



Automatic Reconstruction of X-ray Scenes with Python and DataLab

<https://datalab-platform.com/>

July 21 2026



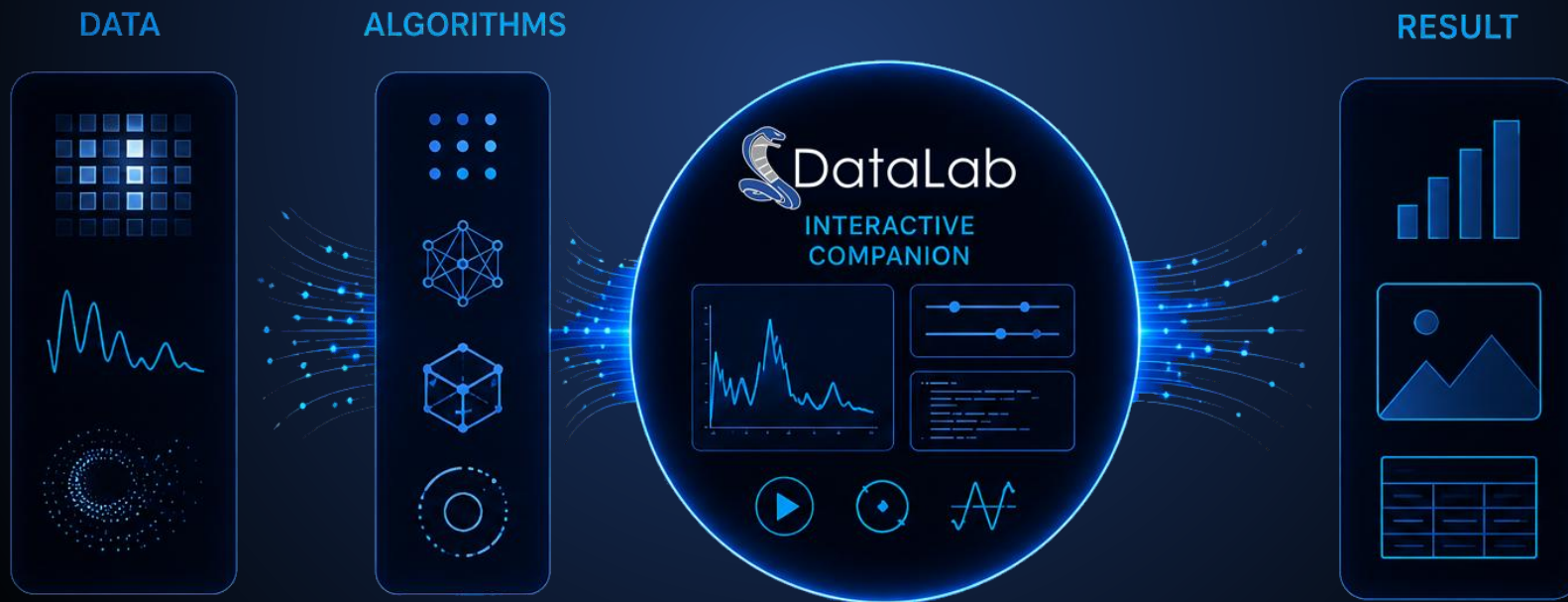
EuroSciPy 2026

CODRA

Pierre RAYBAUT

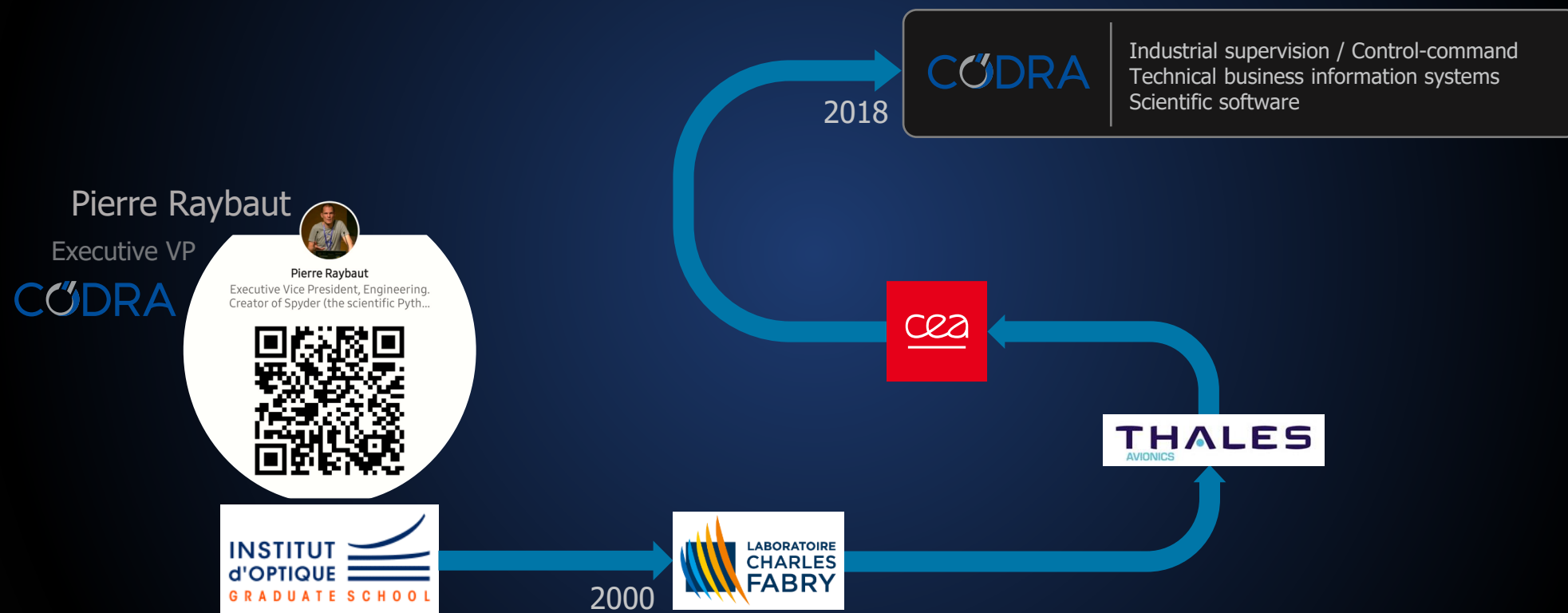
Why this talk?

A real-world case study:
Accelerating scientific algorithm development with DataLab



Introduction

Who am I?



Who am I?



Pierre Raybaut

Executive VP

CODRA



Pierre Raybaut
Executive Vice President, Engineering.
Creator of Spyder (the scientific Pyth...

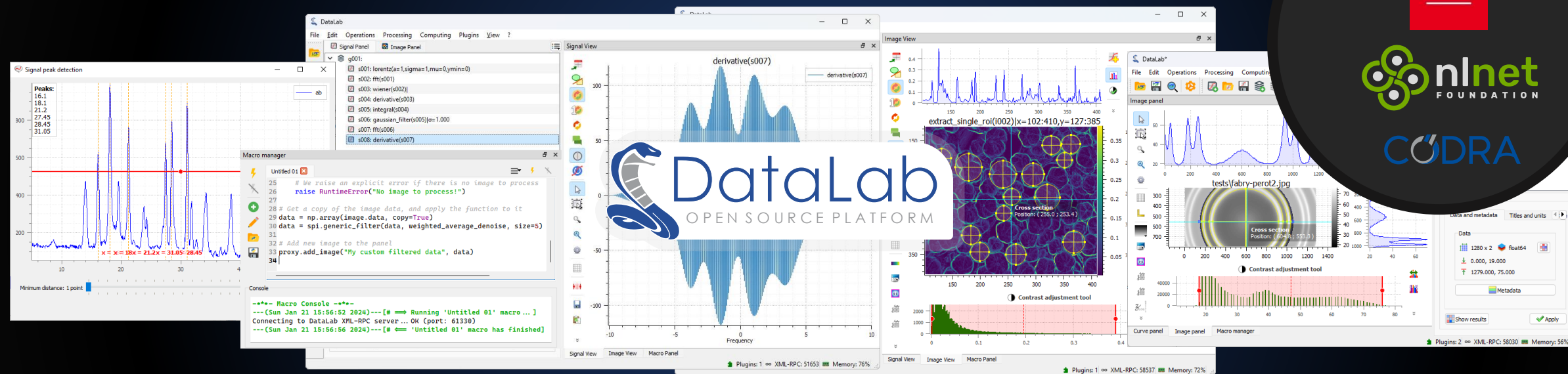


Open-source background



What is DataLab?

