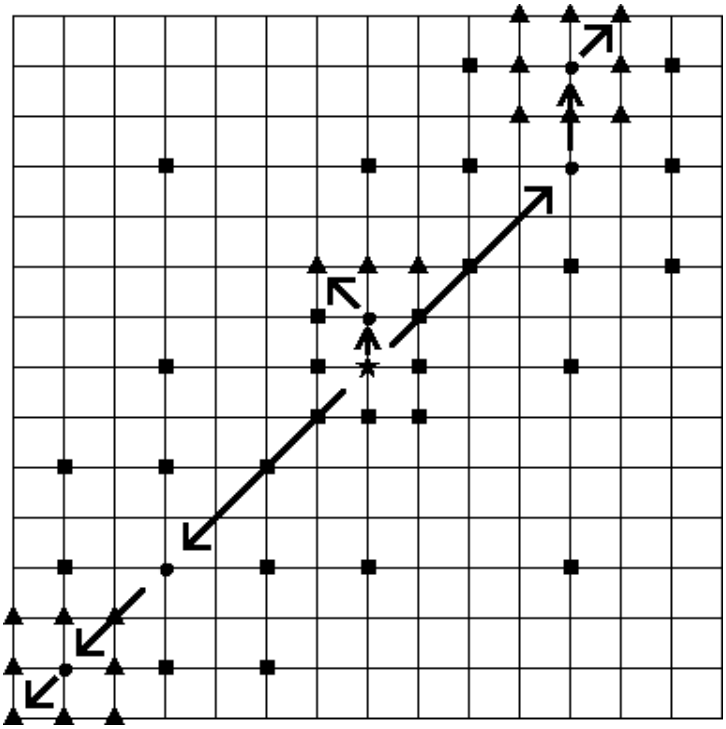


图 4

3） 四步法：

step1：对于最大搜索范围为 7 的搜索，四步法首先建一个以起始点为中心的 5×5 的窗口并计算如图 5(a)所示 9 个点的匹配误差。如果最小匹配误差点为中心，跳到第四步，否则跳到第二步。

