

# HAAG Weekly Report (Simplified) – Nikita Angarski – 3D Modeling

Week 3

## Time-Log

- What did you do this week?
  - Had our first meeting with Dr. Porto on 1/21, which really got us more centered on what our work is to be for the semester. As it is the first semester of this project, this was crucial to get the key details into place for the rest of the project.
  - Read through some research articles prescribed by Dr. Porto on Point set registration, this involved re-learning some linear algebra from undergrad to really get a grasp on how it operates.
  - Set up a GitHub fork for the algorithm we will be directly working on, which is pycpd, forked and linked below.
  - Set up my environment for programming 3D vision, this involves getting anaconda up, and accessing GaTech PyCharm (JetBrain) license, and downloading 3D visualization software (Slicer)
  - For the Programs Seminar Role, I wrote up a procedural document draft, linked here:
- What are you going to do next week
  - Get Pycpd code to work, and explore ways of accomplishing the task: editing the Gaussian MM code to accept a SSM representation (avg) of a point set.
  - Perform PCA process on the point set data.
- Blockers, things you want to flag, problems, etc.
  - Not yet a block, just need some more time to go over the research papers and code.

Abstracts:

## Point-Set Registration: Coherent Point Drift

<https://arxiv.org/abs/0905.2635>

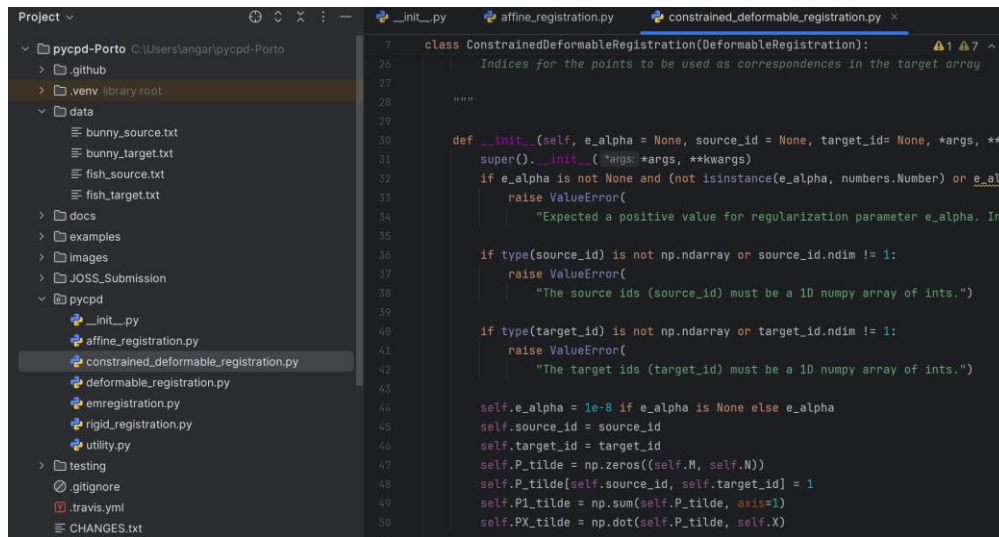
Abstract:

Point set registration is a key component in many computer vision tasks. The goal of point set registration is to assign correspondences between two sets of points and to recover the transformation that maps one point set to the other. Multiple factors, including an unknown non-rigid spatial transformation, large dimensionality of point set, noise and outliers, make the point set registration a challenging problem. We introduce a probabilistic method, called the Coherent Point Drift (CPD) algorithm, for both rigid and non-rigid point set registration. We consider the alignment of two point sets as a probability density estimation problem. We fit the GMM centroids (representing the first point set) to the data (the second point set) by maximizing the likelihood. We force the GMM centroids to move coherently as a group to preserve the topological structure of the point sets. In the rigid case, we impose the coherence constraint by re-parametrization of GMM centroid locations with rigid parameters and derive a closed form solution of the maximization step of the EM algorithm in arbitrary dimensions. In the non-rigid case, we impose the coherence constraint by regularizing the displacement field and using the variational calculus to derive the optimal transformation. We also introduce a fast algorithm that reduces the method computation complexity to linear. We test the CPD algorithm for both rigid and non-rigid transformations in the presence of noise, outliers and missing points, where CPD shows accurate results and outperforms current state-of-the-art methods.

