测试人：刘栩宏

测试时间2023-07-10

测试代码

struct timespec start\_time, end\_time; struct timespec start\_time, end\_time; long long elapsed\_time;  
// 获取开始时间 clock\_gettime(CLOCK\_MONOTONIC, &start\_time);  
// 执行需要计时的代码段 // ... DataPathCB( &(psCmd->sParam.sData) ); runDetectAlgBlocking(); runTrackingAlg();  
// 获取结束时间 clock\_gettime(CLOCK\_MONOTONIC, &end\_time);  
// 计算经过的时间 elapsed\_time = ((end\_time.tv\_sec - start\_time.tv\_sec) \* 1000000000LL +    (end\_time.tv\_nsec - start\_time.tv\_nsec))/1000;  
// 输出经过的时间（以微秒为单位） printf("Elapsed time: %lld us\n", elapsed\_time);

未绑定核心时的算法时间为  
Elapsed time: 2396 us  
Elapsed time: 2350 us  
Elapsed time: 2301 us  
Elapsed time: 2338 us  
Elapsed time: 2338 us  
Elapsed time: 2523 us  
Elapsed time: 2399 us  
Elapsed time: 2343 us  
Elapsed time: 2315 us  
Elapsed time: 2328 us  
绑定到核心2后的算法时间为

Elapsed time: 2037 us  
Elapsed time: 2084 us  
Elapsed time: 2086 us  
Elapsed time: 2114 us  
Elapsed time: 2054 us  
Elapsed time: 2068 us  
Elapsed time: 2039 us  
Elapsed time: 2036 us  
Elapsed time: 2091 us  
Elapsed time: 2105 us

绑定核心代码

cpu\_set\_t cpuSet; cpu\_set\_t cpuSet;    \_\_CPU\_ZERO\_S(sizeof(cpu\_set\_t), &cpuSet);  // 清空CPU集合  
    int cpuCore = 2;  // 要绑定的CPU核心    \_\_CPU\_SET\_S(cpuCore, sizeof(cpu\_set\_t), &cpuSet);  // 将指定的CPU核心添加到CPU集合中  
    // 获取当前线程的ID    pid\_t pid = getpid();  
    // 设置线程的CPU亲和性，将其绑定到指定的CPU核心    if (sched\_setaffinity(pid, sizeof(cpuSet), &cpuSet) == -1) {        printf("Failed to set CPU affinity.\n");        return 1;    }  
    printf("Thread bound to CPU core %d.\n", cpuCore);

测试结论：

绑定CPU后运行效率提升在100多us，提升效果并不非常明显，且不绑定核心效果已远远超出8ms运算的需求，认为暂时不需要绑定核心运行。