step2：以找到最小匹配误差点为新的窗口中心，新建 5×5 的窗口。

A. 如果之前找到的最小匹配误差点为四角，计算新窗口中如图 5(b)所示 5 个黑色点的匹配误差，找到匹配误差最小的点。

B. 如果为四边的中点，计算新窗口中如图 5(c)所示 3 个黑色点的匹配误差。

如果这些点的匹配误差都比中心点大，跳到第四步，否则跳到第三步。

step3：和第二步相同，但终将会跳到第四步。

step4：搜索窗口缩小为 3×3。从如图 5(d)所示的 9 个点中找出最小匹配误差点，即为最终搜索结果。

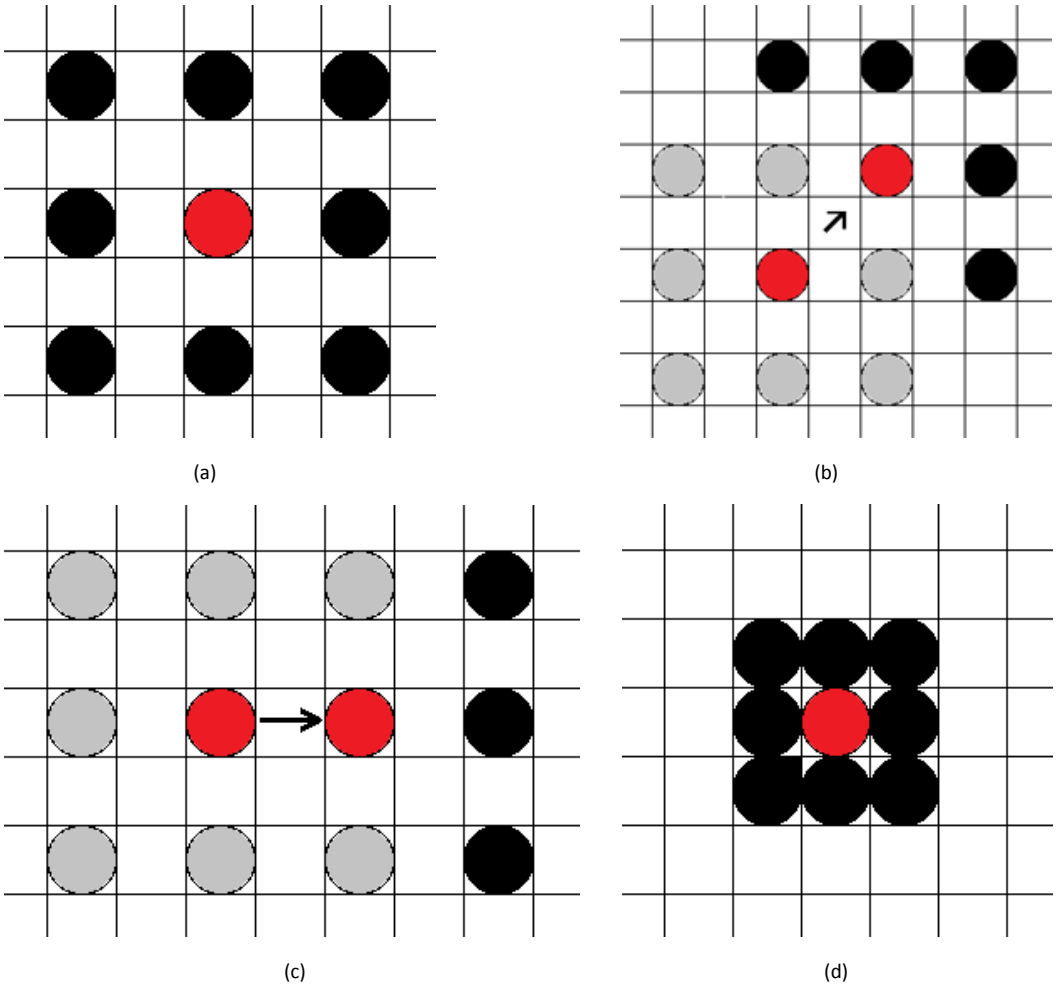


图 5

图 6 是两个四步法路径的例子。

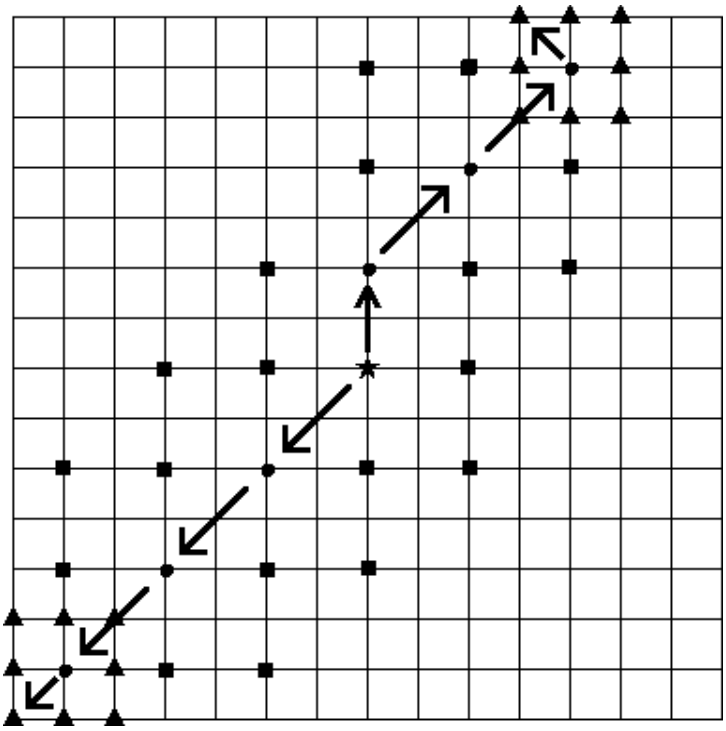


图 6

4） 钻石算法：

step1：对于最大搜索范围为 7 的搜索，钻石算法首先建一个以起始点为中心的 5×5 的窗口并计算如图 7(a)所示 9 个点的匹配误差。如果最小匹配误差点为中心，跳到第三步，否则跳到第二步。

