

AFWERX CHALLENGE

Advanced Microelectronics Design and Prototype
Challenge

Thomas Catalano

github@tomsnode

Enhanced Ground Moving Target Indicator (GMTI) with added forensic analysis on a 14nm SoC.





2.4 Role of the Simulations in the Overall MAJEX07 Architecture

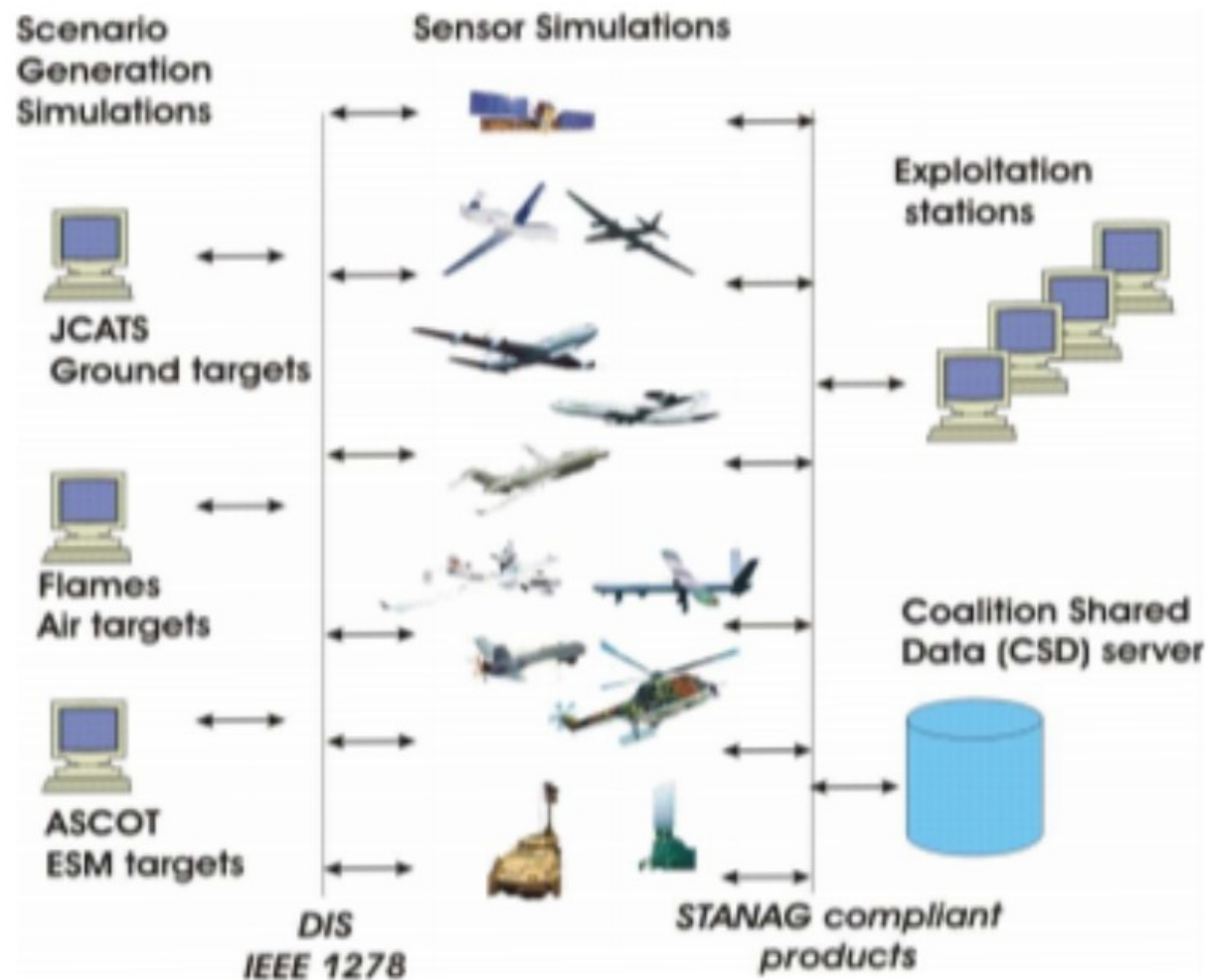


