

Accurate, Dense, and Robust Multi-View Stereopsis论文分析与代码实现（四）



Accurate, Dense, and Robust Multi-View Stereopsis论文分析与代码实现（四）

本文版权属于重庆大学计算机学院刘骥，禁止转载

Accurate, Dense, and Robust Multi-View Stereopsis论文分析与代码实现（四）

稀疏重建代码优化

- 1.程序的整体结构
- 2.数据格式
 - 2.1特征点文件
 - 2.2 匹配文件
 - 2.3 轨迹文件
- 3.类结构
- 4.程序说明

稀疏重建代码优化

本文对上一个文档编写的程序进行若干优化：

1. 优化类结构，以符合面向对象的思想。
2. 优化程序结构，将计算特征点、计算匹配和计算轨迹拆分为3个独立的过程。

第1项优化的目的纯粹是本人爱好，看着结构不好的代码，整个人都不好啦。第2项优化是一个常规操作。因为计算特征点、计算匹配和计算轨迹每一项所耗费的时间都很长，模块独立之后，可以针对某一项的算法进行优化，而不会影响到其他部分。采用这种方法，计算特征点和计算匹配的结果可以保存为文件，计算轨迹时就直接读取这些文件，这样在优化轨迹计算算法时就不用每次都去计算特征点和匹配啦。

1.程序的整体结构

优化后的程序用命令行参数将计算特征点、计算匹配和计算轨迹拆分为3个独立的过程（也可以采用独立的程序，但鄙人认为过于复杂）。`sparse.cpp` 是程序执行的入口代码如下：

```
int main(int argc, char *argv[])
{
    if(argc<2)
    {
        cout<<"使用说明:"<<endl;
        cout<<"计算特征点: sparse sift [图像目录] [参数文件] [特征点文件]"<<endl;
        cout<<"计算匹配: sparse match [特征点文件] [匹配文件]"<<endl;
        cout<<"计算轨迹: sparse track [图像目录] [参数文件] [特征点文件]"<<endl;
        cout<<"计算全部: sparse all [图像目录] [参数文件] [特征点文件]"<<endl;
        exit(0);
    }
    string command=argv[1];

    if(command=="sift")
    {
        //特征点计算
        string imageDir=argv[2];
        string parFileName=argv[3];
        string keypointsFileName=argv[4];
        ImageSet imageset(imageDir, parFileName);
        KeyPoints keypoints(imageset.images);
        keypoints.saveTo(keypointsFileName);
        cout<<"保存特征点到文件"<<keypointsFileName<<endl;

    }else if(command=="match")
    {
        //特征点匹配
        string keypointsFileName=argv[2];
        string matchesFileName=argv[3];
        KeyPoints keypoints(keypointsFileName);
        Matches matches(keypoints);
        matches.saveTo(matchesFileName);
        cout<<"保存匹配到文件"<<matchesFileName<<endl;

    }else if(command=="track")
    {
        //轨迹计算
        string imageDir=argv[2];
        string parFileName=argv[3];
        string keypointsFileName=argv[4];
```

```

        string matchesFileName=argv[5];
        string tracksFileName=argv[6];
        ImageSet imageset(imageDir,parFileName);
        KeyPoints keypoints(keypointsFileName);
        Matches matches(matchesFileName);
        TrackList trackList(keypoints, matches);
        trackList.triangulate(imageset.kpmats);
        trackList.saveTo(tracksFileName);
        cout<<trackList<<endl;
        cout<<"保存轨迹到文件"<<tracksFileName<<endl;
    }else if(command=="all")
    {
        //特征点计算、特征点匹配、轨迹计算
        string imageDir=argv[2];
        string parFileName=argv[3];
        string keypointsFileName=argv[4];
        string matchesFileName=argv[5];
        string tracksFileName=argv[6];
        ImageSet imageset(imageDir,parFileName);
        KeyPoints keypoints(imageset.images);
        keypoints.saveTo(keypointsFileName);
        cout<<"保存特征点到文件"<<keypointsFileName<<endl;
        Matches matches(keypoints);
        matches.saveTo(matchesFileName);
        cout<<"保存匹配到文件"<<matchesFileName<<endl;
        TrackList trackList(keypoints, matches);
        trackList.triangulate(imageset.kpmats);
        trackList.saveTo(tracksFileName);
        cout<<"保存轨迹到文件"<<tracksFileName<<endl;
    }
    return 0;
}