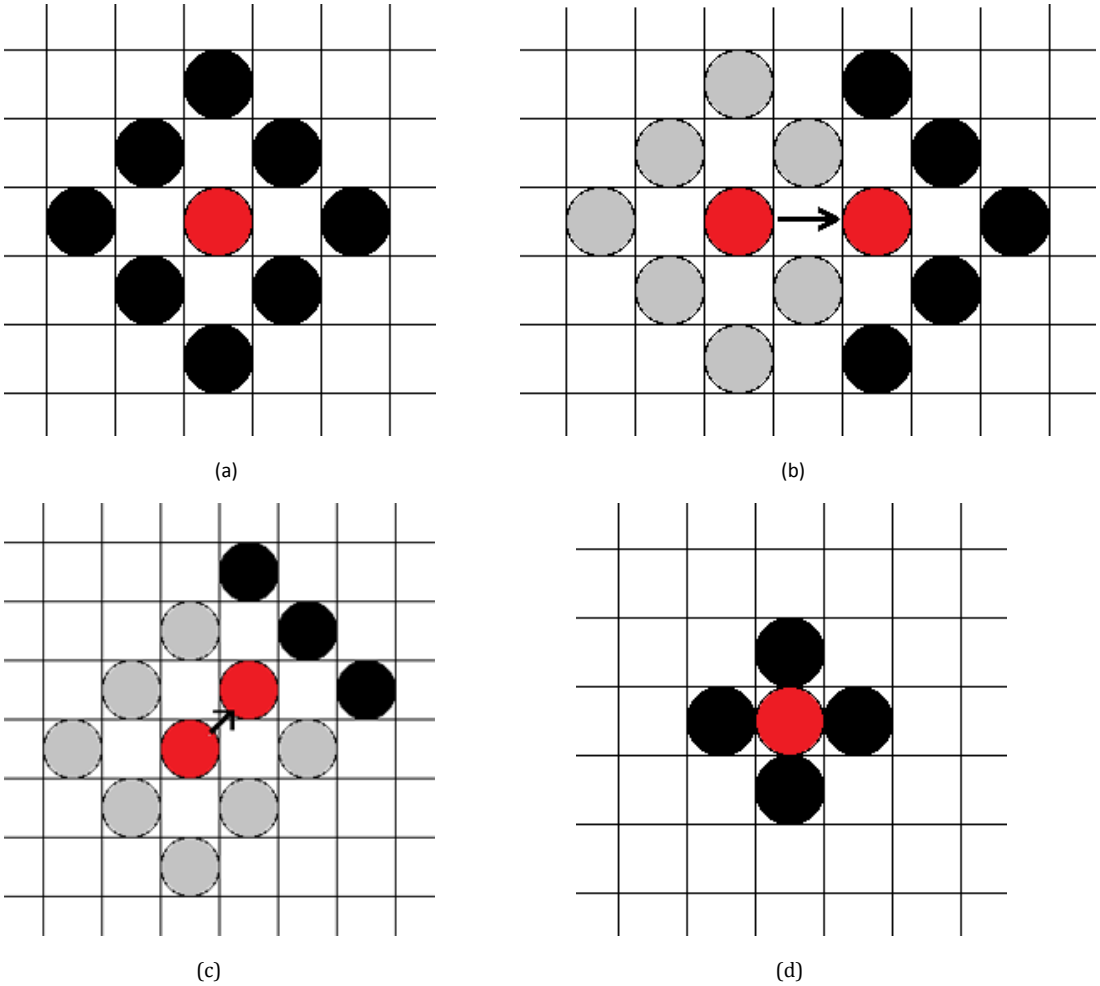
step2：以找到最小匹配误差点为新的窗口中心，新建 5×5 的窗口。

A. 如果之前找到的最小匹配误差点为四角，计算新窗口中如图 7(b)所示 5 个黑色点的匹配误差，找到匹配误差最小的点。

B. 如果为四边的中点，计算新窗口中如图 7(c)所示 3 个黑色点的匹配误差。

如果这些点的匹配误差都比中心点大，跳到第三步，否则跳到第二步。

step3：搜索窗口缩小为 3×3。从如图 7(d)所示的 5 个点中找出最小匹配误差点，即为最终搜索结果。



(a) (b) (c) (d)

图 7

图 8 是两个钻石算法路径的例子。

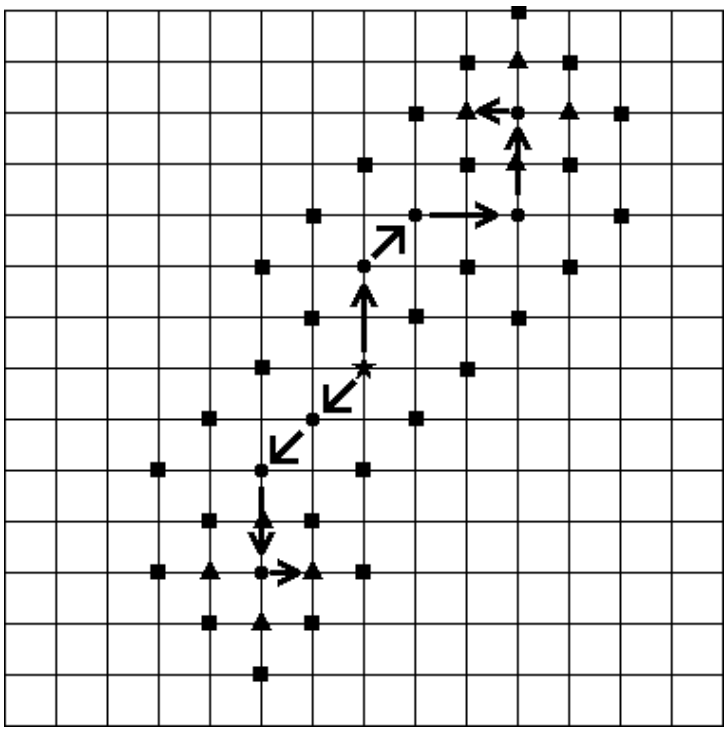


图 8

2. 主要成果

为了方便测试，写了一个基于.Net Framework 4.0 的演示程序。程序读入用户选择的参考帧和当前帧，根据用户选择的块匹配算法和块大小，进行运动估计，并根据运动估计的结果生成预测帧。弹出窗口显示参考帧、当前帧、预测帧和测试信息。