Figure 1: MAJEX07 architecture

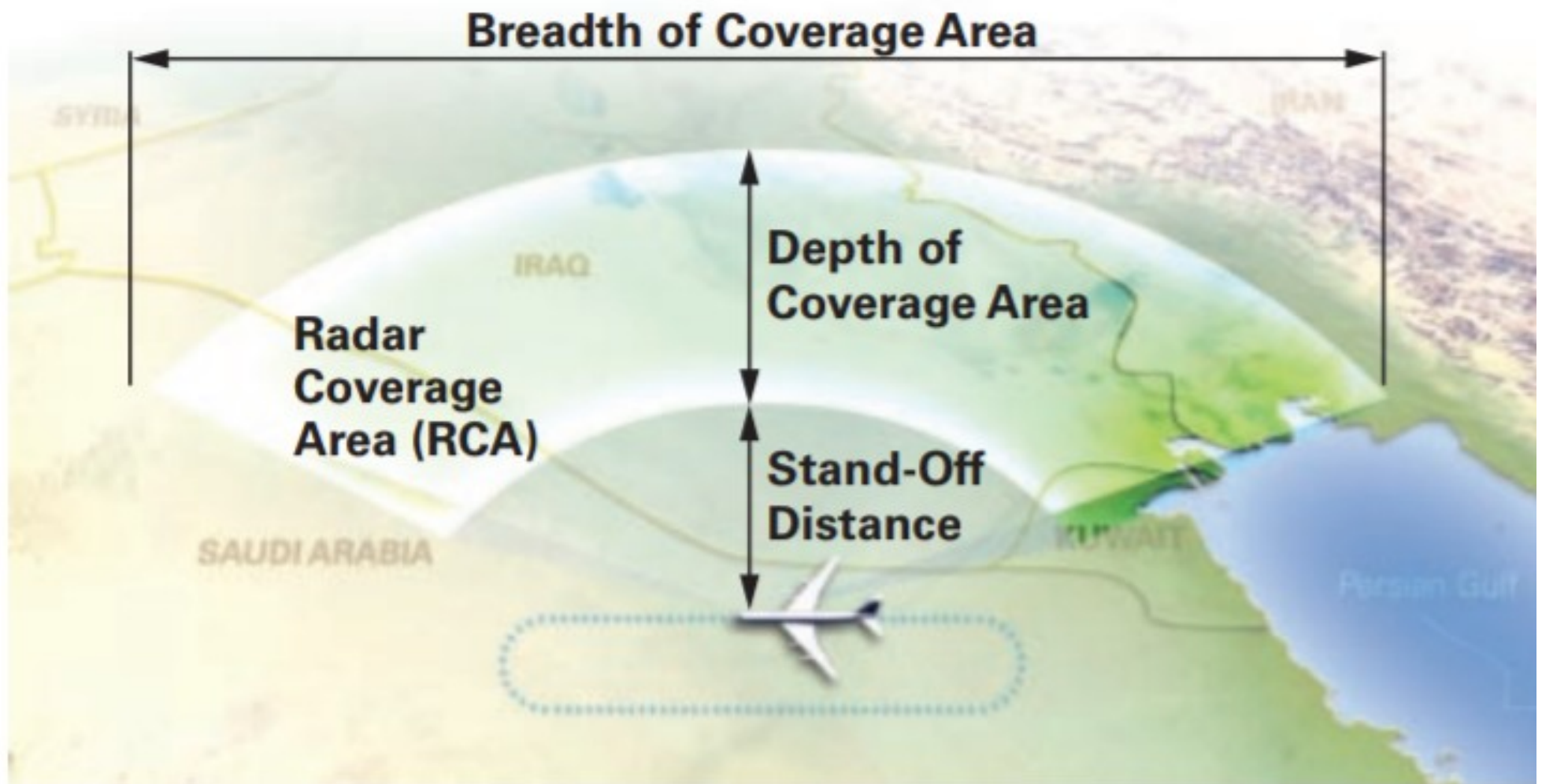
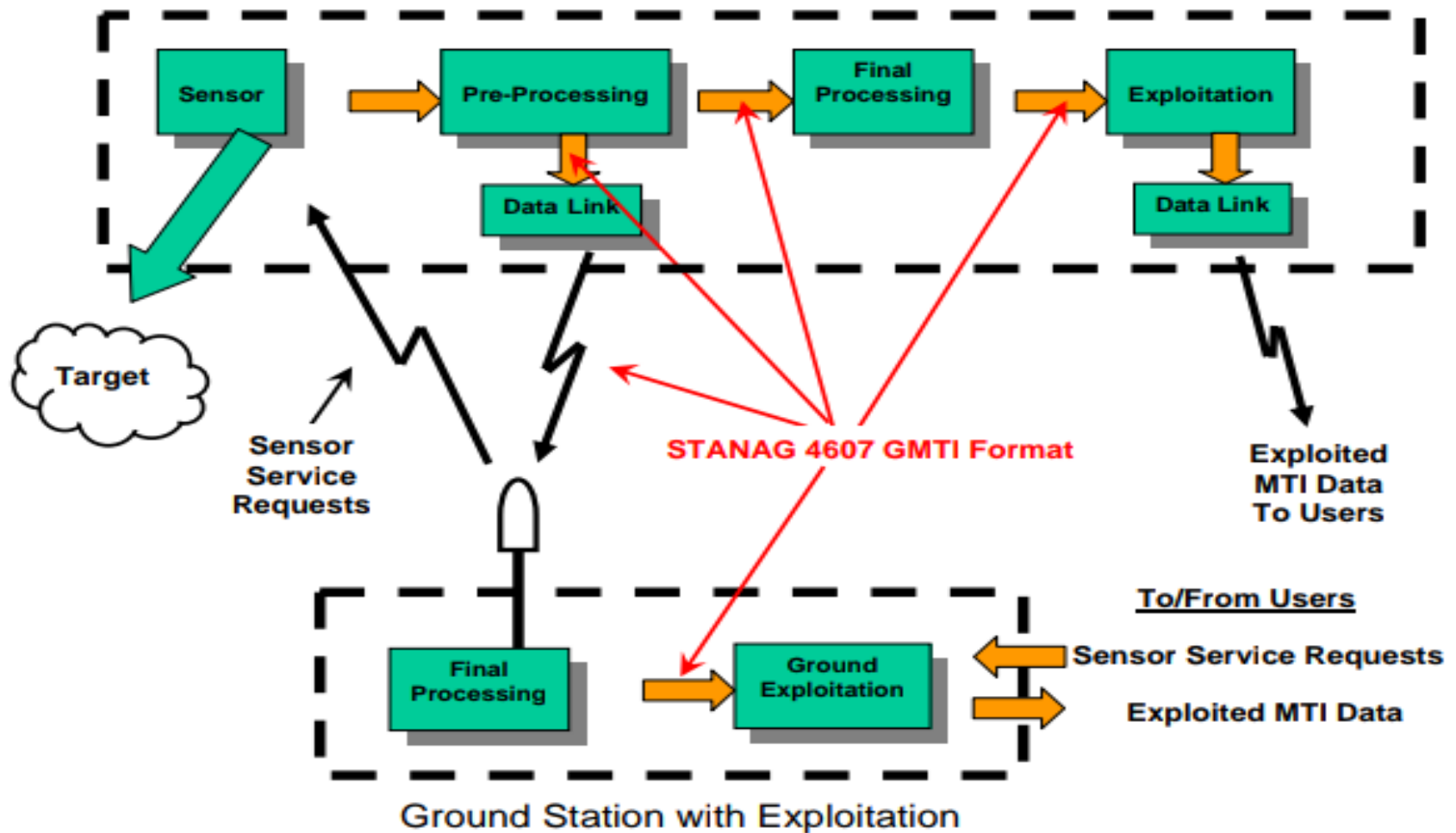


Figure A-2. Stand-off Distance and Radar Coverage Area

Sensor Platform with Exploitation



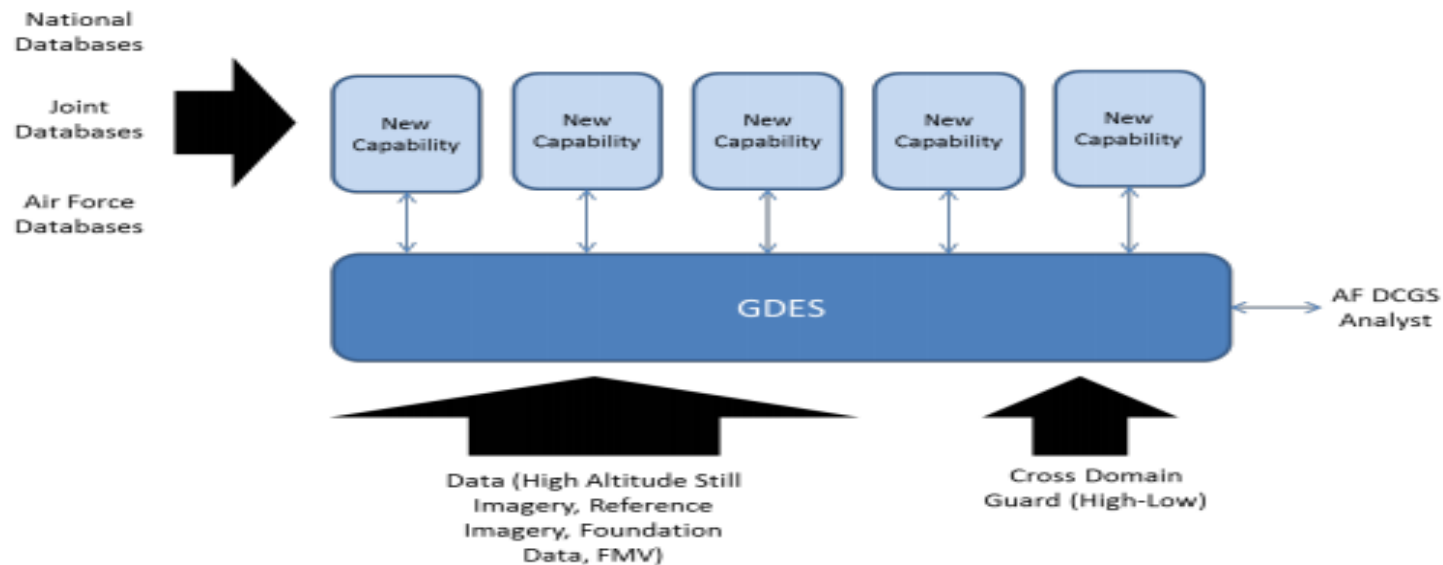
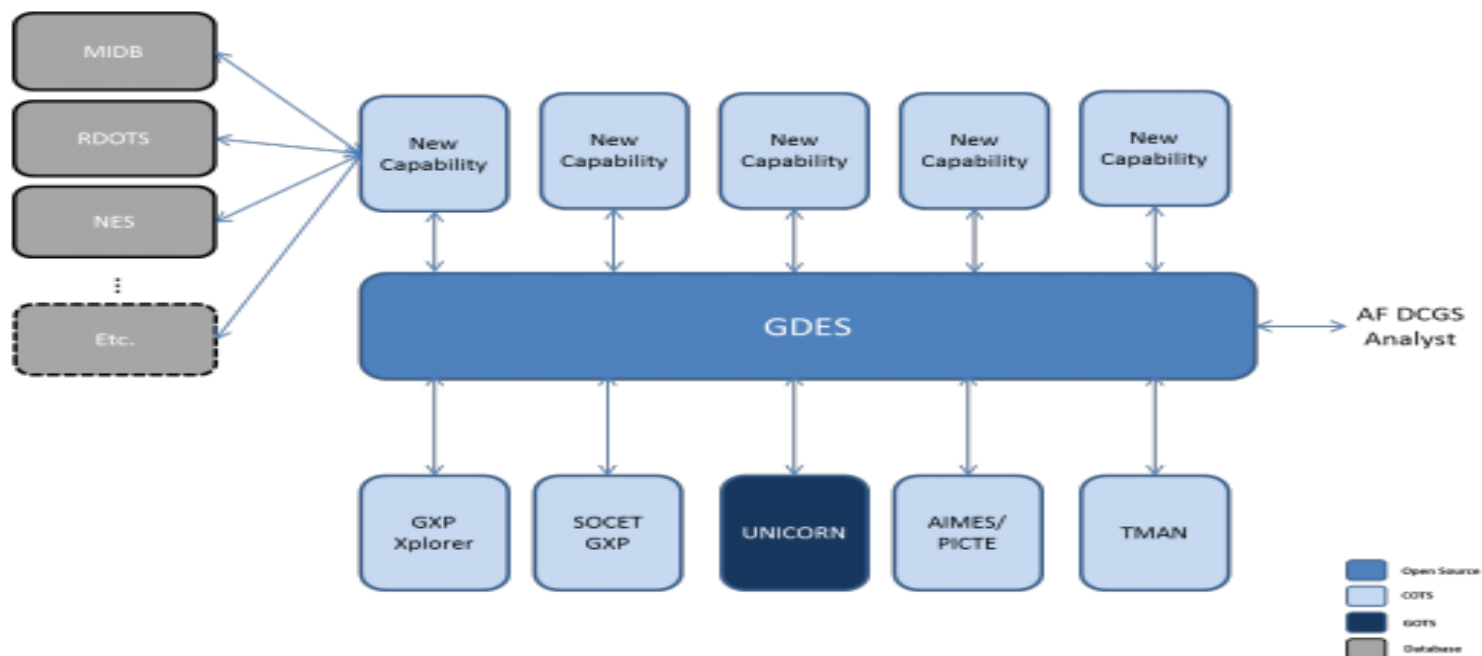