<https://datalab-platform.com>



- Open-source platform for processing and visualizing 1D signals or 2D images
- Built on Python and scientific libraries

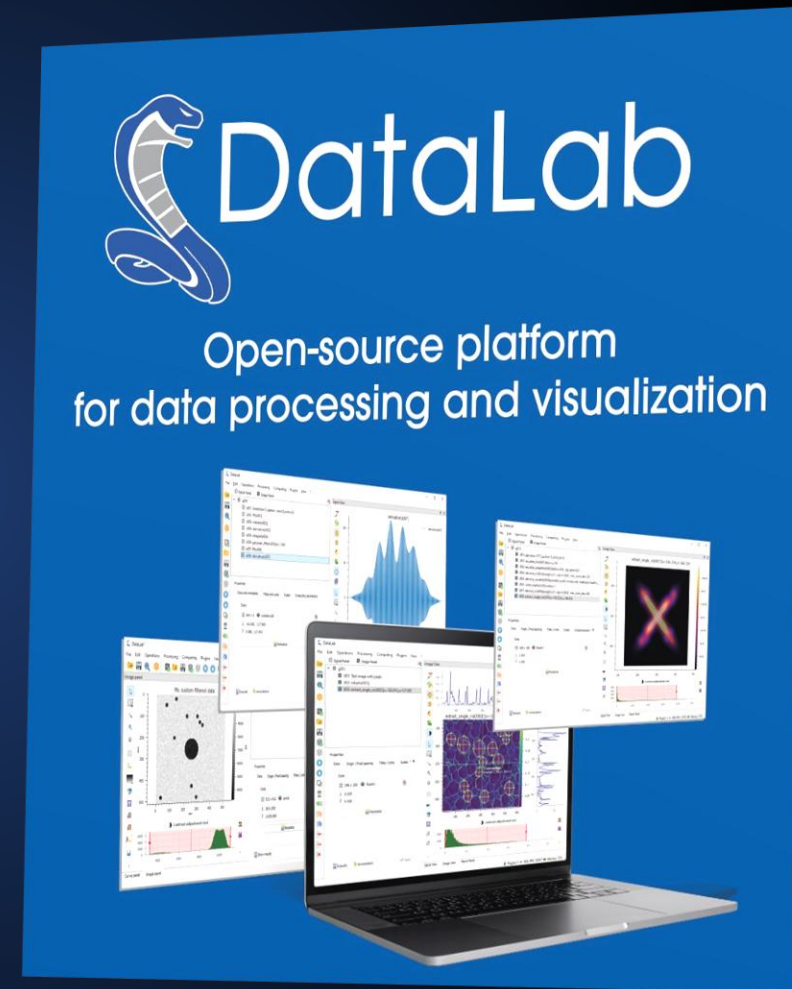


200+ integrated processing features

CODRA

DataLab in a nutshell

- Ergonomic
 - Interoperable (remote-control API)
 - Extensible (Python API: plugins, macros)
 - Robust (industrial-grade software)
-
- Already used in production (e.g. Laser Mégajoule)



DataLab: in research, industry and education...



Spectral measurement processing
(physical sensors)



Medical or microscopy image
analysis



Experimental data
visualization



R&D processing-chain
validation



Interactive debugging of
internal processing workflows

DataLab use-case categories

DataLab use-case categories



Analysis

Explore data, design and tune processing algorithms



Prototyping

Prototype ideas and algorithms rapidly



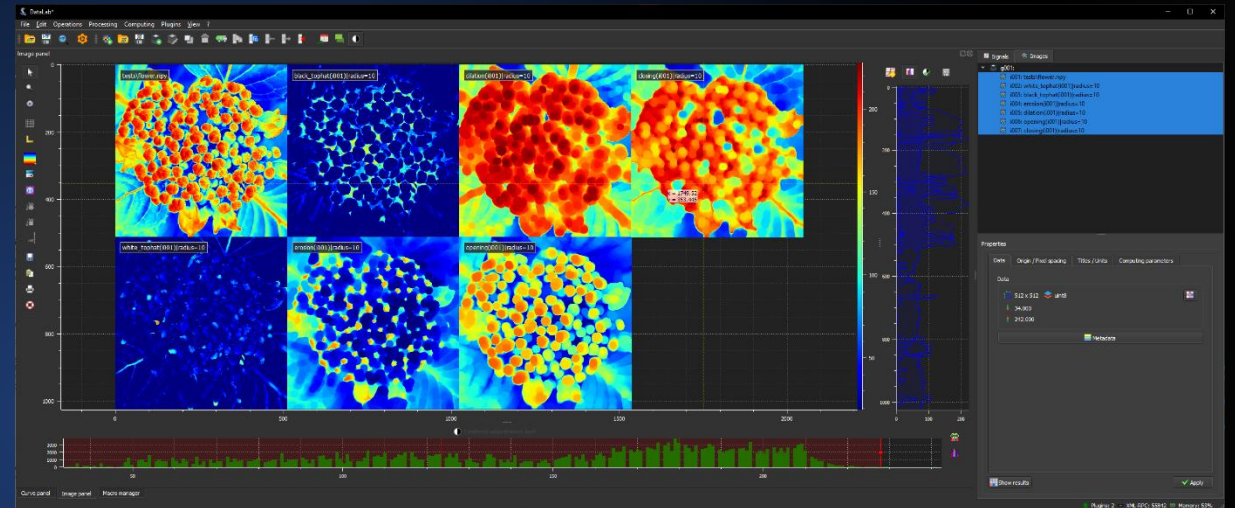
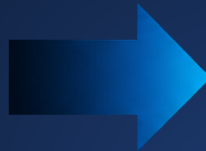
Advanced debugging

Inspect and debug internal processing workflows



Integration

Integrate DataLab in larger applications and workflows



Signal and image processing

DataLab use-case categories

DataLab use-case categories



Analysis

Explore data, design and tune processing algorithms



Prototyping

Prototype ideas and algorithms rapidly



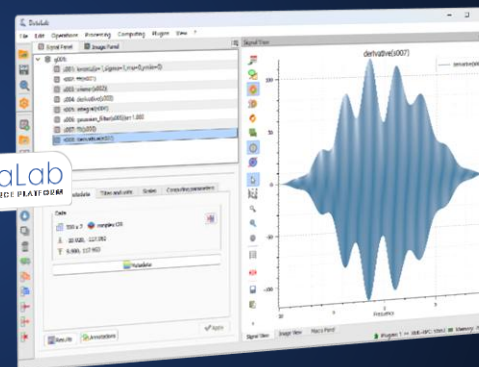
Advanced debugging

Inspect and debug internal processing workflows



Integration

Integrate DataLab in larger applications and workflows



```

[0]: import numpy as np
[1]: from cdfclient import SlepianSlepianProxy
[2]: proxy = SlepianSlepianProxy()
[3]:
[4]: Connecting to DataLab current session.
[5]:
[6]: proxy.connect()
[7]: Connecting to DataLab XRL-IPC server...OK (port: 62547)
[8]: Decoding commands in DataLab:
[9]:
[10]: x = np.random.randn(20, 20)
[11]: proxy.add_signal("x", x)
[12]:
[13]: True
[14]:
[15]: x = np.array([1.0, 2.0, 3.0])
[16]: y = np.array([4.0, 5.0, 6.0])
[17]: proxy.add_signal("y", x, y)
[18]:
[19]: True
[20]:
[21]: proxy.compute_derivative()
[22]:
[23]: True
  
```

Interact with your own code

DataLab use-case categories

DataLab use-case categories



Analysis

Explore data, design and tune processing algorithms



Prototyping

Prototype ideas and algorithms rapidly



Advanced debugging

Inspect and debug internal processing workflows

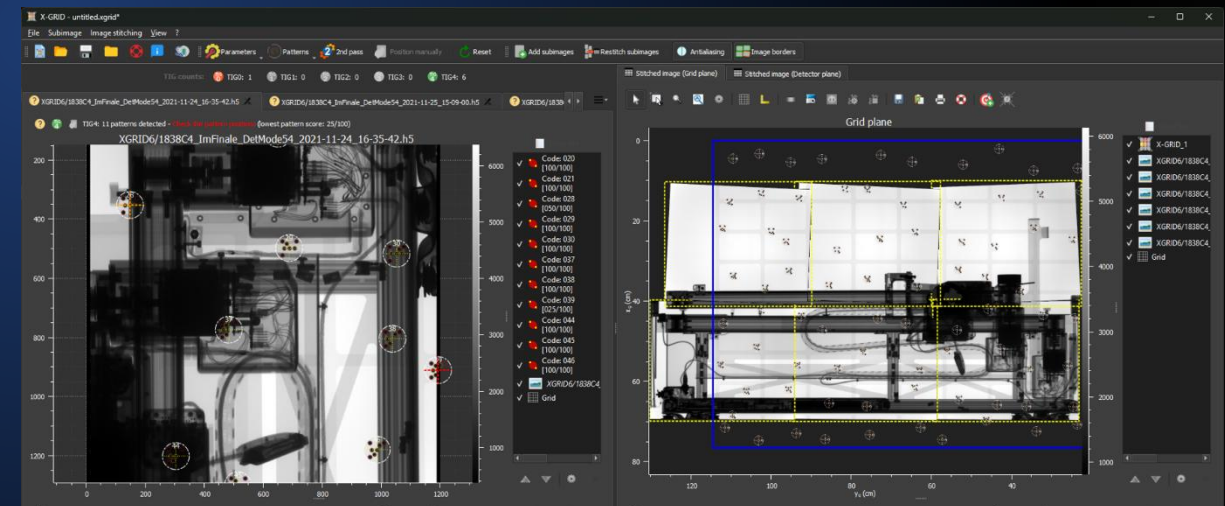


Integration

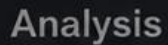
Integrate DataLab in larger applications and workflows



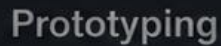
X-GRID application (that's today's topic!)