```

代码不用注释也应该能够看懂吧。程序执行时如果不输入任何命令行参数，则打印使用说明：

```

LiuJi-MacBook-Pro:code liuji$ ./build/sparse
使用说明：
计算特征点：sparse sift [图像目录] [参数文件] [特征点文件]
计算匹配：sparse match [特征点文件] [匹配文件]
计算轨迹：sparse track [图像目录] [参数文件] [特征点文件] [匹配文件] [轨迹文件]
计算全部：sparse all [图像目录] [参数文件] [特征点文件] [匹配文件] [轨迹文件]

```

以技术特征点为例，输入 `sparse sift ../images/templeSparseRing/templeSR_par.txt keypoints.txt`（注意 / 是必须的）意味着对 `../images/templeSparseRing` 目录下的 `templeSR_par.txt` 文件进行解析，并读取该目录下的图像文件，提取sift特征，并将特征点保存在 `keypoints.txt` 文件中。

接下来输入 `sparse match keypoints.txt matches.txt` 就执行特征匹配过程，并将结果保存在 `matches.txt` 文件中。

最后输入 `sparse track ../images/templeSparseRing/ templeSR_par.txt keypoints.txt matches.txt tracks.txt` 执行计算轨迹的过程，结果保存在 `tracks.txt`。

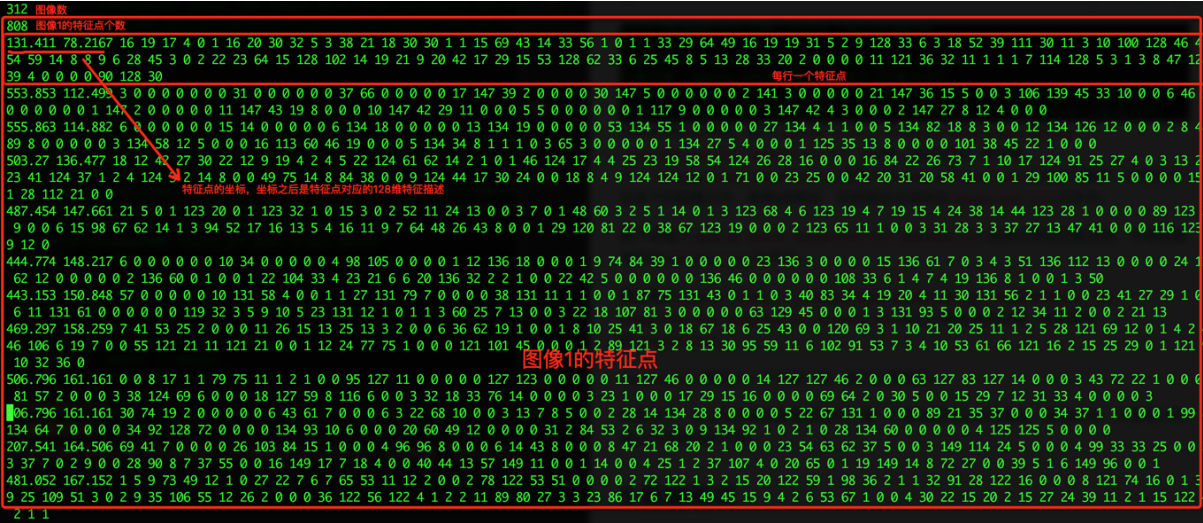
`sparse all` 就和上一个文档一样，执行全部操作。

前面说了，这么划分模块是有好处的，每一个算法都可以单独调试，单独优化，只要输入和输出的结构保持不变。

2.数据格式

不同计算过程之间通过文件交换数据，文件格式是怎么样的呢？

2.1特征点文件



[特征点的x坐标][特征点的y坐标][特征描述128维向量]