下图为演示程序主界面。

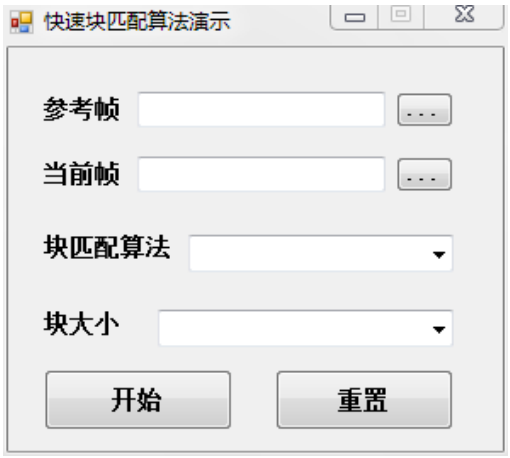


图 9

点击参考帧之后的...按钮可弹出选择参考帧对话框用于选择一张 bmp 图片作为参考帧。

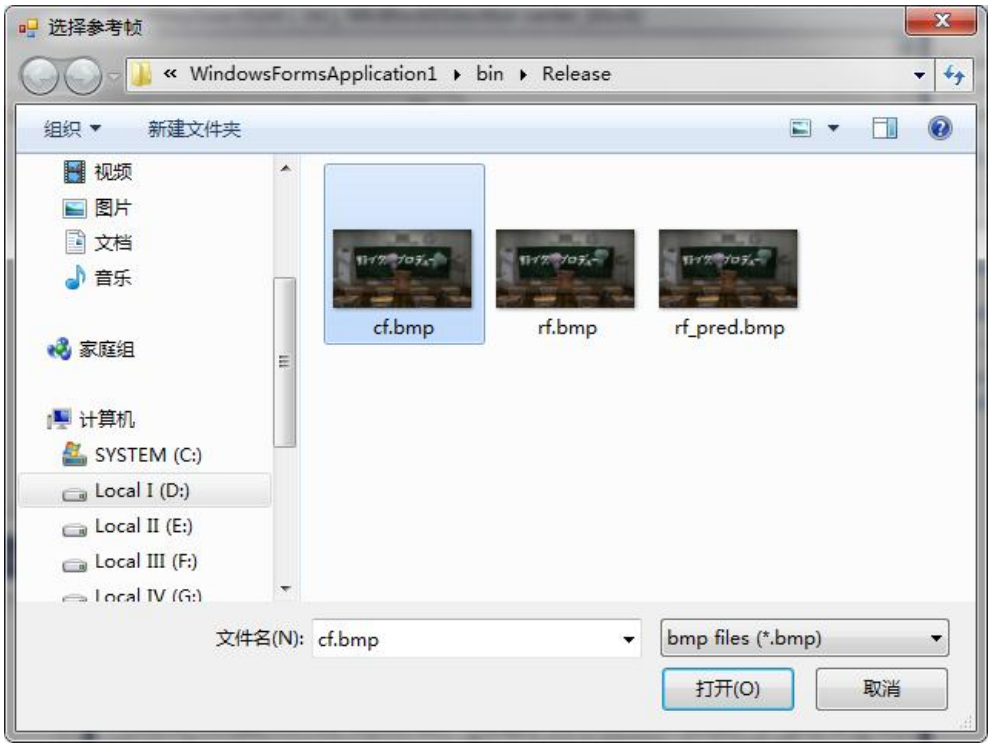


图 10

点击当前帧之后的...按钮可弹出选择当前帧对话框用于选择一张 bmp 图片作为当前帧。

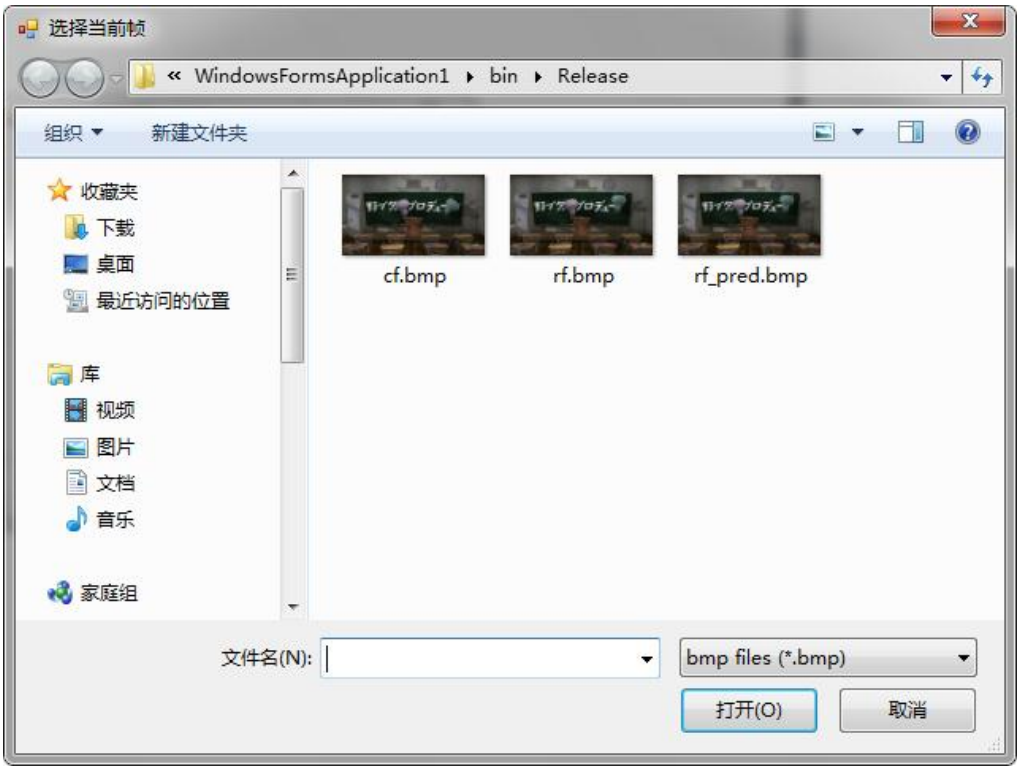


图 11

块匹配算法之后的下拉框用于选择块匹配算法类型。

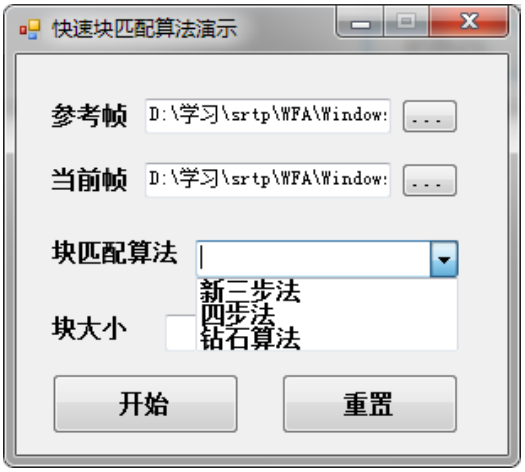