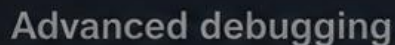
DataLab use-case categories



Explore data, design and tune processing algorithms



Prototype ideas and algorithms rapidly

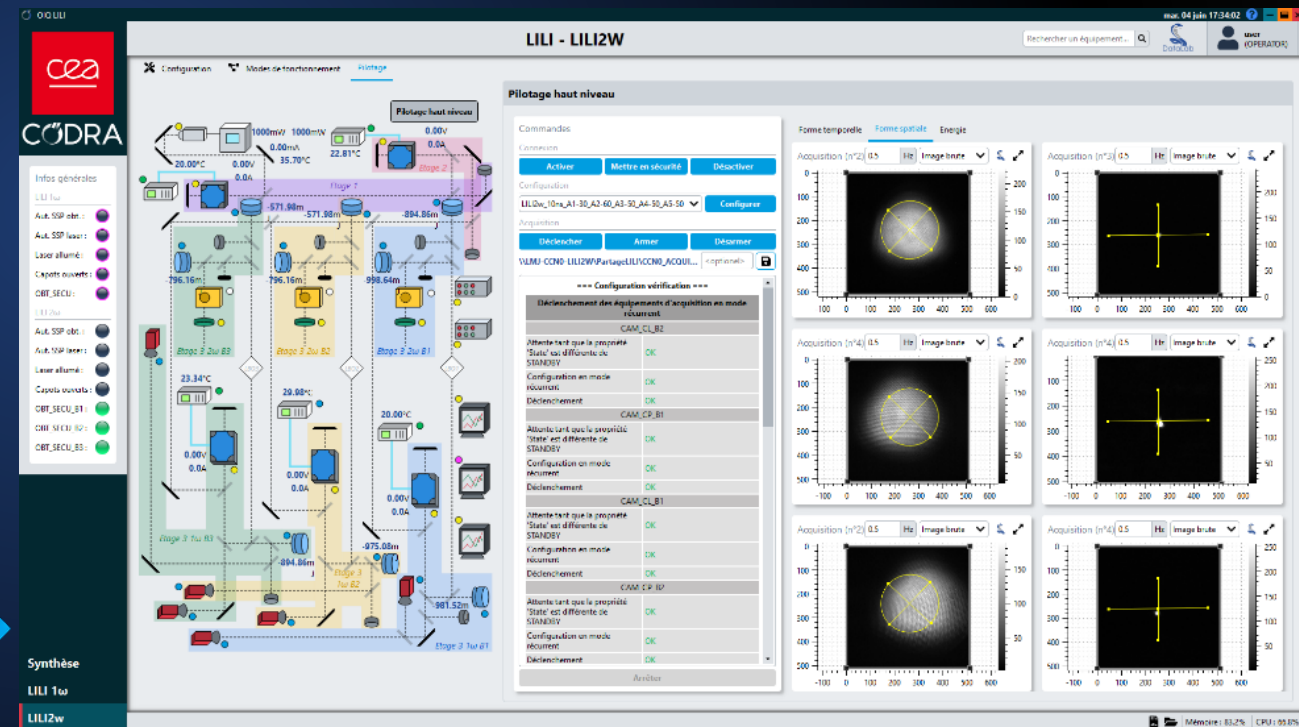


Inspect and debug internal processing workflows



Integrate DataLab in larger applications and workflows

Embedded in Laser Megajoule's control system



From DataLab use cases to X-GRID

DataLab use-case categories

1



Analysis

Explore data, design and tune processing algorithms



2



Prototyping

Prototype ideas and algorithms rapidly



Advanced debugging

Inspect and debug internal processing workflows



Integration

Integrate DataLab in larger applications and workflows



We will now see DataLab in action on this case study

Industrial case study

X-GRID

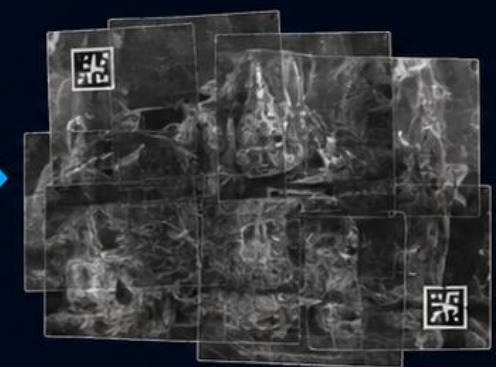
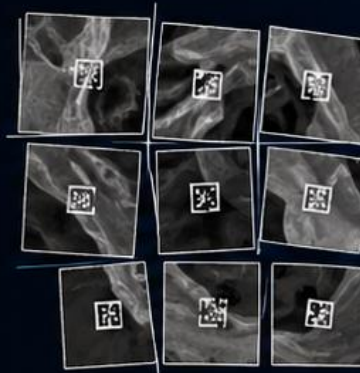
Automatic reconstruction of large X-ray scenes



X-ray image stitching based on automatic grid patterns detection



CODRA



X-GRID: context and motivation

- Use case: **radiography of large objects** → typically 1–2 meters wide
- **Field-deployable** acquisition → indoors, outdoors, wherever inspection is needed
- Mobile X-ray equipment → imaging plates or digital detector, typically 35 cm × 43 cm
- Consequence → the full scene cannot be captured in a single shot
- Need → reconstruct the complete radiographic scene from multiple smaller-field X-ray images
- **X-GRID** is an application specified and developed by CODRA for CEA, to automatically reconstruct complete radiographic scenes from partial X-ray images



X-ray image stitching based on automatic grid patterns detection



Contact @ CEA:

Marc Rabec le Gloahec

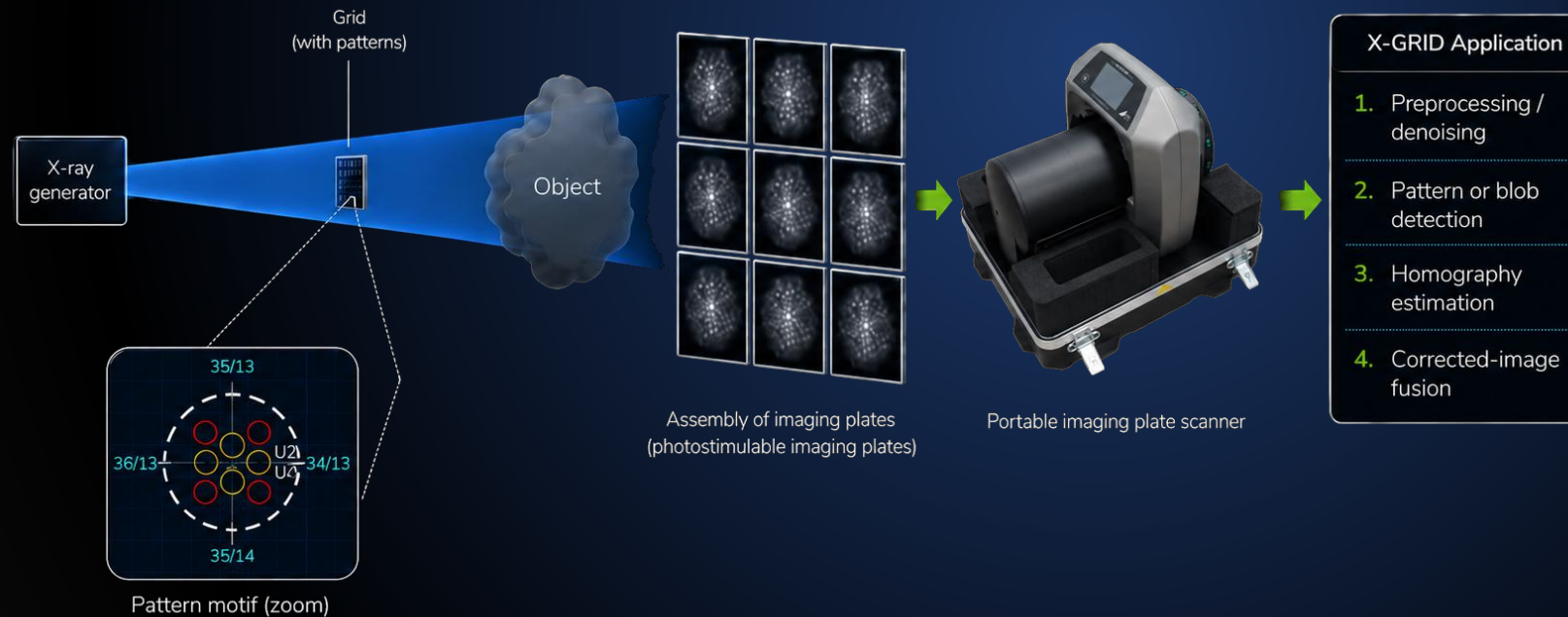


CODRA

CODRA

Reconstructing large X-ray scenes

- **X-GRID** is an application specified and developed by CODRA for CEA, dedicated to the automatic reconstruction of radiographic scenes from partial X-ray images