2.2 匹配文件

312	图像数量	
456	1 1 2 2 5 92 7 102 8 14 9 15 11 16 12 17 14 21 15 22 16 24 18 26 19 27 20 30 21 31 22 32 23 33 24 34 25 36 26 37 31 40 42 163 46 50 48 51 49 192 50 193 51 54 52 56 53 5	
7 55 58 56 59 57 60 59 61 60 65 62 67 63 68 64 69 65 72 70 259 71 260 72 266 73 267 75 75 77 283 78 79 79 80 81 81 90 98 91 99 94 320 100 117 101 116 104 112 106 120 107 33		
0 108 334 109 123 111 125 114 129 115 345 116 126 117 127 118 134 121 131 122 135 123 137 124 132 125 133 127 136 129 138 132 140 133 141 139 145 141 39 146 150 148 41 159		
166 162 365 164 170 163 168 166 174 170 171 171 175 172 172 173 173 174 380 175 49 176 386 178 381 179 180 180 181 182 184 186 186 187 187 188 188 189 190 190 191 191 194 1		
92 202 193 195 194 200 195 201 197 208 198 199 199 196 201 197 202 205 203 206 204 207 205 210 206 212 207 213 208 216 209 211 210 215 211 217 212 218 213 220 214 224 215 2		
26 216 227 218 216 218 222 221 237 224 235 225 236 226 63 227 238 228 239 230 242 232 245 233 246 234 247 235 248 236 71 242 253 244 449 245 450 247 256 248 257 249 258 250		
262 251 263 255 270 256 271 257 272 258 274 259 273 262 275 263 276 264 278 265 279 267 280 268 281 269 282 271 287 272 288 273 290 274 291 275 293 277 295 278 294		
283 300 284 296 286 303 288 301 291 315 296 509 297 508 298 512 300 323 301 326 303 327 305 114 306 517 307 121 308 329 309 333 310 122 312 335 313 336 314 343 315 344 316		
342 317 337 318 338 319 340 320 346 321 348 322 332 323 349 324 139 325 350 326 351 327 353 328 149 331 357 335 358 336 361 337 366 341 367 347 169 348 371 349 370 350 372		
351 375 353 378 354 178 358 382 359 383 360 384 361 385 387 418 388 419 390 421 391 422 392 229 393 423 394 424 395 425 396 427 397 231 398 232 399 428 400 429 401 431 403 430		
407 381 408 382 409 383 411 384 410 385 413 386 415 387 418 388 419 390 421 391 422 392 229 393 423 394 424 395 425 396 427 397 231 398 232 399 428 400 429 401 431 403 430		
407 434 409 438 410 439 412 440 413 441 414 442 415 443 416 249 417 444 418 250 419 445 425 451 426 452 429 453 430 456 431 457 433 458 435 459 436 460 437 462 438 463 439		
464 440 277 441 465 442 468 443 469 444 470 445 466 448 472 449 473 450 476 451 477 452 478 453 481 454 479 455 480 456 482 457 485 458 292 474 499 476 504 477 622 479 511		
480 514 482 518 483 519 484 520 485 521 486 523 487 524 488 525 491 528 492 635 498 533 501 535 503 529 504 637 508 640 511 537 513 538 514 377 515 385 517 543 518 650 520		
550 522 557 524 555 525 556 526 558 527 562 528 563 529 560 531 561 532 564 533 565 534 420 536 570 537 566 538 567 539 571 540 572 545 574 546 433 547 575 550 437 551 579		
558 671 560 581 561 583 564 588 565 589 566 591 568 592 569 593 570 596 571 597 572 598 573 599 574 600 576 603 579 680 585 617 586 618 591 620 593 625 595 627 597 630 598		
631 599 632 600 633 608 643 609 644 611 646 612 647 613 693 614 648 615 649 618 695 619 547 620 652 621 656 622 655 623 659 624 573 627 432 628 660 629 577 631 665 632 666		
633 663 634 664 636 670 639 673 640 674 644 601 645 678 646 679 647 602 658 683 662 629 663 686 664 687 666 688 667 634 671 639 673 641 674 691 675 692 679 715 680 717 681		