Figure 1. Proposed data flow for initiative



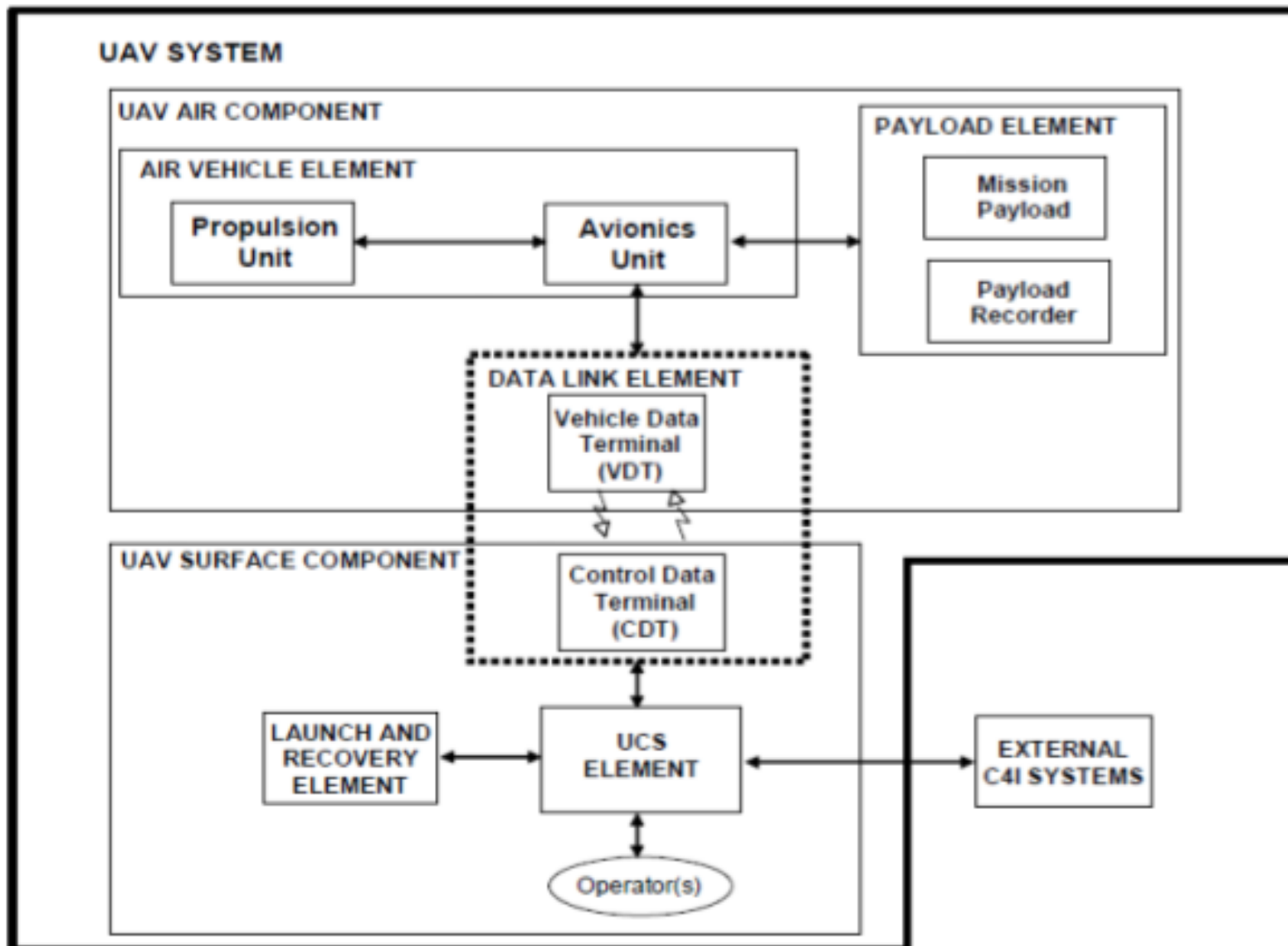


Figure 2: UAV System Elements [4].