X-GRID
X-ray image stitching based on automatic grid patterns detection

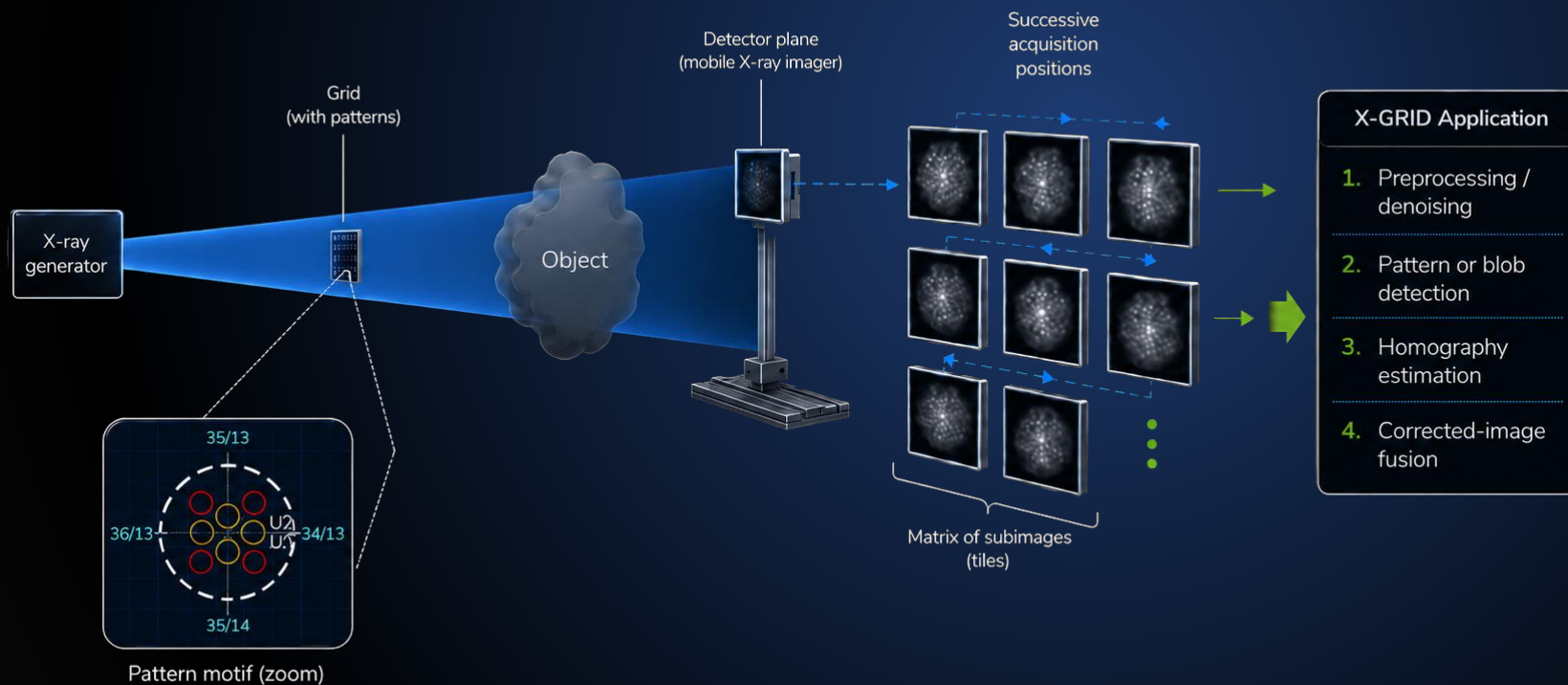


CODRA

CODRA

Reconstructing large X-ray scenes

- **X-GRID** is an application specified and developed by CODRA for CEA, dedicated to the automatic reconstruction of radiographic scenes from partial X-ray images



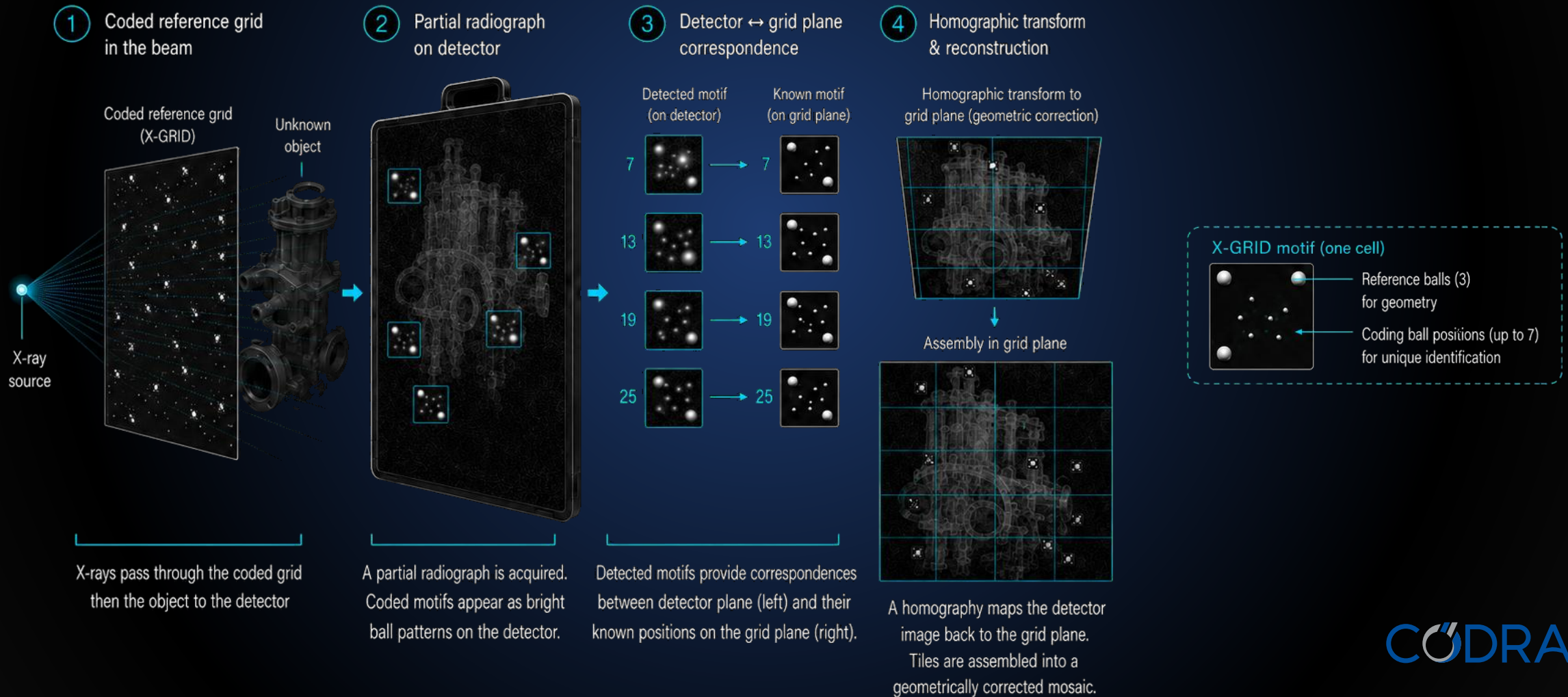
X-GRID
X-ray image stitching based on automatic grid patterns detection