696 683 661 684 698 685 701 686 667 688 704 689 677 690 705 701 735 706 710 707 689 711 712 712 713 714 716 715 718 719 720 721 700 722 723 724 724 727 746 734 759 736 731		
740 739 741 740 745 743 748 767 749 749 750 750 751 751 752 753 753 754 755 728 761 737 766 763 767 764 769 765 770 766 771 768 772 769 774 773 775 774 777 775 780 776 781		
777 782 778 783 779 786 783 788 786 791 787 792 788 793 790 794 791 797 792 798 793 799 794 805 798 806 799		图像1和图像2的特征点匹配
323 7 111 8 14 9 15 13 20 15 120 16 22 17 126 19 28 20 134 21 29 24 31 25 33 33 37 46 43 47 380 49 184 50 185 51 48 53 49 54 50 55 51 56 52 57 53 58 56 60 68 62 67 65 73 68		
72 72 237 73 238 74 247 86 285 93 114 95 110 100 328 102 25 104 117 105 329 106 330 108 132 109 129 111 337 112 124 114 136 115 340 116 133 117 131 118 143 119 127 121 137		
123 146 124 140 125 141 127 138 128 142 130 145 139 151 146 36 164 365 166 168 167 167 168 159 170 164 171 170 172 165 179 174 180 375 181 537 182 176 184 172 185 236 186		
177 187 178 188 180 190 183 191 188 192 206 194 193 197 208 198 391 199 186 201 187 202 189 203 190 204 192 205 55 206 195 207 196 208 197 209 194 210 199 211 408 212 200 2		
15 204 216 205 218 57 225 64 230 221 231 437 233 224 235 226 244 233 247 443 248 445 252 450 255 244 256 246 257 251 258 254 259 250 263 252 264 253 265 255 269 79 271 261		
274 262 277 264 283 279 284 489 285 276 286 286 291 310 295 312 297 315 298 513 300 318 303 325 305 327 307 130 308 520 310 128 312 331 313 332 314 342 315 343 317 333 318		
334 319 336 322 324 323 347 324 345 325 527 326 348 328 154 335 355 341 359 348 371 350 166 351 366 353 382 354 368 356 360 358 379 359 377 360 378 362 538 364 383 365 179		
367 386 368 542 369 388 370 389 371 550 372 398 374 393 375 396 376 395 377 191 378 397 379 400 380 402 381 403 382 404 383 406 385 410 386 412 387 415 388 416 389 418 390		
419 392 209 393 210 394 420 395 421 397 214 399 425 403 424 413 434 414 436 418 230 430 580 431 452 432 453 433 454 435 457 436 458 437 459 438 460 439 461 440 456 441 463		
443 467 444 468 445 469 446 472 447 473 449 471 450 476 453 592 454 258 455 259 456 477 474 514 479 512 480 326 482 521 484 523 485 524 486 528 491 532 492 530 498 641 500		
661 501 536 512 643 513 535 514 381 518 540 520 548 525 554 526 555 528 409 531 556 533 557 534 417 537 558 538 559 540 564 541 563 544 567 545 426 546 429 547 754 549 571		
550 433 553 570 554 440 558 572 564 449 566 583 568 464 569 465 570 591 571 589 574 593 580 737 591 507 592 374 593 630 595 632 597 633 598 708 599 706 600 707 608 653 611		
649 612 650 613 720 615 651 619 654 620 658 622 561 623 562 624 666 627 427 628 566 632 671 633 668 634 669 636 681 639 579 640 682 642 686 644 596 646 687 655 696 661 705		
662 634 664 710 666 635 667 711 675 718 676 717 681 723 684 727 685 729 686 672 689 685 690 733 702 744 705 640 708 745 711 716 715 753 719 726 722 758 735 768 739 805 748		
804 749 780 751 784 752 785 758 786 770 783 774 818 779 807 781 816 782 828 783 829 793 837 797 838 798 839 799 842		