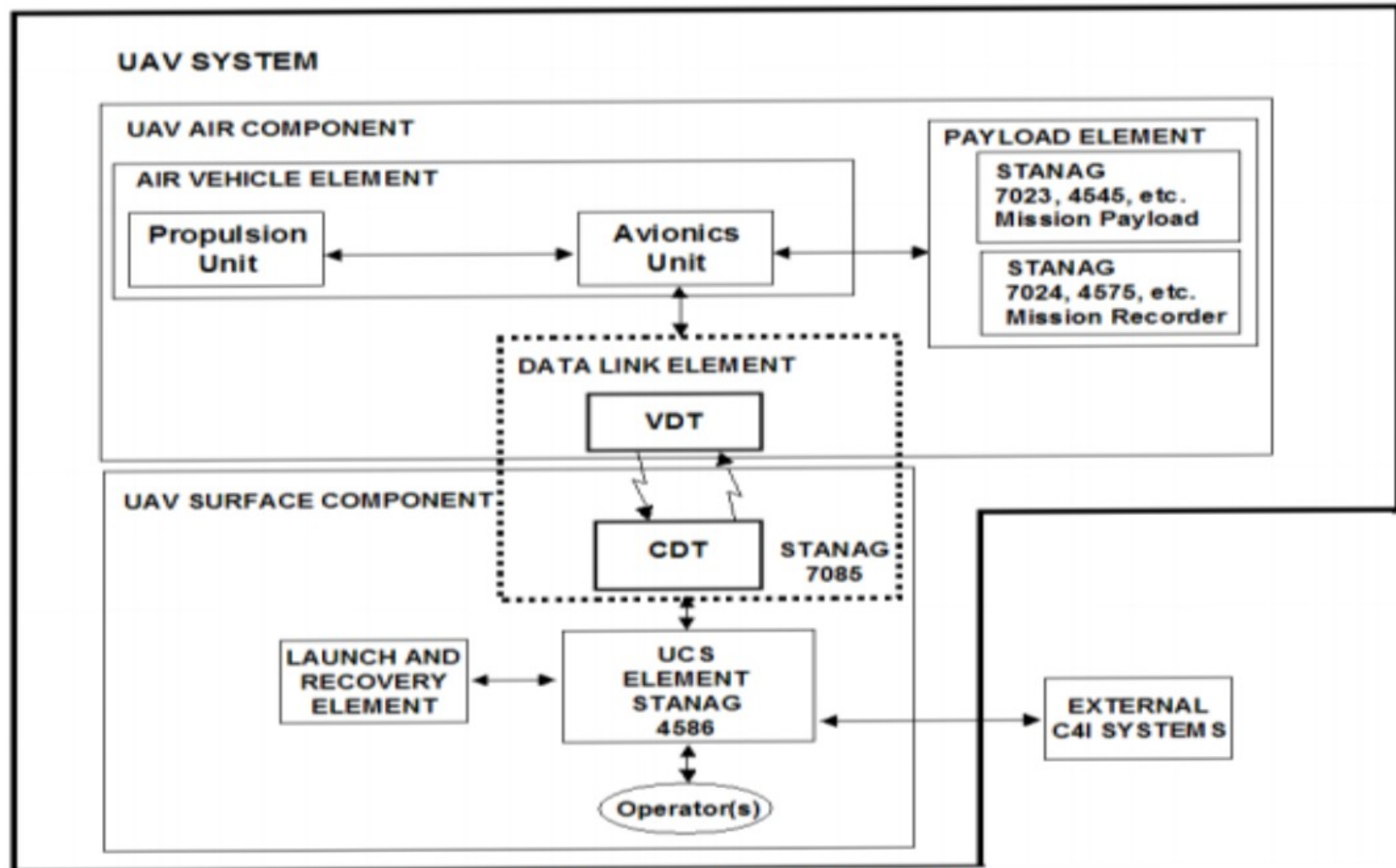


Figure 4: UAV Interoperability Architecture [4].

CLAW BENEFITS

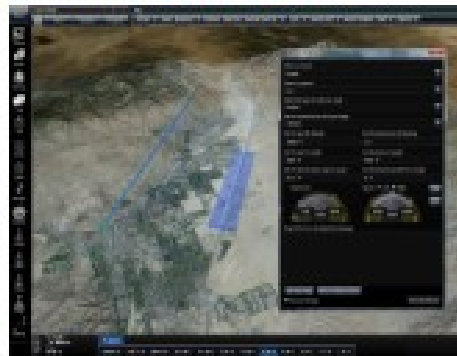
Comprehensive Dynamic Situational Awareness

ADVANCED MISSION PLANNING



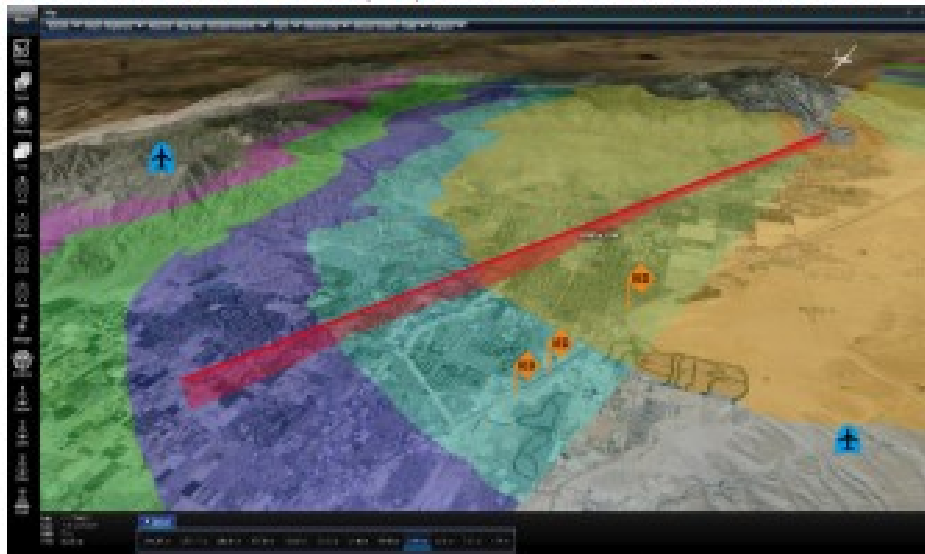
- Intuitive user interface supports menu-driven mission planning and cross-cuing
- Missions developed directly within 3D environment

AUTOMATED SENSOR CONTROL



- Integrated sensor automation and control facilitates rapid mission generation
- Dynamic map footprint provides interactive sensor control

COMMON OPERATIONAL PICTURE (COP) FEEDS AND REAL-TIME SENSOR OVERLAYS

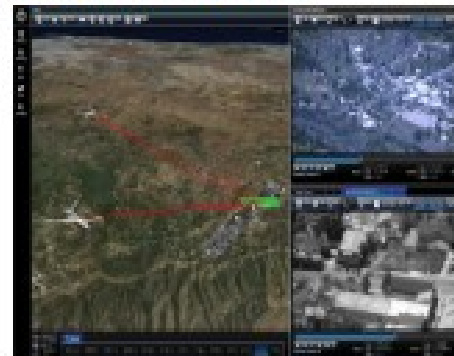


- Enhanced 3D map displays real-time auxiliary feed tracking
- Sensor overlays and aircraft symbology provide improved situational awareness

EXPLOITATION AND DISSEMINATION

Image Data Manipulation

VIDEO MANAGEMENT SUITE



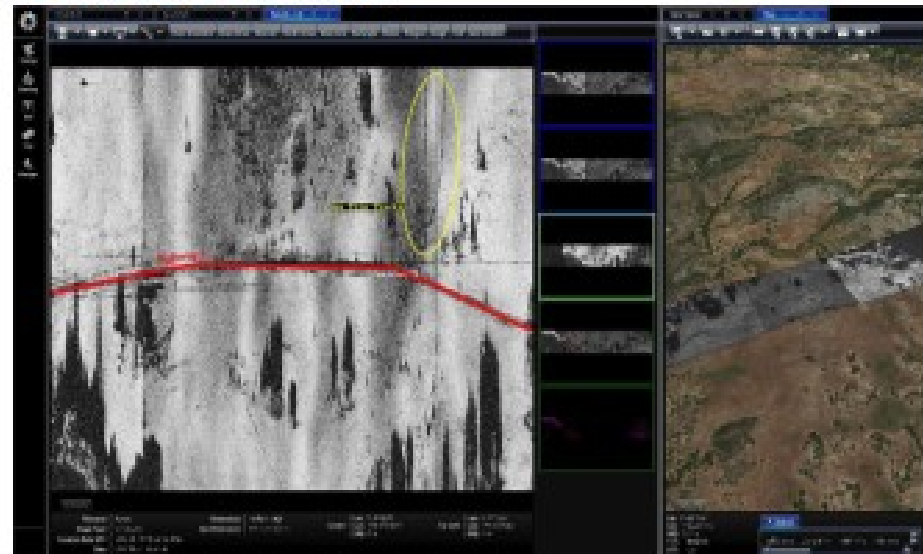
- Simultaneous multiple sensor footprints and feeds increase real-time intelligence product analysis
- Video recording, playback, and annotation tools augment exploitation capabilities