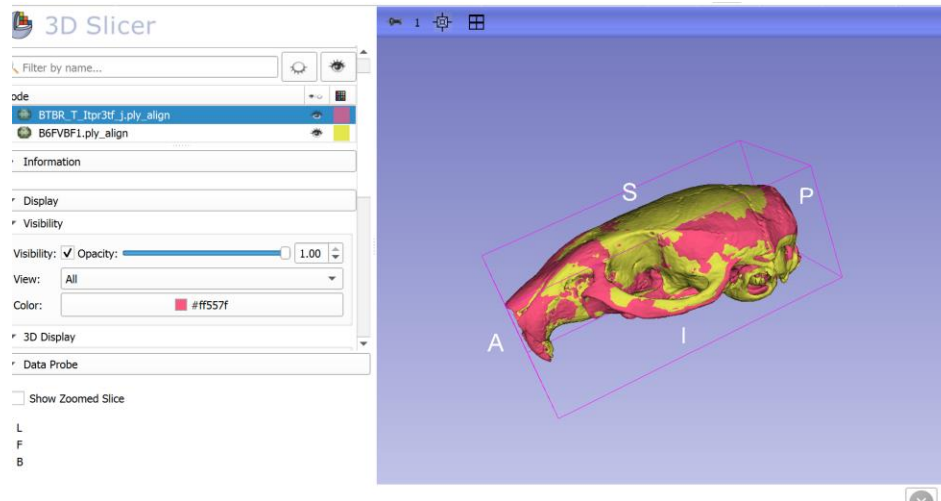
Summary: This research paper introduces the Coherent Point Drift (CPD) algorithm, a probabilistic method for point set registration—the process of aligning two sets of points. CPD addresses both rigid and non-rigid transformations, handling noise, outliers, and missing data effectively. The algorithm uses a Gaussian Mixture Model (GMM) framework and an Expectation-Maximization (EM) approach for optimization. A key innovation is enforcing "coherence" in the movement of points to preserve topological structure. The authors also present a fast implementation of the CPD algorithm to handle large datasets.