图 12

块大小之后的下拉框用于选择块的大小（4x4, 8x8, 16x16）。

图 13

点重置按钮，清空已选参考帧、当前帧、块匹配算法和块大小，重新进行选择。

图 14

任何一个选项为空，点开始，都会弹出提示信息。

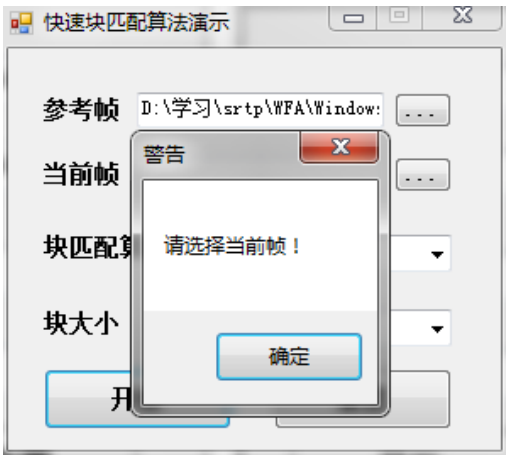


图 15

选择好参考帧、当前帧等信息后，点开始，开始进行快速块匹配，结果存在参考帧所在目录下的以 参考帧名_record.txt 命名的文件中，预测帧为参考帧所在目录下以 参考帧名_pred.bmp 命名的文件。同时弹出如下窗口显示参考帧、当前帧、预测帧和算法信息。