FORENSIC ARCHIVAL AND RETRIEVAL



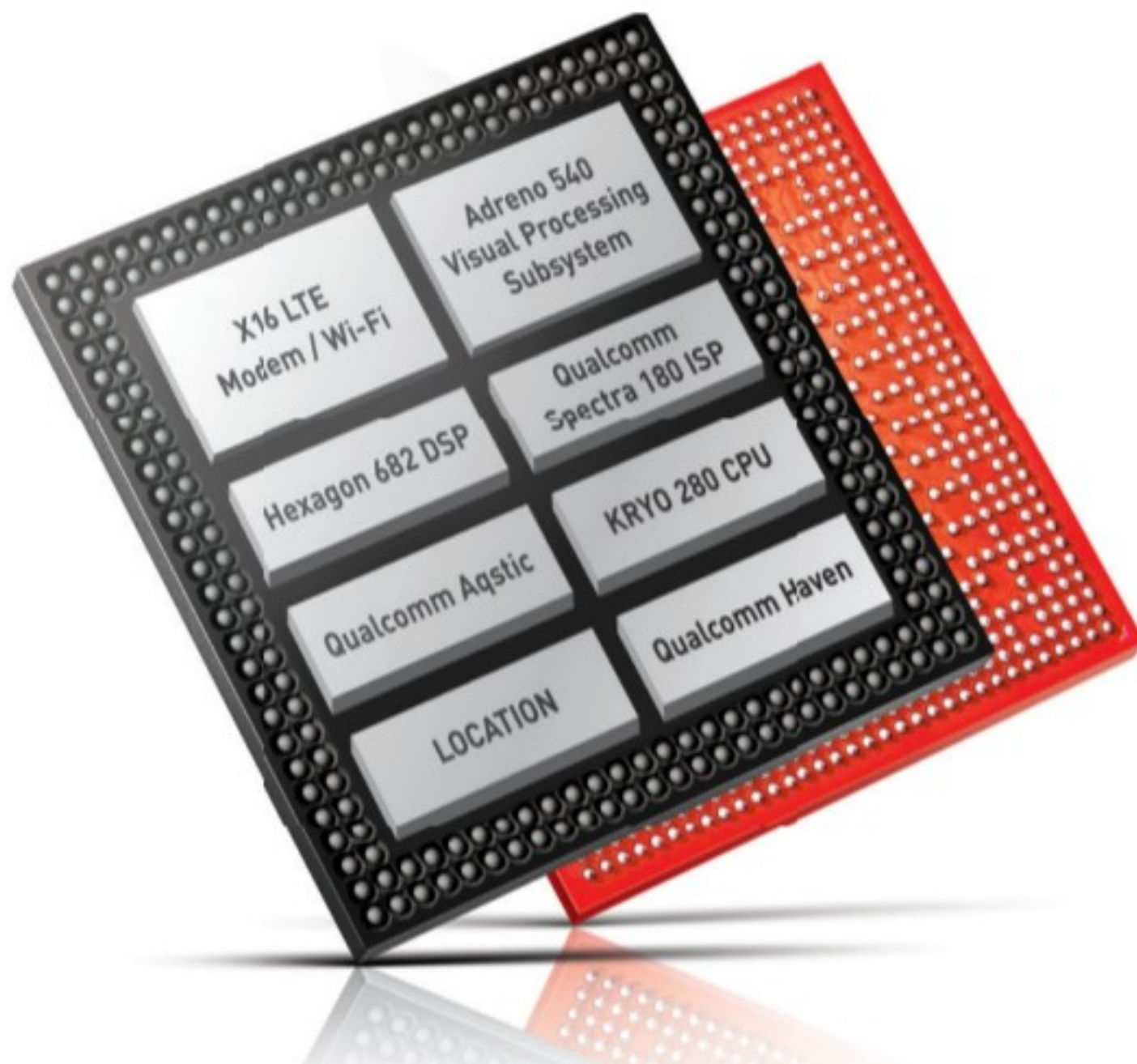
- Integrates System for Tactical Archival, Retrieval, and Exploitation (STARE) server for comprehensive data management
- Perform data queries within federated STARE database
- Access SIGACT and other external intelligence through DCGS network

EXPLOITATION AND DISSEMINATION

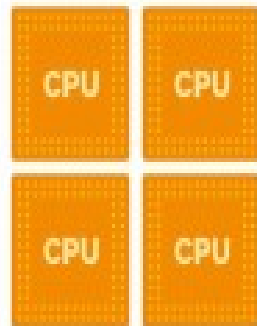


- Create intelligence products with advanced video and exploitation features
- View, cross-reference, and annotate multiple file types for comprehensive analysis
- Disseminate imagery directly from CLAW as GeoPDFs, Powerpoint, GeoTIFFs, and others





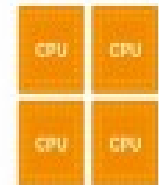
Kryo 280 Efficiency Cluster Optimization



Performance

Up to 2.45GHz
2MB L2

20% performance uplift over
range of use cases such as app load
time, web browsing, VR

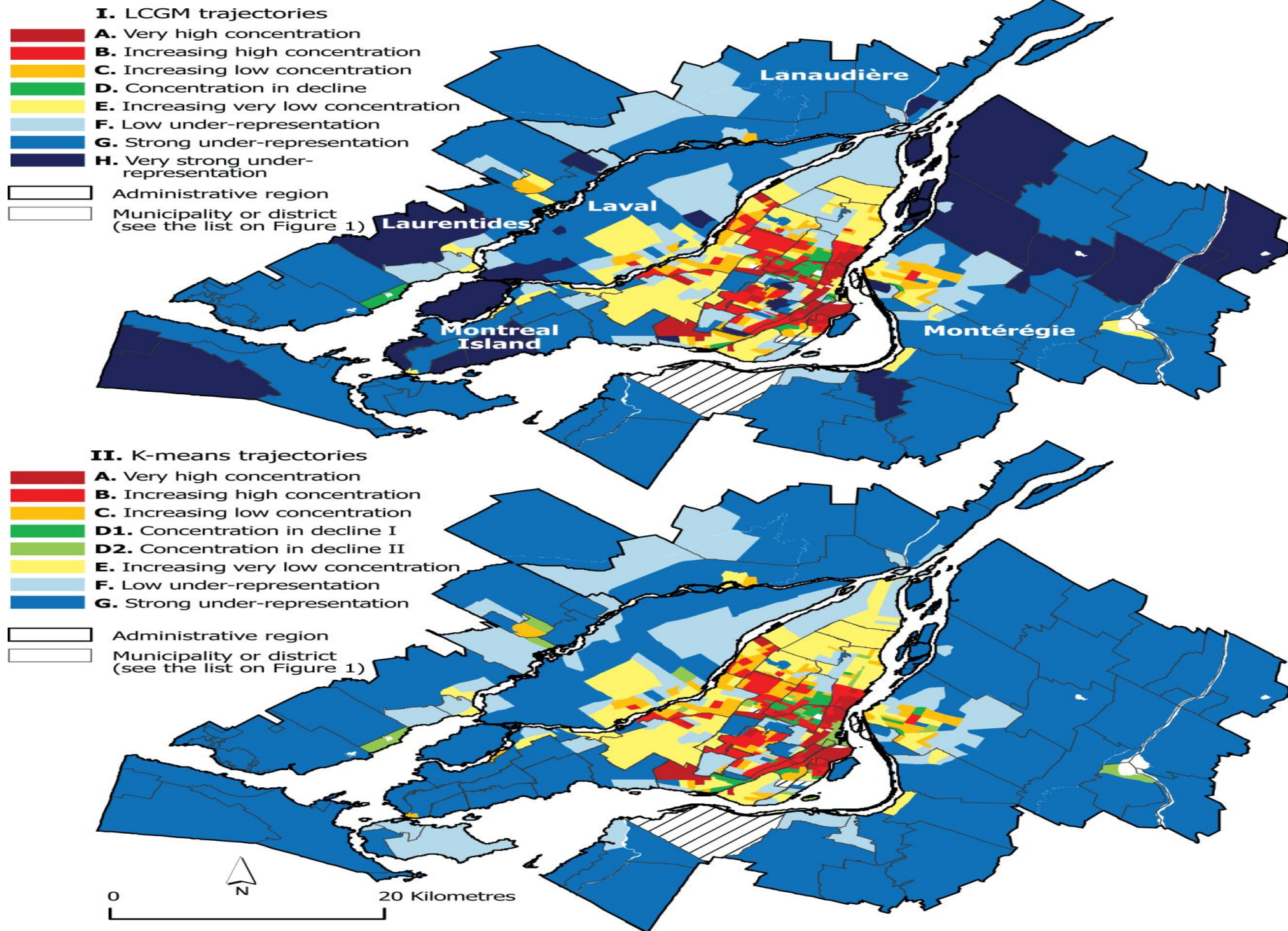


Efficiency

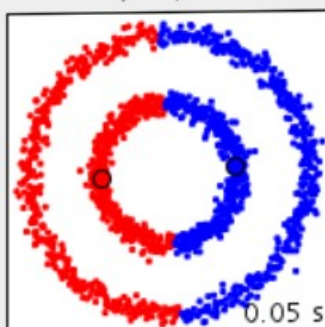
1.9GHz
1MB L2

80% of time is spent
on efficiency cluster

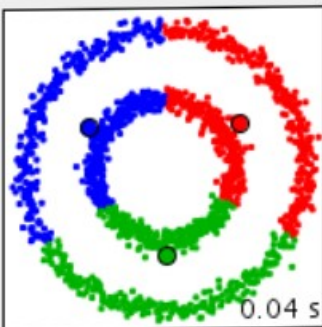
Minimized memory transaction
power with larger L2 cache