## What did you do and prove it

Much of this week was setup and reading more background information on point set registration and the main algorithm, which is coherent point drift. I'm still getting my head around it, but already have done some testing and looking over code provided by Dr. Porto. Have not been able to get the main algorithm working yet, but I hope to have at least an idea of how to edit the algorithm to take a statistical shape model and use it as a basis for GMM. The paper I looked at this week helped a lot, since it details a lot of the math behind the eventual algorithm we will be working on.



```
Project v
├── pycpd-Porto C:\Users\angar\pycpd-Porto
│   ├── .github
│   ├── .venv library root
│   └── data
│       ├── bunny_source.txt
│       ├── bunny_target.txt
│       ├── fish_source.txt
│       └── fish_target.txt
│   ├── docs
│   ├── examples
│   ├── images
│   ├── JOSS_Submission
│   └── pycpd
│       ├── __init__.py
│       ├── affine_registration.py
│       ├── constrained_deformable_registration.py
│       ├── deformable_registration.py
│       ├── emregistration.py
│       ├── rigid_registration.py
│       └── utility.py
│   ├── testing
│   ├── .gitignore
│   ├── .travis.yml
│   └── CHANGES.txt
└── 26
    27
    28
    29
    30
    31
    32
    33
    34
    35
    36
    37
    38
    39
    40
    41
    42
    43
    44
    45
    46
    47
    48
    49
    50
    51
    52
    53
    54
    55
    56
    57
    58
    59
    60
    61
    62
    63
    64
    65
    66
    67
    68
    69
    70
    71
    72
    73
    74
    75
    76
    77
    78
    79
    80
    81
    82
    83
    84
    85
    86
    87
    88
    89
    90
    91
    92
    93
    94
    95
    96
    97
    98
    99
    100
    101
    102
    103
    104
    105
    106
    107
    108
    109
    110
    111
    112
    113
    114
    115
    116
    117
    118
    119
    120
    121
    122
    123
    124
    125
    126
    127
    128
    129
    130
    131
    132
    133
    134
    135
    136
    137
    138
    139
    140
    141
    142
    143
    144
    145
    146
    147
    148
    149
    150
    151
    152
    153
    154
    155
    156
    157
    158
    159
    160
    161
    162
    163
    164
    165
    166
    167
    168
    169
    170
    171
    172
    173
    174
    175
    176
    177
    178
    179
    180
    181
    182
    183
    184
    185
    186
    187
    188
    189
    190
    191
    192
    193
    194
    195
    196
    197
    198
    199
    200
    201
    202
    203
    204
    205
    206
    207
    208
    209
    210
    211
    212
    213
    214
    215
    216
    217
    218
    219
    220
    221
    222
    223
    224
    225
    226
    227
    228
    229
    230
    231
    232
    233
    234
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    250
    251
    252
    253
    254
    255
    256
    257
    258
    259
    260
    261
    262
    263
    264
    265
    266
    267
    268
    269
    270
    271
    272
    273
    274
    275
    276
    277
    278
    279
    280
    281
    282
    283
    284
    285
    286
    287
    288
    289
    290
    291
    292
    293
    294
    295
    296
    297
    298
    299
    300
    301
    302
    303
    304
    305
    306
    307
    308
    309
    310
    311
    312
    313
    314
    315
    316
    317
    318
    319
    320
    321
    322
    323
    324
    325
    326
    327
    328
    329
    330
    331
    332
    333
    334
    335
    336
    337
    338
    339
    340
    341
    342
    343
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368
    369
    370
    371
    372
    373
    374
    375
    376
    377
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
class ConstrainedDeformableRegistration(DeformableRegistration):
    """
    Indices for the points to be used as correspondences in the target array
    """
    def __init__(self, e_alpha = None, source_id = None, target_id = None, *args, **kwargs):
        super().__init__(*args, **kwargs)
        if e_alpha is not None and (not isinstance(e_alpha, numbers.Number) or e_alpha <= 0):
            raise ValueError(
                "Expected a positive value for regularization parameter e_alpha. Ins"
            )
        if type(source_id) is not np.ndarray or source_id.ndim != 1:
            raise ValueError(
                "The source ids (source_id) must be a 1D numpy array of ints."
            )
        if type(target_id) is not np.ndarray or target_id.ndim != 1:
            raise ValueError(
                "The target ids (target_id) must be a 1D numpy array of ints."
            )
        self.e_alpha = 1e-8 if e_alpha is None else e_alpha
        self.source_id = source_id
        self.target_id = target_id
        self.P_tilde = np.zeros((self.M, self.N))
        self.P_tilde[self.source_id, self.target_id] = 1
        self.P1_tilde = np.sum(self.P_tilde, axis=1)
        self.PX_tilde = np.dot(self.P_tilde, self.X)
```



Link to Repo: <https://github.com/Nikitos1865/pycpd-Porto>

Link to Seminar Procedure Doc:

[https://gtvault.sharepoint.com/:w/s/HAAG/ER3TQ5a7m4VlrxEAwTbH4U4Ba\\_HO0qqJSGRwLSVB6a9pdg?e=cpuWuY](https://gtvault.sharepoint.com/:w/s/HAAG/ER3TQ5a7m4VlrxEAwTbH4U4Ba_HO0qqJSGRwLSVB6a9pdg?e=cpuWuY)