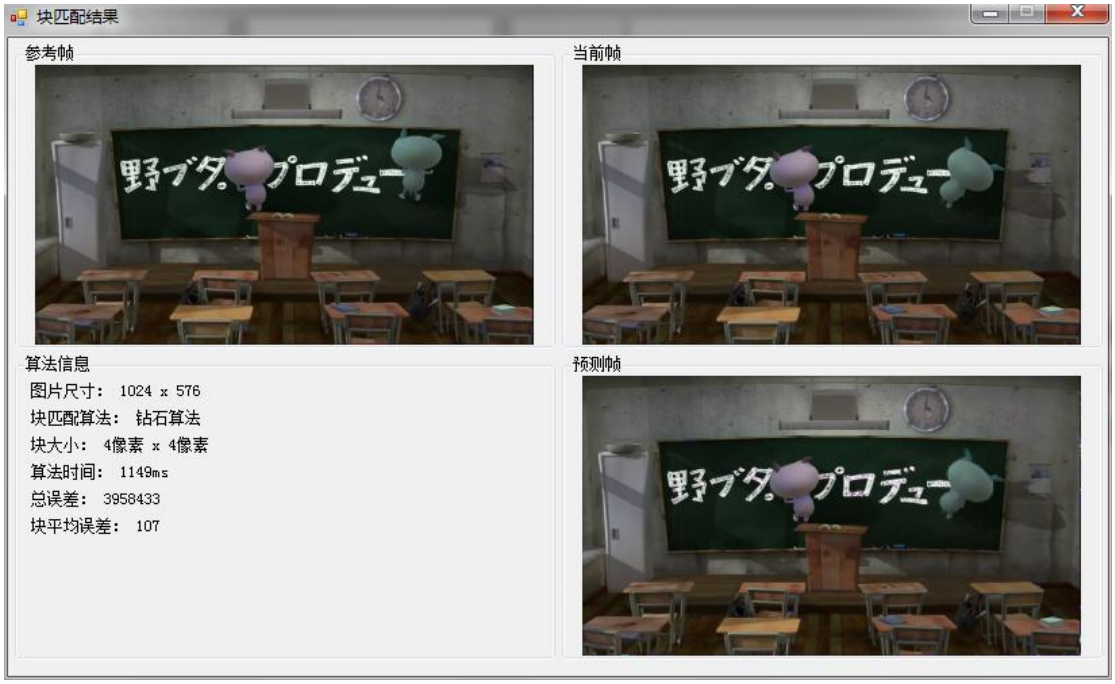


图 16

点击图片可以在弹出窗口查看原图。

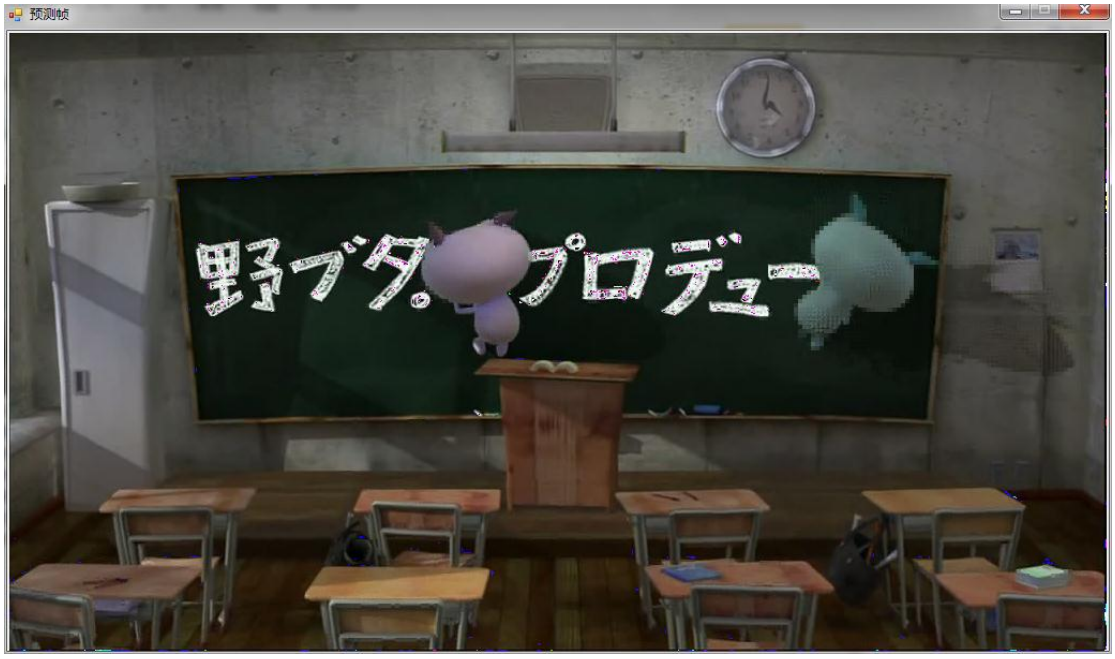


图 17

3. 分析与讨论

为了比较这 3 种算法，我们在程序实现算法后用 2 组图片来测试了算法。2 组图片每组 9 张，均来自于电影，图片分辨率 1920*1080.第一组是动态的连续帧，第二组是静态的连续帧，以此来衡量 3 种算法在 2 种情况下的优劣。