文件格式是：

[图像数量]

[图像1和图像2的匹配]

[图像1和图像3的匹配]

[图像1和图像4的匹配]

.....

[图像2和图像3的匹配]

[图像2和图像4的匹配]

[图像2和图像5的匹配]

.....

其中 图像i和图像j的匹配 格式如下：

[匹配点数][图像i的特征点序号][图像j的特征点序号].....[图像i的特征点序号][图像j的特征点序号].....

2.3 轨迹文件

5434 图像数量

```

-0.0514439 0.154766 -0.00178954 5 0 553.853 112.495 1 555.537 111.895 57 556.857 109.984 26 555.384 111.936 59 553.596 105.882
-0.0290771 0.0847568 0.0255063 3 0 366.938 169.815 1 367.118 175.944 3 368.054 197.673
-0.0260711 0.134002 0.0232333 10 0 505.143 175.167 1 505.784 182.131 2 506.556 192.426 23 502.661 158.347 24 503.832 165.057 25 504.854 173.054 26 505.813 182.322 29 497.316
163.615 30 497.457 156.959 28 497.145 171.896
-0.000147368 0.0444462 -0.0087567 5 6 0 367.733 254.118 1 367.633 252.124 25 367.769 255.24 56 370.555 252.175 57 370.385 254.052 58 371.944 254.139
0.00254213 0.032923 -0.00442744 25 0 229.513 263.273 1 229.68 261.838 2 229.9 259.265 3 230.067 256.083 24 229.122 265.672 25 229.403 264.1 26 229.683 262.055 28 232.516 26
3.759 29 231.6 265.053 55 234.163 258.946 56 233.376 261.506 60 240.057 258.032 103 247.049 257.485 5 230.401 245.614 61 241.199 254.851 62 242.198 251.295 250 373.96 205.49
3 52 235.862 248.433 53 235.385 252.337 63 243.092 247.357 101 250.038 250.499 102 248.662 254.268 100 251.173 246.348 248 362.1 210.595 247 364.466 206.222
0.00480865 0.0470901 -0.00483269 26 0 268.719 268.617 1 268.821 267.054 2 269.005 264.477 3 269.101 261.239 27 268.301 271.493 24 268.456 270.702 25 268.688 269.431 26 268.8
05 267.325 27 71.875 267.074 28 270.941 269.015 29 270.128 270.556 54 273.298 261.046 55 272.591 264.308 56 271.754 266.961 57 271.285 268.414 82 275.926 266.145 4 269.294
257.575 5 269.244 253.424 53 273.086 257.318 59 276.037 266.217 60 277.385 263.468 61 278.697 260.102 52 276.558 253.129 62 279.915 256.294 63 280.908 252.039 100 287.697 25
1.045 轨迹对应的三维坐标 5个视图可见该点 视图0上的轨迹坐标 视图59上的轨迹坐标
0.00654517 0.0470945 -0.00502296 6 0 268.916 273.058 1 268.946 271.708 23 268.699 275.838 24 268.775 275.311 25 268.857 273.973 26 268.942 272.002
0.0115748 0.0223175 0.0190286 17 0 196.215 287.032 1 196.639 290.982 2 197.424 296.046 26 196.687 291.19 57 192.147 286.405 3 197.974 299.78 27 193.14 290.425 53 198.838 301
.535 82 192.239 289.063 5 199.569 303.622 54 196.977 299.052 56 193.245 290.267 62 201.731 300.361 58 190.599 285.709 63 205.037 301.511 51 203.06 302.733 49 206.651 298.826
0.0325129 0.0889011 -0.0103173 10 0 380.811 340.475 1 380.647 337.549 3 379.203 324.068 28 382.948 342.575 56 384.052 338.335 4 378.768 315.006 52 388.803 305.016 63 398.557
303.936 51 389.737 293.469 50 389.928 281.015
0.0331784 0.0659644 -0.00837062 19 0 320.38 343.198 1 320.056 340.529 2 319.897 334.924 24 320.59 346.097 25 320.4 344.453 26 320.113 340.718 27 323.619 340.739 28 322.056 3
44.507 54 326.766 327.776 55 325.288 335.01 56 323.764 340.6 57 322.667 343.309 58 324.967 343.308 81 323.592 343.948 82 327.505 340.139 59 327.522 340.188 80 319.491 345.54
9 4 319.296 318.934 29 320.196 346.165
0.0464387 0.0959126 -0.00698567 8 0 400.423 377.666 1 399.627 374.853 25 400.802 378.777 26 399.632 375.08 28 401.037 379.872 57 401.537 378.823 58 401.176 379.827 81 399.49
9 380.414
-0.0338888 0.117225 0.0231755 9 0 457.588 155.139 1 458.182 160.881 25 457.308 152.503 26 458.221 161.025 27 448.588 160.033 28 449.022 151.344 56 448.673 159.926 57 448.963
154.142 82 436.161 158.55
-0.0331921 0.0893558 0.0211879 14 0 379.682 159.017 1 380.113 163.54 25 379.622 156.918 26 379.982 163.785 27 371.002 163.563 28 371.965 156.616 56 371.169 163.406 57 371.56
3 158.804 58 362.336 158.678 59 361.036 162.867 82 360.976 162.874 29 373.541 150.618 104 349.124 162.829 83 348.945 162.918
0.0290457 0.0297197 0.0246086 4 0 213.564 173.088 1 213.289 178.591 56 206.291 179.05 3 212.441 198.017
-0.0283219 0.0446451 0.0253534 5 0 254.934 174.163 1 254.712 179.843 2 253.929 189.198 25 255.136 171.547 26 254.637 179.93
-0.0282974 0.0465145 0.024639 8 0 260.16 174.211 1 260.325 179.753 25 260.144 171.699 26 260.206 179.9 28 252.662 171.85 29 253.612 164.895 56 251.884 179.935 57 252.378 174

```

文件格式是：

[图像数量]

[轨迹1]

[轨迹2]

[轨迹3]

.....

其中 轨迹*i* 的格式如下：

[x坐标][y坐标][z坐标][可见轨迹的视图数量][视图序号][视图上的x坐标][视图上的y坐标].....

[视图序号][视图上的x坐标][视图上的y坐标].....