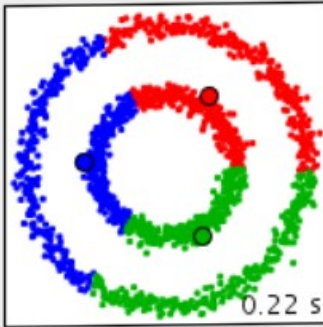
KMeans
($k=2$)



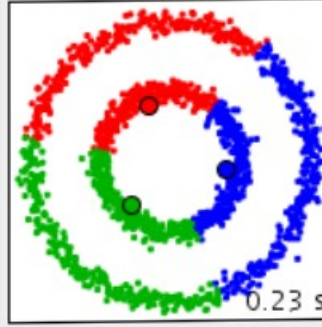
KMeans
($k=3$)



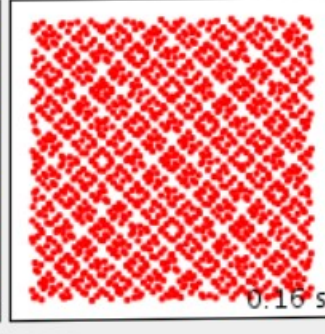
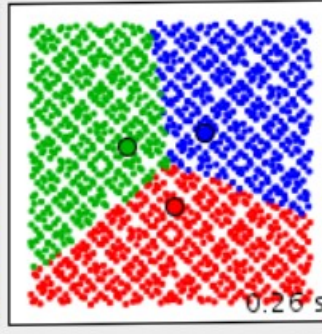
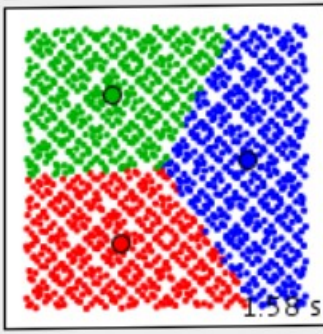
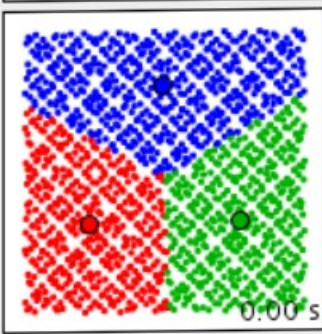
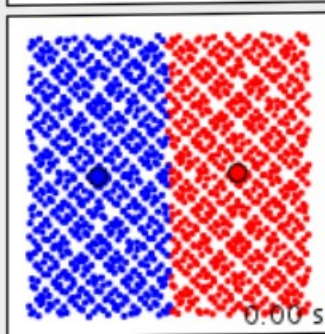
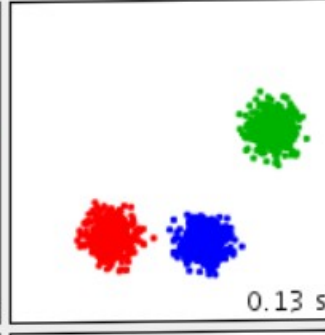
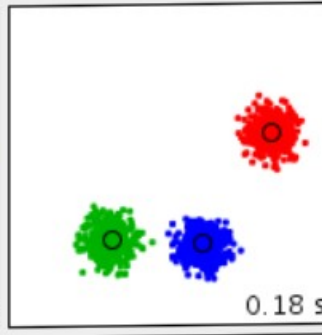
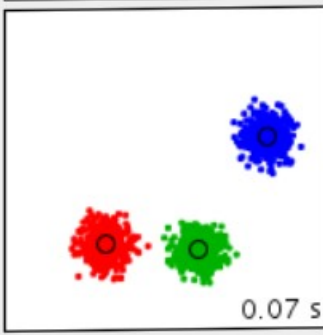
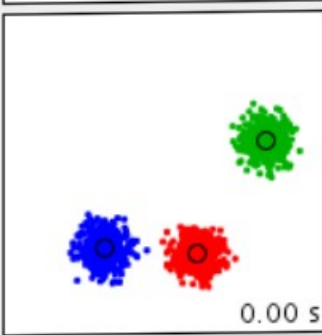
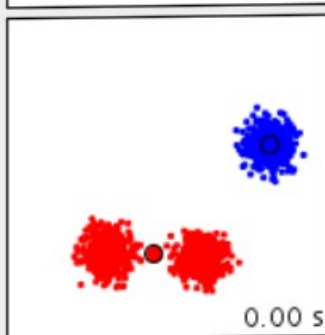
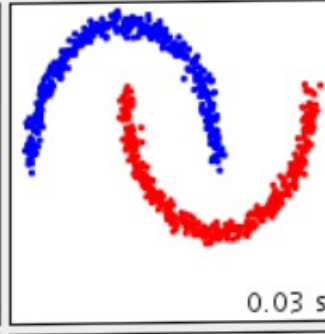
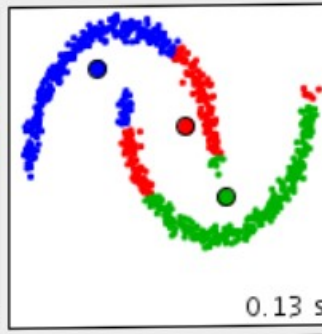
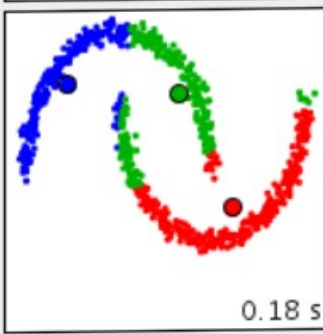
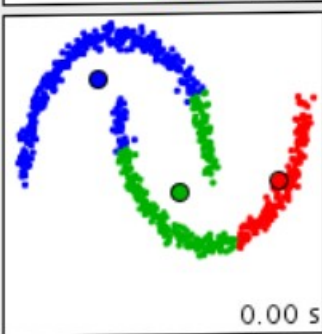
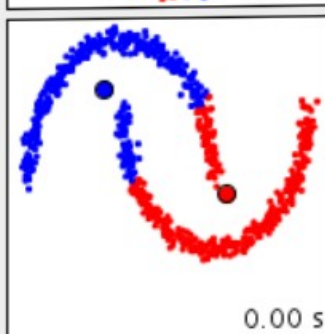
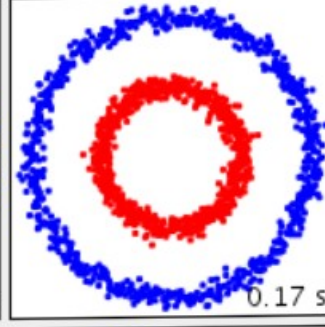
FuzzyKMeans
($k=3$, fuzzy=2)



FuzzyKMeans
($k=3$, fuzzy=10)



DBSCAN
($\text{eps}=.1$, $\text{min}=3$)