新三步法：

新三步法									
	4*4			8*8			16*16		
	总误差	块平均误差	算法时间	总误差	块平均误差	算法时间	总误差	块平均误差	算法时间
高动态图片									
1-2	55502680	428	3143	64469096	1989	3146	76774108	9549	3118
2-3	103432821	789	3240	112212598	3463	3206	125491897	15608	3212
3-4	76305937	588	3389	84219484	2599	3298	96479438	11199	3258
4-5	143140388	1104	3633	151609395	4679	3077	164651509	20479	3238
5-6	78354822	604	3471	84693193	2613	3156	94087474	11702	3152
6-7	110655008	853	3492	117183295	3616	3217	126982290	15793	3174
7-8	124588895	961	3474	131395119	4055	3209	142014965	17663	3224
8-9	103861222	801	3616	111182104	3431	3348	122159100	15193	3118
AVG	99480222	766	3432	107120536	3306	3207	118580098	14648	3187
高静态图片									
1-2	19787523	152	3231	22395489	691	2988	25272885	3143	3033
2-3	32148216	248	3329	35447304	1094	3281	39501883	4913	3266
3-4	18787396	144	2875	21340499	658	2735	24135831	3001	2732
4-5	18812941	145	2957	21391638	660	2870	24393541	3034	2905
5-6	19794391	152	2843	22448640	692	2661	25360409	3154	2753
6-7	20632645	159	2913	23417787	722	2699	26658934	3315	2685
7-8	36576740	282	2939	40182110	1240	2971	44319601	5512	2909
8-9	21603472	166	2938	24411764	753	2838	27510384	3421	2772
AVG	23517916	181	3003	26379404	814	2880	29644184	3687	2882

图 18

四步法：

	四步法								
	4*4			8*8			16*16		
	总误差	块平均误差	算法时间	总误差	块平均误差	算法时间	总误差	块平均误差	算法时间
高动态图片									
1-2	43911735	338	3373	44317615	1376	3892	45862546	5704	4417
2-3	87436806	674	3382	83658611	2582	4197	84905427	10560	5232
3-4	55485563	428	3469	53923918	1664	4006	56443186	7020	4877
4-5	118941359	917	3544	113248909	3495	4230	113922469	14169	5111
5-6	56730908	437	3613	54926639	1695	4170	55700315	6927	5204
6-7	87010457	671	3908	84025193	2593	4323	84428443	10501	5141
7-8	101372786	782	3707	96818681	2988	4542	97380819	12112	5187
8-9	85680827	661	3684	82309834	2540	4289	83346681	10366	5202
AVG	79571305	614	3585	76653675	2367	4206	77748736	9670	5046
高静态图片									
1-2	17391892	134	3038	18787932	579	3075	21279576	2646	3092
2-3	29245420	225	3097	29438364	908	3453	29662537	3689	3651
3-4	17006578	131	2763	18451989	569	2805	20729210	2578	2845
4-5	16815262	129	2846	18306266	565	2869	20230483	2516	2938
5-6	17958189	138	2818	19123110	590	2931	21022867	2614	2960
6-7	19320066	149	2826	20207647	623	2966	22052571	2742	3075
7-8	33886069	261	3099	33051736	1020	3423	33300572	4141	3680
8-9	19264395	148	2795	19860073	612	2942	20903659	2599	3056
AVG	21360984	164	2910	22153390	683	3058	23647684	2941	3162

图 19

钻石算法：

	钻石算法								
	4*4			8*8			16*16		
	总误差	块平均误差	算法时间	总误差	块平均误差	算法时间	总误差	块平均误差	算法时间
高动态图片									
1-2	46024181	355	3742	44761376	1381	4630	46003342	5721	5664
2-3	90286018	696	3711	84342773	2603	5237	84708676	10535	7251
3-4	57741470	445	3721	54433893	1680	4871	56573169	7036	6315
4-5	121291497	935	4025	113516931	3503	5256	114117209	14193	7109
5-6	58355264	450	3966	55432683	1710	5194	56155860	6984	6485
6-7	89254809	688	4020	84689383	2613	5091	84924267	10562	6717
7-8	104146836	803	3983	97488214	3008	5323	97400885	12114	6955
8-9	88021357	679	4011	83512314	2577	5286	83467355	10381	6774
AVG	81890179	631	3897	77272196	2384	5111	77918845	9691	6659
高静态图片									
1-2	17643054	136	2797	18901506	583	3052	21008694	2613	3105
2-3	30555744	235	3048	30124410	929	3492	29838921	3698	4506
3-4	16920374	130	2587	18120221	559	2691	20315271	2526	2990
4-5	17145781	132	2669	18089682	558	2766	20154193	2506	2869
5-6	17984986	138	2596	19002450	586	2798	20947489	2605	2821
6-7	19148044	147	2672	19909177	614	2875	21686519	2697	2948
7-8	34447690	265	3089	33048147	1020	3453	32645934	4060	3964
8-9	19691397	151	2749	19782375	610	2887	20550347	2556	3045
AVG	21692134	167	2776	22122246	682	3002	23393421	2908	3281