CODRA

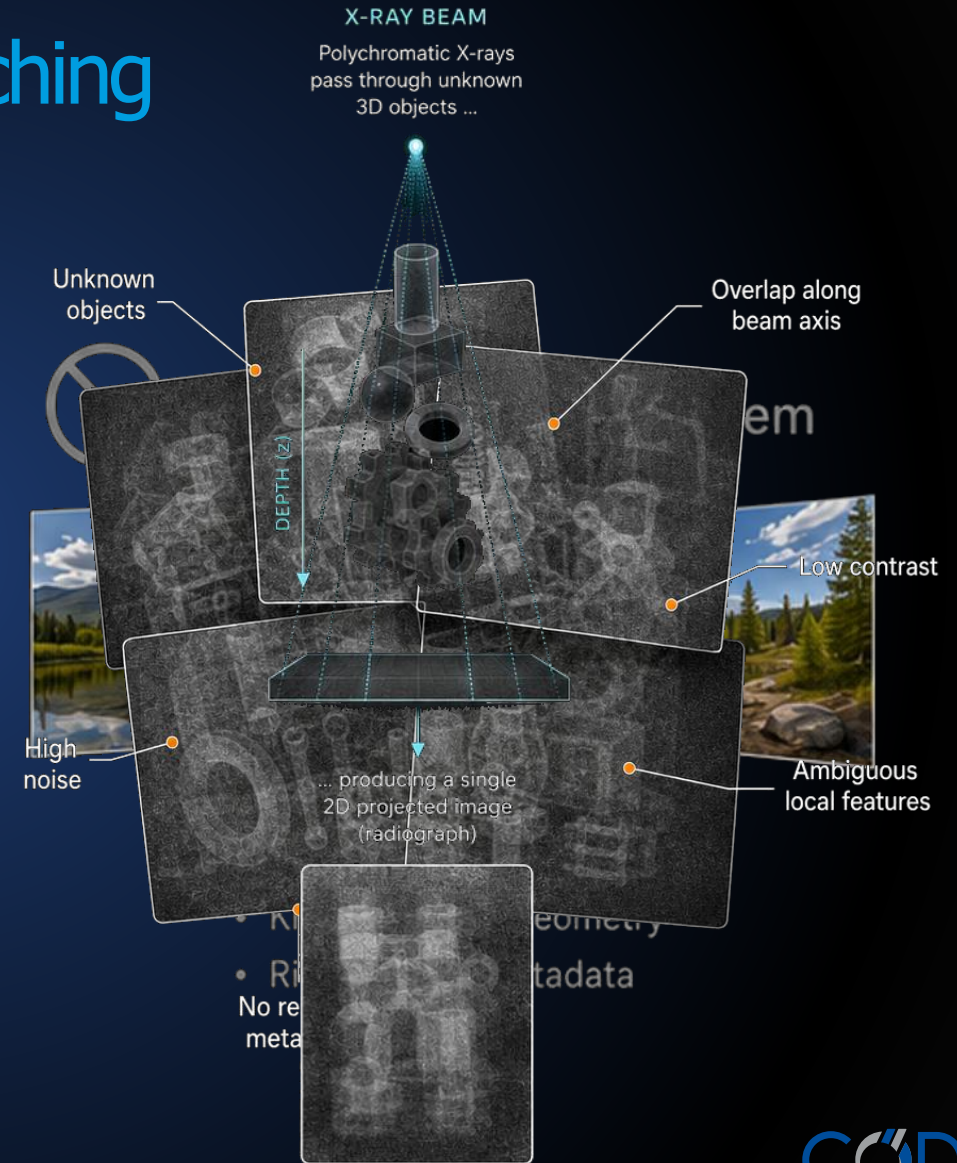
CODRA

The key idea: a coded reference grid



Why this is not standard image stitching

- X-ray images are not outdoor photographs
- They are projections through unknown objects, with structures superimposed along the beam axis
- Typical difficulties:
 - Low contrast
 - High noise
 - No natural texture continuity
 - No reliable acquisition metadata
- So this is not a panorama problem
- It requires a dedicated reconstruction approach

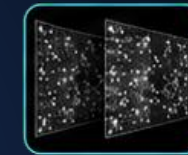


From physical principle to software reconstruction

- The coded grid makes reconstruction possible
- The software challenge is:
 - To go from **raw radiographs** to a **geometrically corrected mosaic**:
 - Detect/decode
 - Estimate homography
 - Correct/assemble
 - And to do it **fully automatically**:
 - No manual positioning
 - No operator tuning
 - No prior knowledge of the scene
- This is where CODRA developed **X-GRID**
— and where **DataLab** became essential



Fully automatic



Raw
radiographs
only



No manual
positioning



No operator
tuning



No prior
knowledge of
the scene

About X-GRID application

- X-GRID is a  python™ desktop application that strongly leverages on:

- Qt for modern user interfaces



- Powerful open-source processing libraries:
SciPy, **scikit-image**, **OpenCV**



- **PlotPyStack** libraries (guidata, PlotPy)
for efficient 2D visualization



<https://github.com/PlotPyStack>

The only practical solution for working smoothly with images this large:

- Each sub-image can be up to 8600 pixels × 7000 pixels × 16 bits ~120MB (uncompressed data in memory)
- A typical assembly contains 4 to 8 sub-images

- **DataLab** for prototyping
and advanced debugging (remote control)



<https://datalab-platform.com>

DataLab's Use case 1 – Building X-GRID processing pipeline

DataLab use-case categories

1



Analysis

Explore data, design and tune processing algorithms



Prototyping

Prototype ideas and algorithms rapidly



2



Advanced debugging

Inspect and debug internal processing workflows



Integration

Integrate DataLab in larger applications and workflows

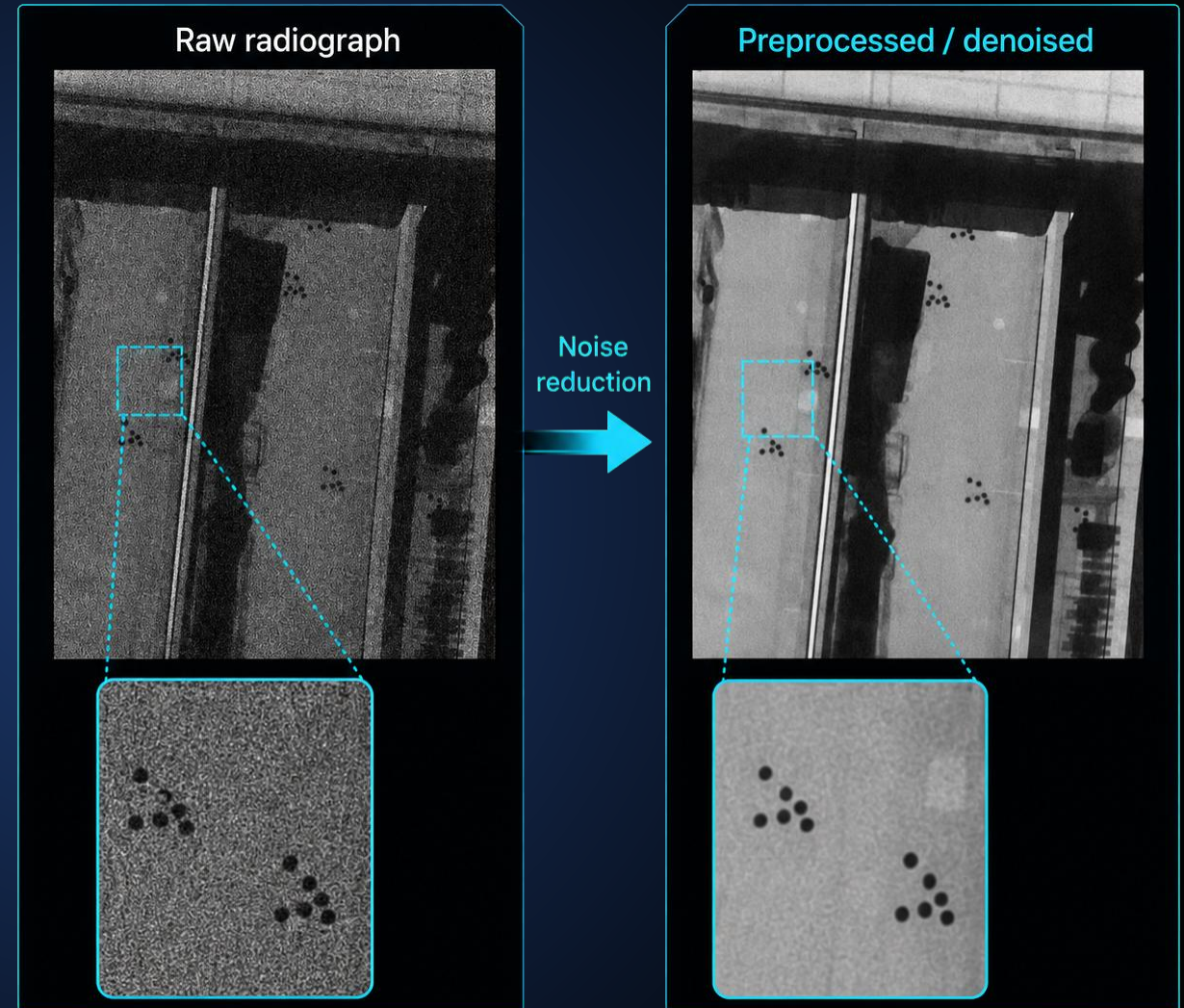


How **DataLab** helped building X-GRID processing pipeline

DataLab's Use case 1

Pipeline: from Raw radiographs to geometrically corrected mosaic

1. Preprocessing / denoising
2. Coded-pattern detection & decoding
3. Homography estimation
4. Geometric correction & image fusion



What DataLab made possible

DataLab made it easy to rapidly explore and tune a denoising strategy optimized for robust blob detection.



scikit-image

`exposure.equalize_hist`
`restoration.denoise_bilateral`
`restoration.denoise_tv_chambolle`