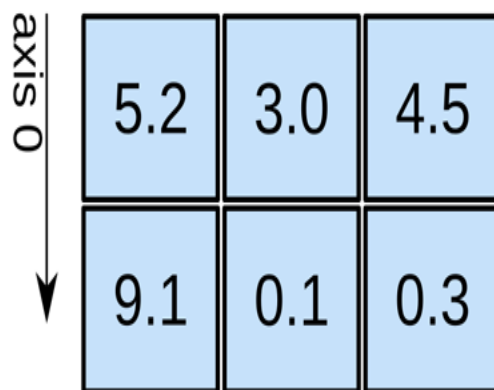
1D array



axis 0 →

shape: (4,)

2D array

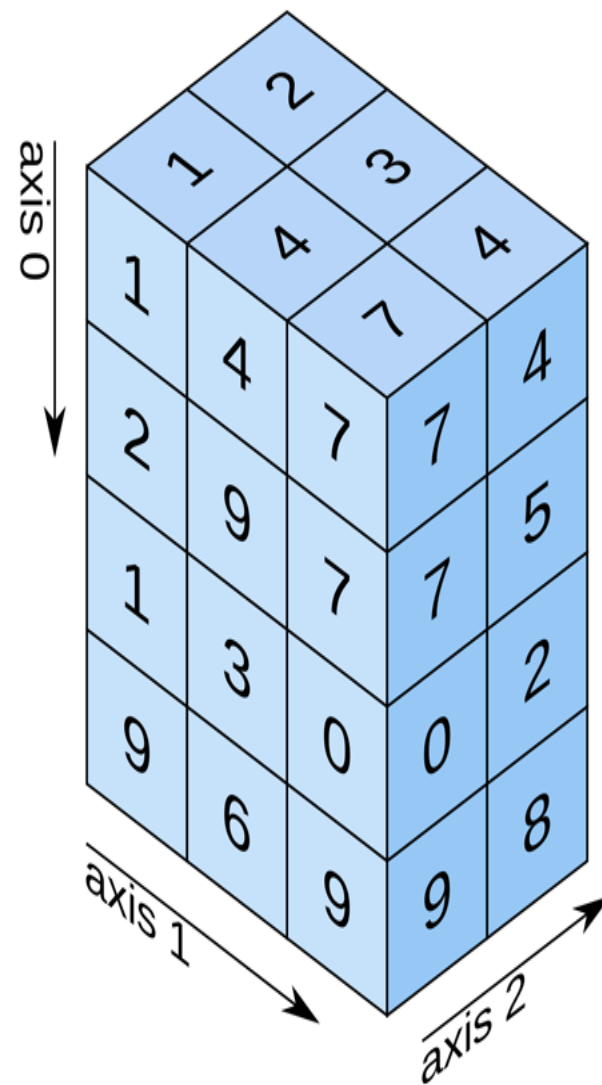


axis 0 ↓

axis 1 →

shape: (2, 3)