3.类结构

程序在上一个文档的基础上增加了 `ImageSet`、`KeyPoints` 和 `Matches` 三个类。其定义分别为：

```

class ImageSet{
public:
    ImageSet(const string&imageDir,const string&parFileName);
    vector<Mat> images;
    vector<string> imageNames;
    vector<Mat> kmats;
    vector<Mat> pmats;
    vector<Mat> kpmats;

};

class KeyPoints{
private:
    vector<vector<KeyPoint> > keypointsVec;
    vector<Mat> descriptorsVec;
public:

```

```

    KeyPoints(const vector<Mat>&images);
    KeyPoints(const string&fileName);
    int getFrameNum() const;
    int getKeyPointNum(int i) const;
    KeyPoint getKeyPoint(int i,int j) const;
    const vector<KeyPoint>& getKeyPoints(int i) const;
    const Mat& getDescriptors(int i) const;
    void saveTo(string fileName);
};

class Matches{
private:
    void pairwiseMatch(const vector<KeyPoint>&keypoints1,const vector<KeyPoint>&keypoints2,const Mat&descriptors1,const Mat&descriptors2,vector<DMatch>& matches);
    vector<vector<vector<DMatch> > > matchesTable;
public:
    Matches(const KeyPoints&keyPoints);
    Matches(const string&fileName);
    const vector<DMatch>& getMatches(int i,int j) const;
    void saveTo(const string&fileName);
};

```

`KeyPoint` 和 `Matches` 都有两个构造函数，其中一个用数据构造，另一个则直接读取文件。两者都有一个 `saveTo` 方法用于保存结果。 `TrackList` 也增加了对应的方法：

```

class TrackList{
private:
    vector<Track> tracks;
public:
    TrackList(const KeyPoints&keypoints,const Matches&matches);
    TrackList(const string&fileName);
    //三角化
    void triangulate(const vector<Mat>&pmats);
    void getColor(const vector<Mat>&images);
    //保存到ply文件
    void save2ply(const string&fileName);
    void saveTo(const string&fileName);
    friend ostream&operator<<(ostream&os,const TrackList&trackList);
};

```

下面贴一下 `KeyPoint` 的构造函数以及 `saveTo` 方法，其他类类似，各位可以直接阅读代码。

```

KeyPoints::KeyPoints(const vector<Mat>&images):keypointsVec(image
s.size(),vector<KeyPoint>()),descriptorsVec(images.size(),Mat())
{
    SIFT sift;
    for (int i=0; i<images.size(); i++) {
        cout<<"计算视图"<<i<<"的特征点, ";
        sift(images[i],Mat(),keypointsVec[i],descriptorsVec[i]);
        cout<<"特征点共有"<<keypointsVec[i].size()<<"个"<<endl;
    }
}
KeyPoints::KeyPoints(const string&fileName)
{
    ifstream ifs(fileName.c_str());
    int nImages;
    ifs>>nImages;
    keypointsVec.resize(nImages,vector<KeyPoint>());
    descriptorsVec.resize(nImages);
    for(int i=0;i<nImages;i++)
    {
        int nPoints;
        ifs>>nPoints;
        descriptorsVec[i]=Mat(nPoints, 128, CV_32F);
        for (int j=0; j<nPoints; j++) {
            KeyPoint point;
            Mat descriptor=descriptorsVec[i].row(j);
            ifs>>point.pt.x>>point.pt.y;
            for (int k=0; k<descriptor.cols; k++) {
                ifs>>descriptor.at<float>(0,k);
            }
            keypointsVec[i].push_back(point);
        }
    }
}
void KeyPoints::saveTo(string fileName)
{
    ofstream ofs(fileName.c_str());
    ofs<<keypointsVec.size()<<endl;
    for (int i=0; i<keypointsVec.size(); i++) {
        ofs<<keypointsVec[i].size()<<endl;
        for (int j=0; j<keypointsVec[i].size(); j++) {
            ofs<<keypointsVec[i][j].pt.x<<" "<<keypointsVec[i][j]
.pt.y<<" ";
            Mat descriptor=descriptorsVec[i].row(j);
            for (int k=0; k<descriptor.cols; k++) {
                if(k==descriptor.cols-1)
                    ofs<<descriptor.at<float>(0,k)<<endl;
                else
                    ofs<<descriptor.at<float>(0,k)<<" ";
            }
        }
    }
    ofs.close();
}

```


4.程序说明

在程序目录下执行 `make` 即可完成编译，编译后程序在 `build` 目录下，之后在目录下执行 `build/sparse` 运行程序。