SciPy

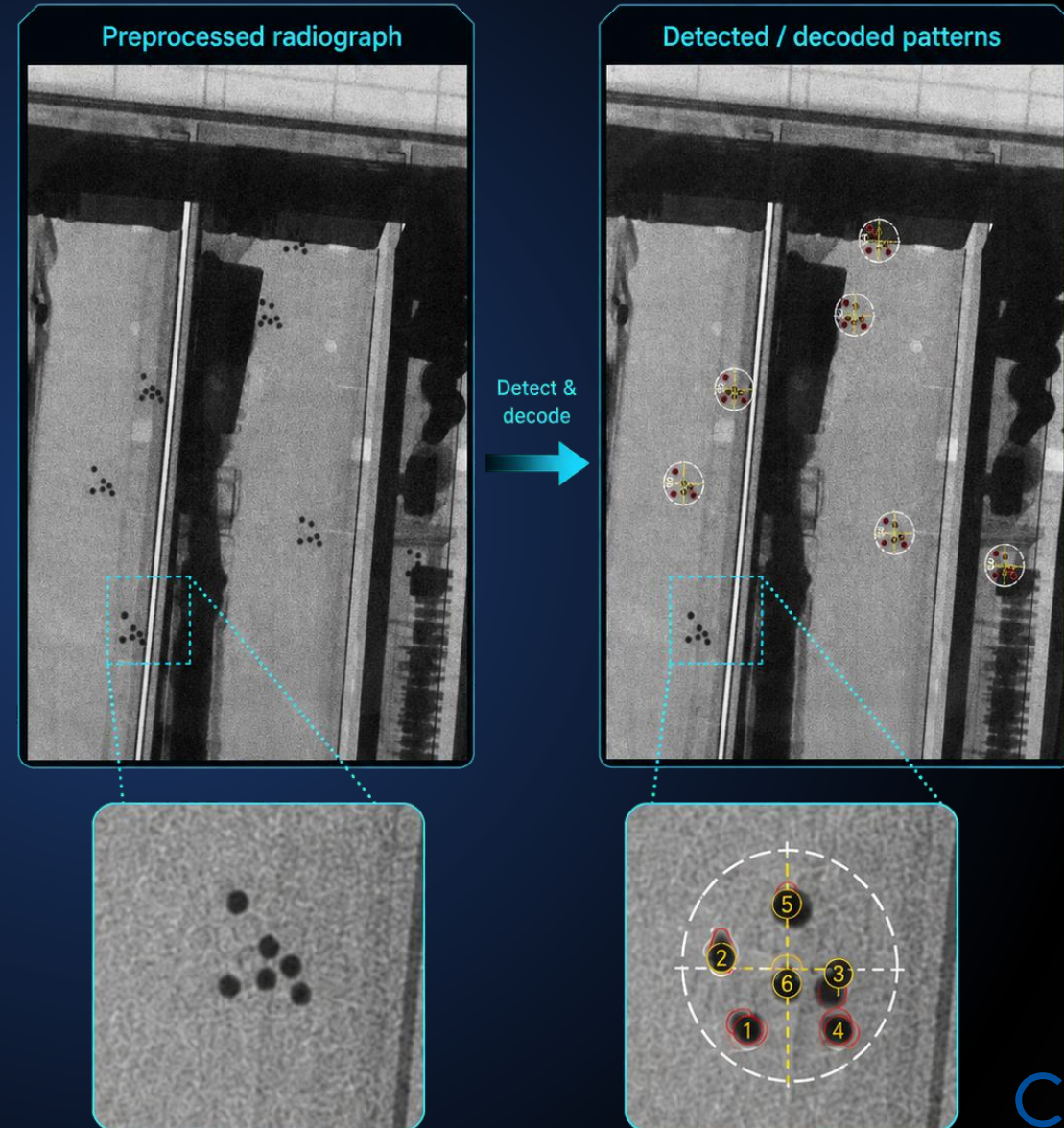
`ndimage.median_filter`
`ndimage.uniform_filter`

Easy comparison of variants, interactive parameter tuning, and combination of operators from multiple open-source libraries.

DataLab's Use case 1

Pipeline: from Raw radiographs to geometrically corrected mosaic

1. Preprocessing / denoising
2. Coded-pattern detection & decoding
3. Homography estimation
4. Geometric correction & image fusion



What DataLab made possible

DataLab helped tune blob-detection parameters, select OpenCV SimpleBlobDetector, and design the full coded-pattern detection algorithm.



OpenCV SimpleBlobDetector
blob detector selection and parameter tuning



Remote control



Advanced debugging



Design of the full coded-pattern detection algorithm

DataLab made it possible to go beyond isolated blob detection and iteratively build the complete coded-pattern detection and decoding workflow.

DataLab's Use case 1

Pipeline: from Raw radiographs to geometrically corrected mosaic

1. Preprocessing / denoising
2. Coded-pattern detection & decoding
3. Homography estimation
4. Geometric correction & image fusion



What DataLab made possible

DataLab helped tune geometric-transformation parameters, select OpenCV findHomography, and design the full homography-estimation workflow.



OpenCV
findHomography
projective transform estimation



Remote control



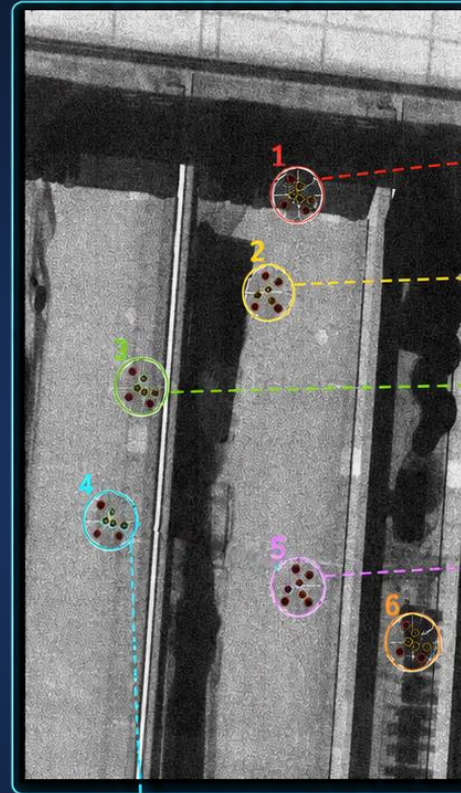
Advanced debugging



Design of the full homography-estimation workflow

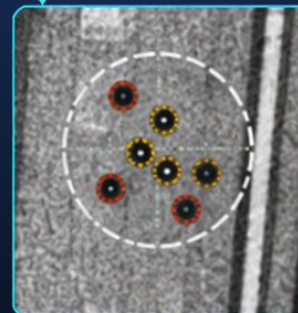
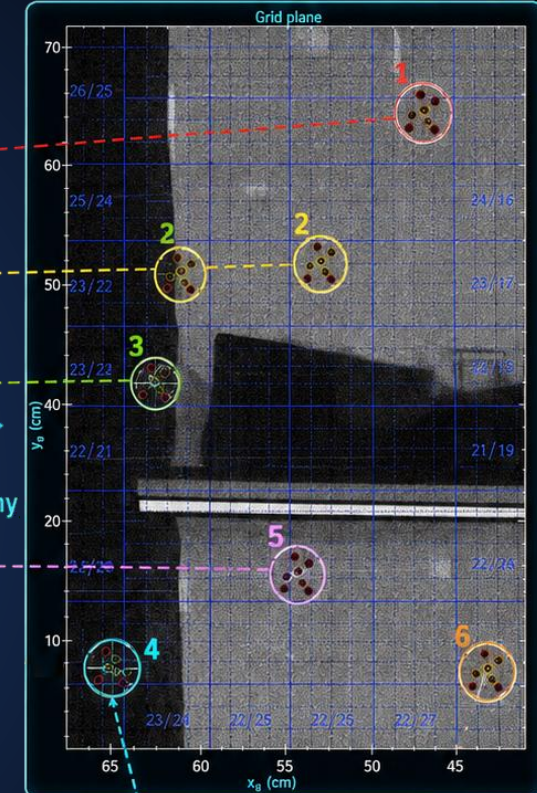
DataLab made it possible to go beyond isolated OpenCV calls and iteratively build the complete homography-estimation workflow.