3D array



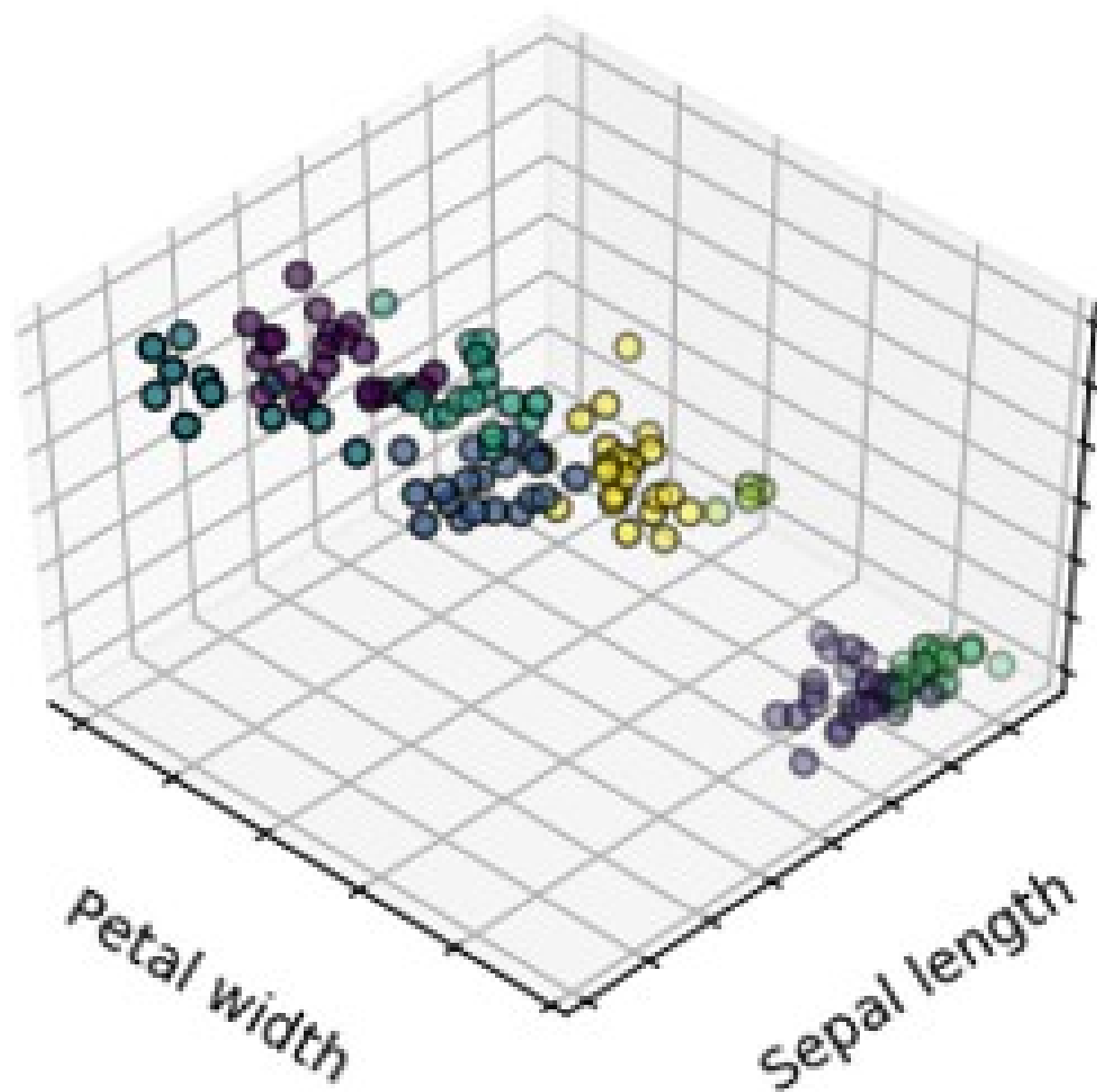
axis 0 ↓

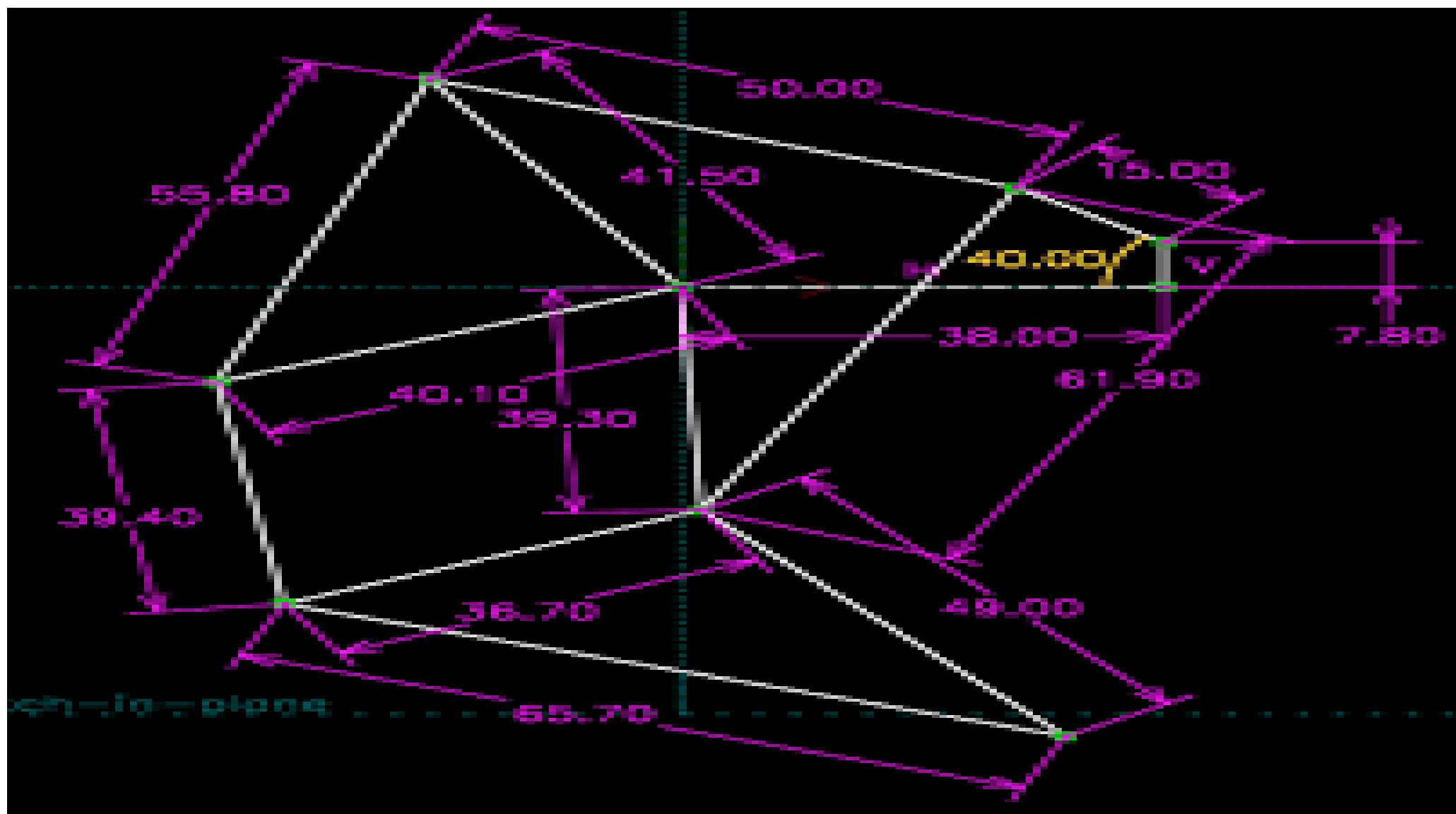
axis 1 ↘

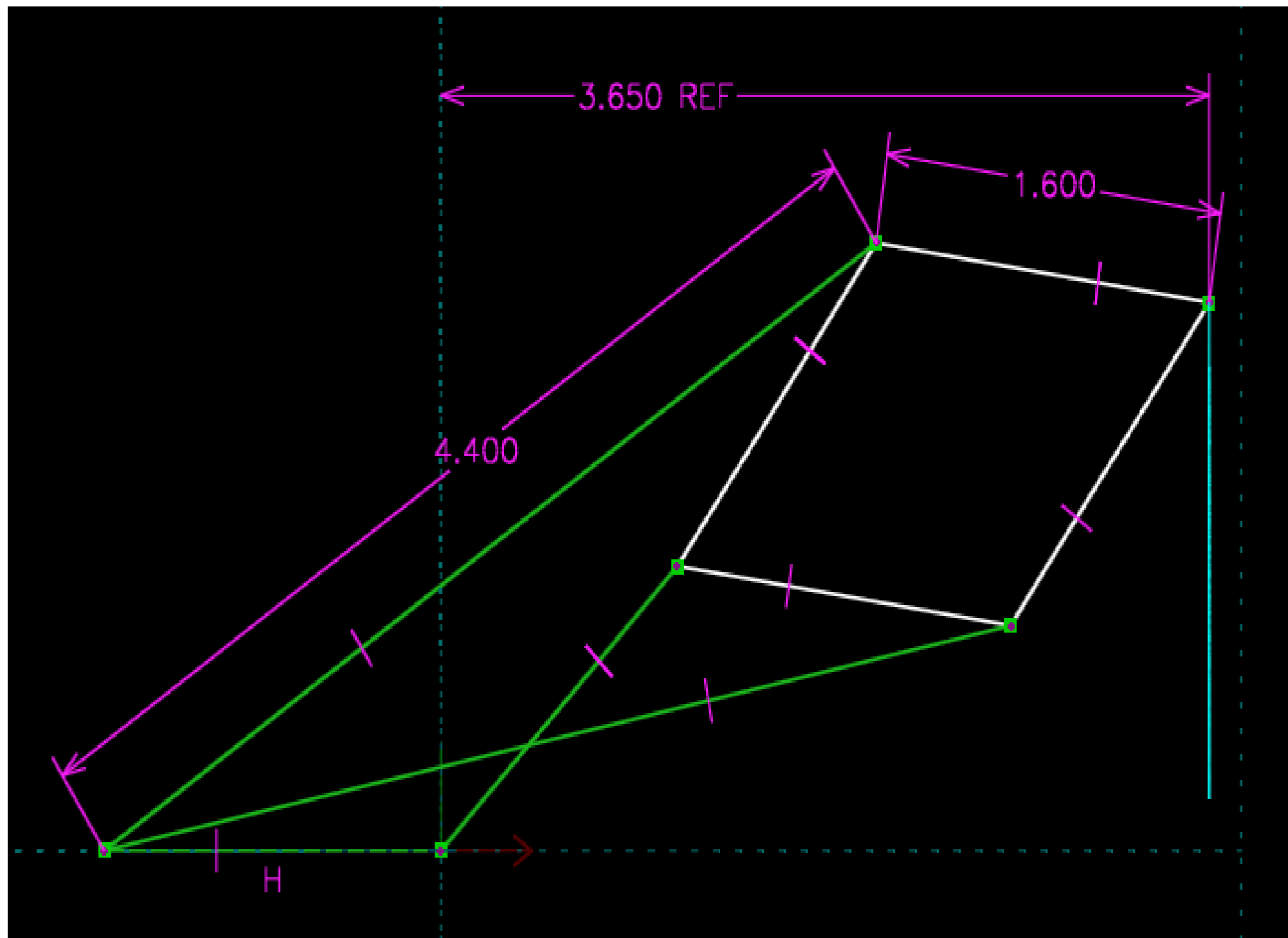
axis 2 ↗

shape: (4, 3, 2)

8 clusters







#Step 4: Taking mean and repeat

```
c_old=deepcopy(centroid)
```

```
for i in range(len(belongs_to)):
```

```
    if belongs_to[i]==0:
```

```
        mean[0][0]=np.mean(x[i][0])
```

```
        mean[0][1]=np.mean(x[i][1])
```

```
    else:
```

```
        continue
```

```
print("New Centroid for cluster 1:",mean[0])
```

```

1  import numpy as np
2
3  def cluster_points(X, mu):
4      clusters = {}
5      for x in X:
6          bestmukey = min([(i[0], np.linalg.norm(x-mu[i[0]])) \
7                          for i in enumerate(mu)], key=lambda t:t[1])[0]
8          try:
9              clusters[bestmukey].append(x)
10         except KeyError:
11             clusters[bestmukey] = [x]
12     return clusters
13
14 def reevaluate_centers(mu, clusters):
15     newmu = []
16     keys = sorted(clusters.keys())
17     for k in keys:
18         newmu.append(np.mean(clusters[k], axis = 0))
19     return newmu
20
21 def has_converged(mu, oldmu):
22     return (set([tuple(a) for a in mu]) == set([tuple(a) for a in oldmu]))
23
24 def find_centers(X, K):
25     # Initialize to K random centers
26     oldmu = random.sample(X, K)
27     mu = random.sample(X, K)
28     while not has_converged(mu, oldmu):
29         oldmu = mu
30         # Assign all points in X to clusters
31         clusters = cluster_points(X, mu)
32         # Reevaluate centers
33         mu = reevaluate_centers(oldmu, clusters)
34     return(mu, clusters)

```