图 20

从上面三张图可观察出以下几点现象：

首先，不同的测试图片对测试结果的影响。

对于处理高动态图片，新三步法在效率上相比四步法和钻石算法有很大优势，但是误差也比较大。四步法和钻石算法效果相当，但是四步法花费较少的时间。

对于处理高静态图片，新三步法在效率上的优势不大。新三步法误差较大。

其次，不同块大小对测试结果的影响。

新三步法：块大小的增大，块数量减少，算法时间减少。误差随块大小增大而增大，误差增加幅度最大。

四步法：块大小增大，算法时间增加，并且动态图片比静态图片增加幅度大。静态图片的误差随块大小增大而增大，动态图片的误差不显著随块大小变化。

钻石算法：块大小增大，算法时间增加，并且增加幅度比四步法大，动态图片比静态图片增加幅度大。静态图片的误差随块大小增大而增大，动态图片的误差不显著随块大小变化。

第三部分 结论

1. 研究结论

三种块匹配算法中新三步法对于存在大量运动的连续帧处理速度比较理想，但是也是三种算法中误差最大的。四步法对于大量运动的连续帧处理速度可以接受，误差也比较小。钻石算法处理大量运动的连续帧时则应该使用较小的块大小，以提高算法速度。而对于基本保持静止的连续帧的处理，新三步法由于误差太大并且速度上也无绝对的优势。其他两种算法在误差上相差无几，只是钻石法在匹配块比较小时速度上有优势，而四步法在匹配块比较大时速度上有优势。

根据以上结果，如果应用中不是十分在乎图像质量，但是实时性要求比较高的系统可以采用新三步法，并选择较大的匹配块以加快速度。如果系统中存在大量运动的连续帧，且对图片质量的要求比较高，可用四步法。如果系统中存在大量基本不变的连续帧，且对图片质量要求比较高，可用钻石算法，并选择较小的匹配块以减小误差。

2. 建议和展望

我们只实现了最为基础的运动估计算法，在此基础上可以有多种方法对算法进行优化。运动估计算法的效率主要体现在图像质量和搜索速度(复杂度)两方面。运动估计越准确，预测补偿的图像质量越高，补偿的残差就越小，补偿编码所需位数也就越少，且比特率也就越小；运动估计速度越快，越有利于实时应用。提高图像质量，加快估计速度，减小比特率是运动估计算法研究的目标。通过研究初始搜索点的选择、匹配准则、运动搜索策略可以有效提高算法效率。例如：利用相邻块的运动矢量来预测当前块的搜索起点、采用中止判别准则来减少搜索复杂度、采用子采样方法计算块匹配失真来降低计算复杂度等等。这些可以留作后续研究的内容。

3. 问题与对策

由于希望对 bmp 格式有些了解并且基于对效率的考虑，并没有用.NET 中图形处理的类，而是自己写了一个 bmp 输入输出类以获取一部分需要用到的 bmp

图像信息，该类的 C++代码由陆系群老师提供。我们将其移植到了.NET 平台下。

新三步法的算法描述不是很清楚，除了加入一步中止和两部中止，其他部分和新步法一致，但没有给出三步法的描述。所以一开始写的新三步法中与三步法相同的部分不正确，在上网查阅三步法详细描述之后才将之改正。

开始时将参考帧也分块，采取参考帧中块与当前帧中块的匹配，得到预测帧效果相当不好。后来反复阅读论文即其他参考资料后，改为参考帧中不分块，从参考帧中寻找当前帧的匹配块。这种方法使得预测帧效果大大提高。

开始算法都是 C++程序实现，但是由于最后需要写一个演示程序，需要一些简单的界面，组内成员对于 MFC、Qt 等不熟悉，故改用.NET 平台编写界面。后来发现 C++代码与.NET 平台难以兼容，故将所有算法重新用 C#语言重写。

参考文献：

[1] G. K. Wallace, “The JPEG still picture compression standard,” IEEE Trans. On Consumer Electronics, Dec. 1991

[2] R. Li, B. Zeng, and M. L. Liou, “A new three-step algorithm for block motion estimation,” IEEE Trans. On Circuits and Systems for Video Technology, 4(4): 438-442, 1994

[3] L.-M. Po and W.-C. Ma, “A novel four-step search algorithm for fast block motion estimation,” IEEE Trans. On Circuits and Systems for Video Technology, 6(3): 313-317, 1996

[4] S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast block-matching motion estimation,” IEEE Trans. On Image Processing, 9(2): 287-290, 2000