DETECTOR-PLANE RADIOGRAPH



Estimate homography

GRID-PLANE REFERENCE



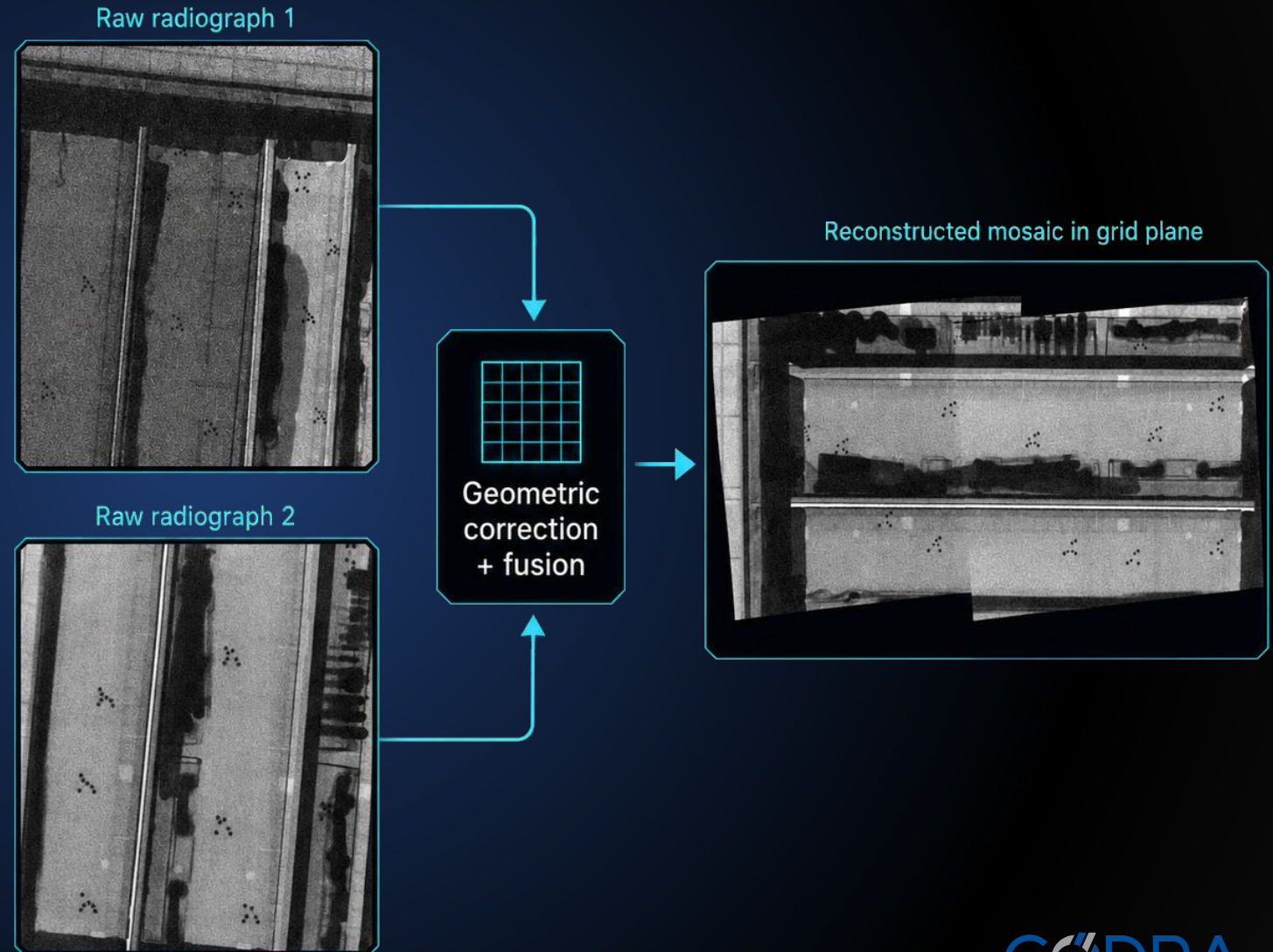
Zoom: ID 4



DataLab's Use case 1

Pipeline: from Raw radiographs to geometrically corrected mosaic

1. Preprocessing / denoising
2. Coded-pattern detection & decoding
3. Homography estimation
4. Geometric correction & image fusion



What DataLab made possible

DataLab helped tune geometric-correction parameters, leverage OpenCV warpPerspective, and design the full geometric-correction and image-fusion workflow.



OpenCV warpPerspective

perspective warping into the grid plane and geometric correction



Remote control



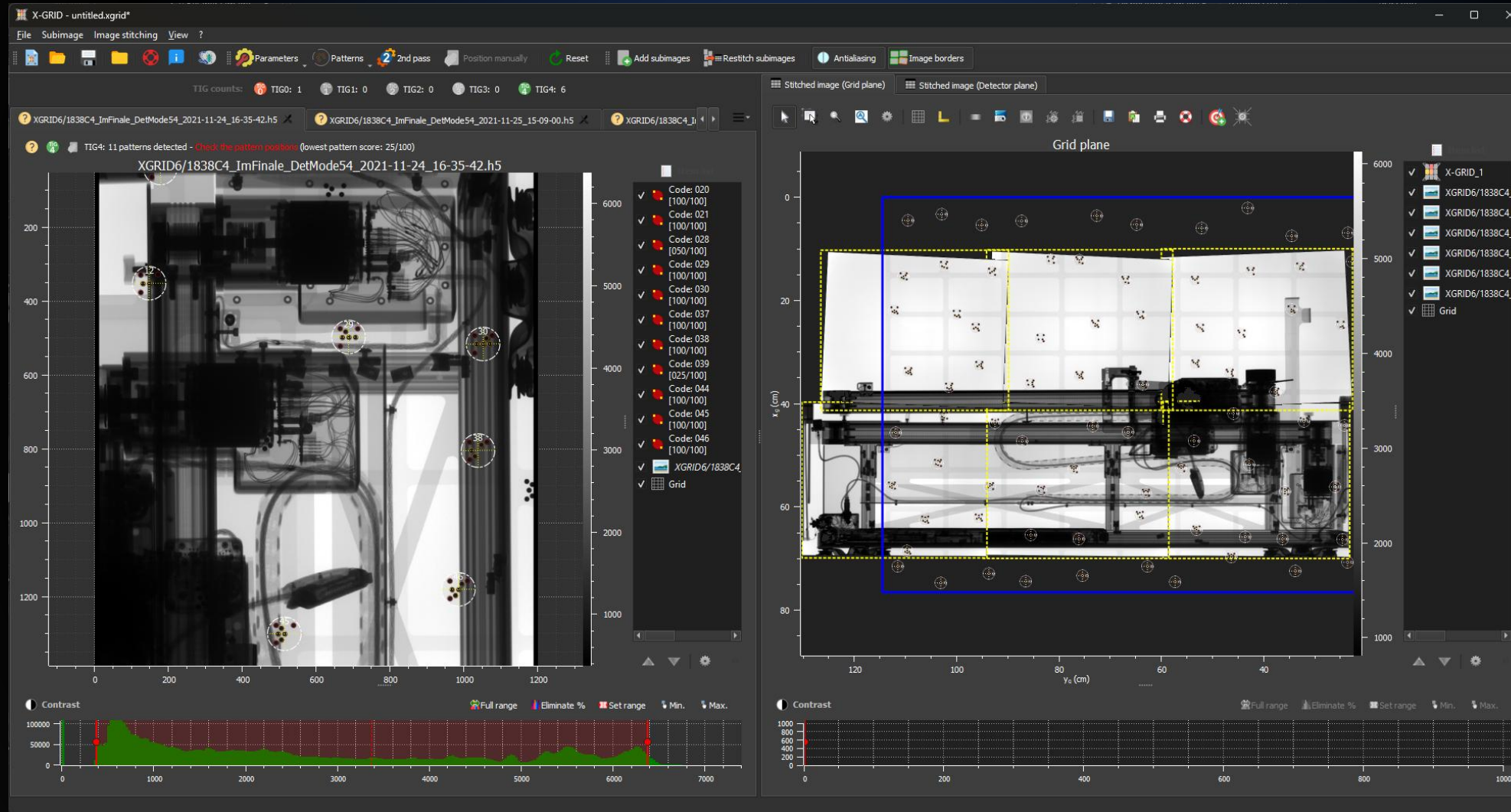
Advanced debugging



Design of the full geometric-correction and image-fusion workflow

DataLab made it possible to go beyond isolated OpenCV calls and iteratively build the complete geometric-correction and image-fusion workflow.

Demo time: X-GRID in action



DataLab's Use case 2 – Advanced debugging inside X-GRID

DataLab use-case categories

1



Analysis

Explore data, design and tune processing algorithms



Prototyping

Prototype ideas and algorithms rapidly



2



Advanced debugging

Inspect and debug internal processing workflows



Integration

Integrate DataLab in larger applications and workflows



Once the first algorithm was implemented in X-GRID, **DataLab** became a live inspection and debugging companion

Classical debugger

Code (xgrid_processing.py)

```

21 def process_image(image, n);
22     acc = 0.0
23     if cond:
24         compute(image)
25         for i in range(n):
26             acc += process(buf[i])
27             update(result)
28     return result

```

Stack

process_image()
detect_patterns()
main()
<module>()

Variables

acc	= 0.345
i	= 42
n	= 256
ptr	= <memory ...>

Arrays

12	7	3	91	...
4	0	8	16	...
23	5	9	11	...

References

image	<ndarray>
buf	<ndarray>
result	<ndarray>

Breakpoints

xgrid_processing.py:24	☒
xgrid_processing.py:42	☐

Running in X-GRID



Processing pipeline



Live inspection of intermediate processing results

Remote control

DataLab-connected debugger

X-GRID (Application)

Acquire radiograph ●

Preprocess ●

Detect coded features ●

Estimate geometry ●

Reconstruct mosaic ●

Update results ●

● Running

DataLab (Live inspection)

Intermediate images Annotations ROIs Detections Transformations

Raw radiograph

Denoised / Enhanced

Coded-pattern detections

ROIs

Transformations (grid plane)

Result / Overlay (mosaic on radiograph)

Timeline (processing steps)

Remote-controlling DataLab from X-GRID

- Sigima (DataLab's engine) provides a remote control client for DataLab:



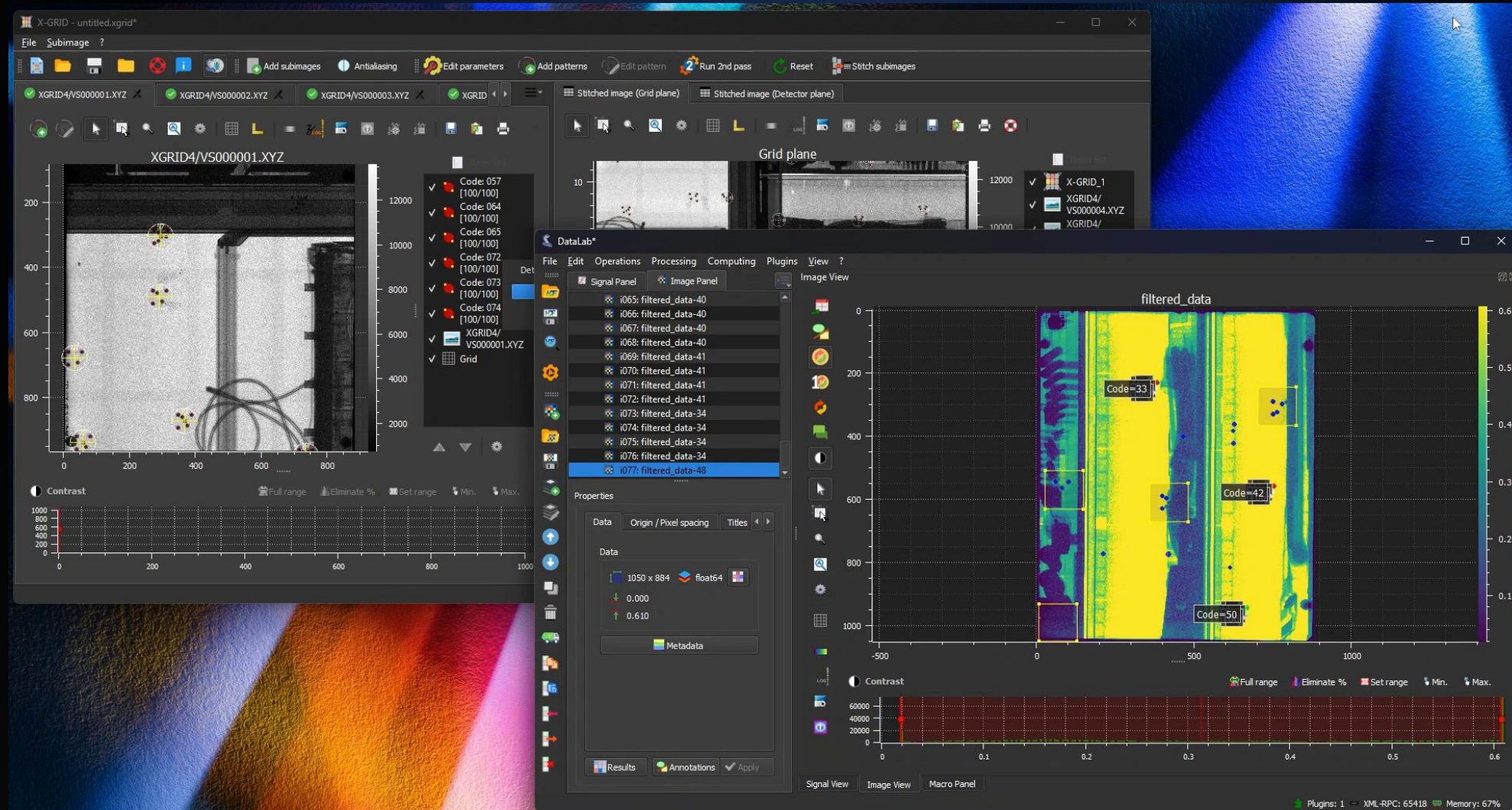
```
# DataLab remote control client:
from sigima.client import SimpleRemoteProxy
```

- At any breakpoint, X-GRID can send intermediate data to DataLab for interactive inspection:



```
107 class Grid:
118     def __init__(
129         self:
130         self.dl_proxy = SimpleRemoteProxy()
131         self.dl_proxy.connect(timeout=1.0, retries=1)
132         except Exception: # pylint: disable=broad-except
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
```

Demo: X-GRID running, DataLab observing



Takeaways

- 1. Scientific Python can solve hard industrial imaging problems**
X-GRID turns noisy partial X-ray images into a geometrically corrected mosaic
- 2. DataLab accelerates algorithm design**
Interactive experimentation makes it easier to test, combine and tune processing ideas
- 3. DataLab bridges exploration and production**
Remote control brings interactive inspection directly into production software



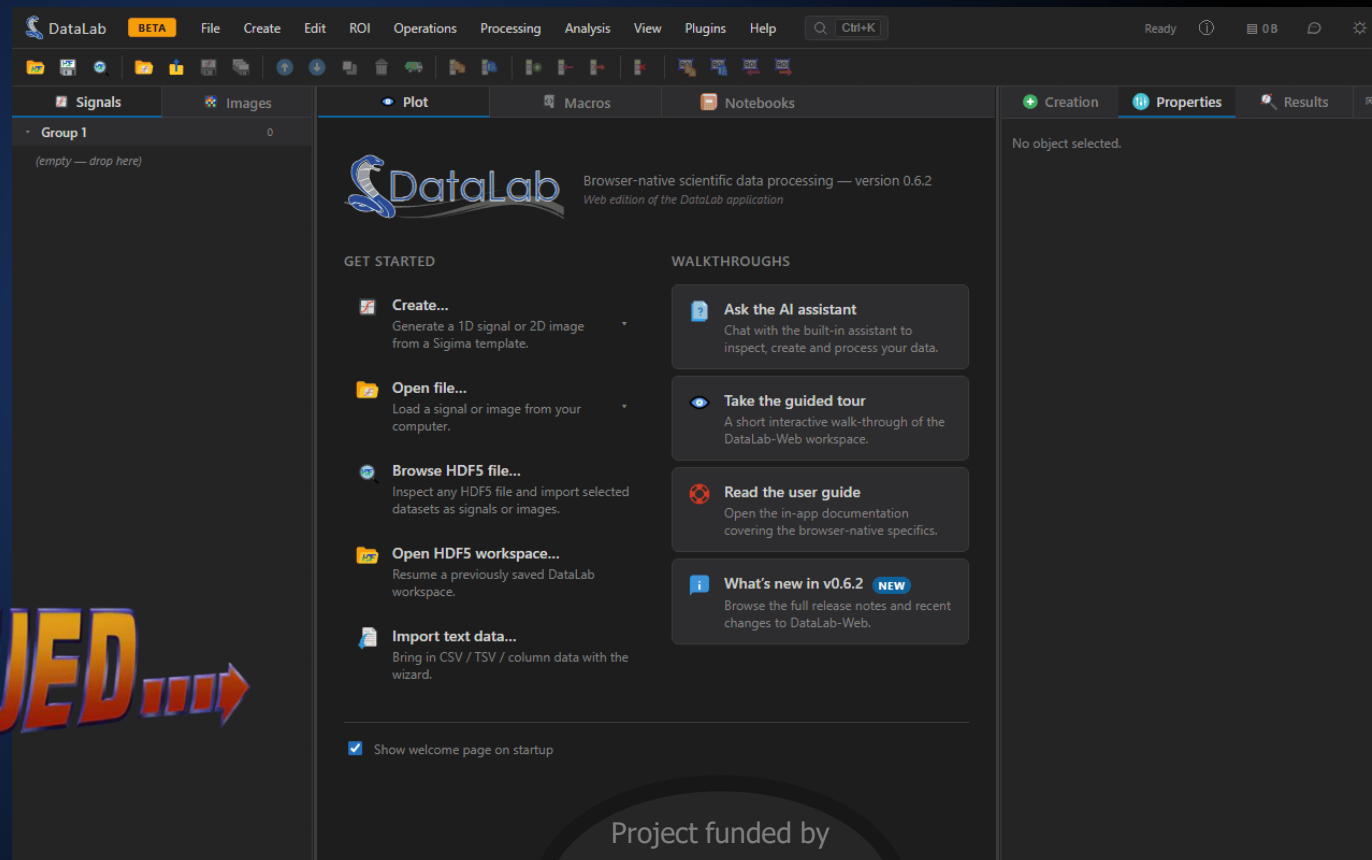
See it. **Understand it.** **Improve it.**

Beyond desktop: DataLab Web

- All DataLab features, accessible from a browser
- Local execution, no backend, no upload
- SDK TypeScript for embedding and remote control



TO BE
CONTINUED...➡



Available since June 2026:

<https://datalab-platform.com/web>



CODRA



Pierre Raybaut

Executive Vice President, Engineering.
Creator of Spyder (the scientific Pyth...



Thank you!



EuroSciPy 2026