ACCELERATION PAR GPU

Morgan Fouque

Olivier STRAUSS, Michelle JOAB, Alexandre DUHANT, Christophe ARCHIER

Plan de la présentation

- L'entreprise et l'avant-stage
- Les missions proposées
- Premiers pas dans le GPU
- Métriques de qualité
- Résultats finaux

T-Waves Technologies

Analysis & Control inside the matter

Worldwide

Contrôle qualité

Contrôle packaging

Caractérisation

Tomographie

Imagerie

Métrologie

Spectroscopie

Réflectométrie

Aéronautique

Automobile

Aérospatial

Médical

Génie Civil

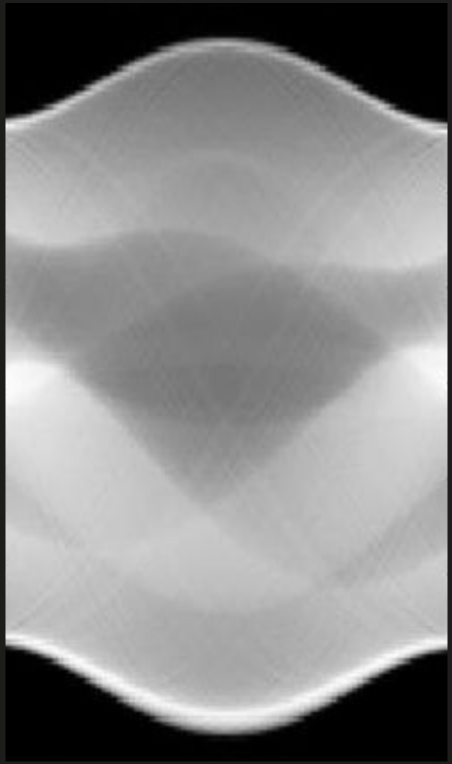
Défense / Armement

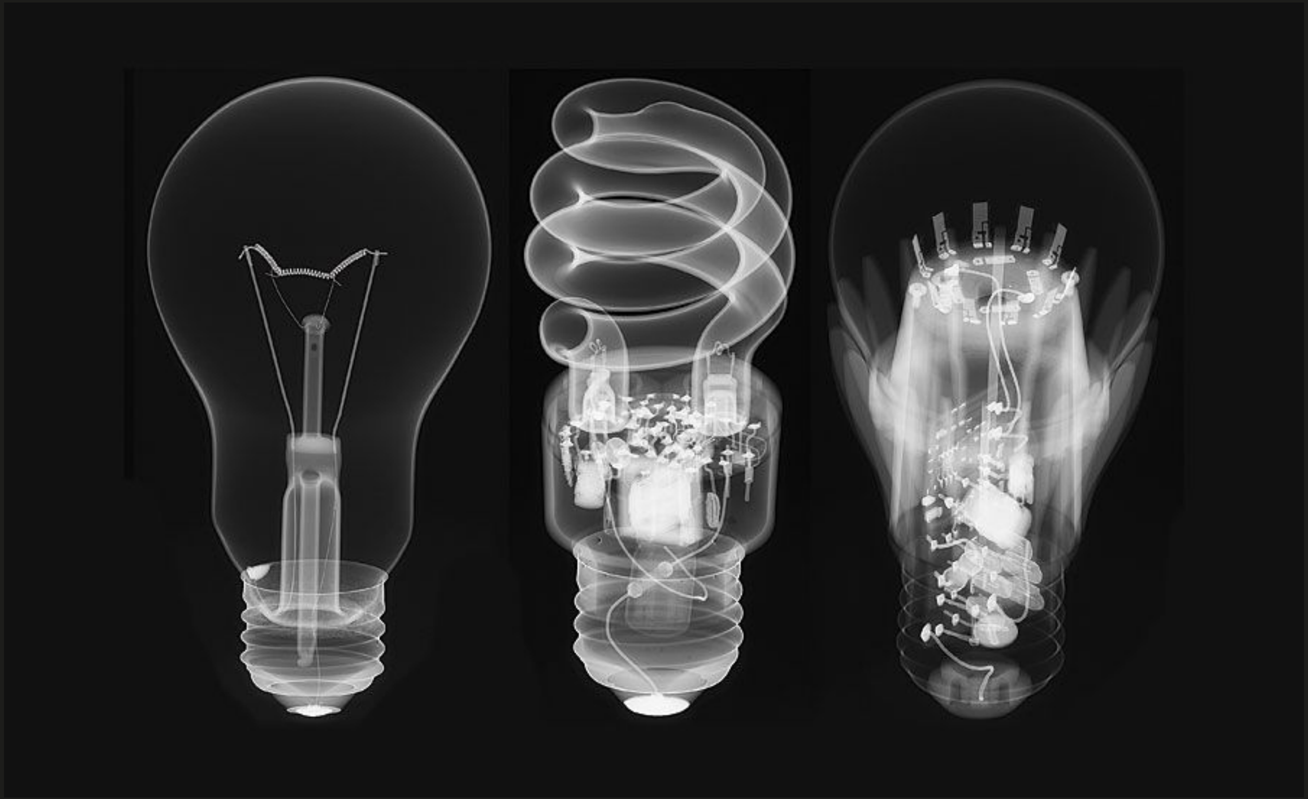
…

Etat des travaux: Tomographie

Maximum Likelihood Expectation Maximization

Simultaneous Iterative Reconstruction Technique



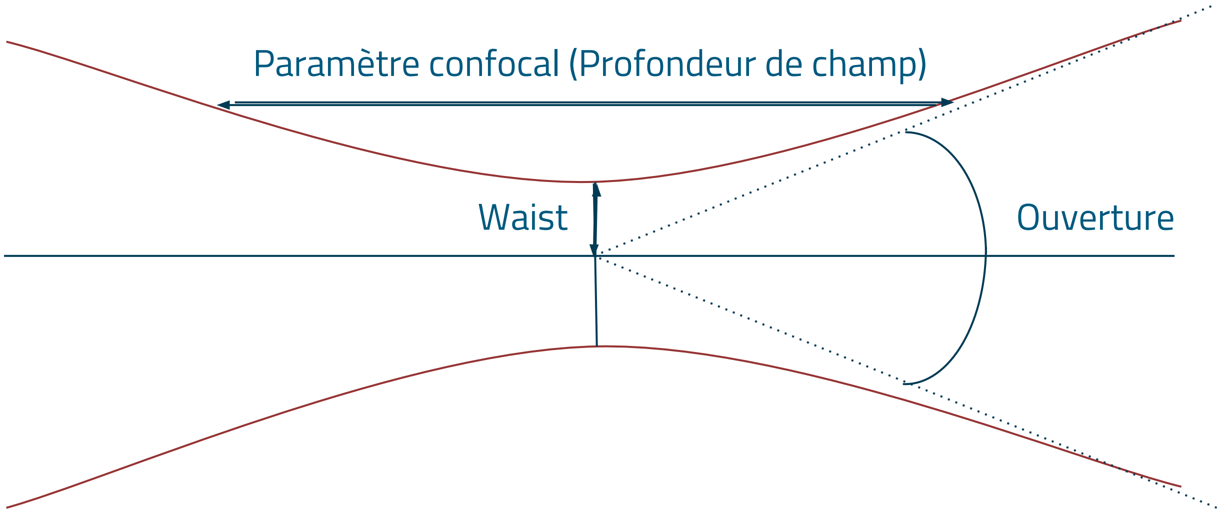


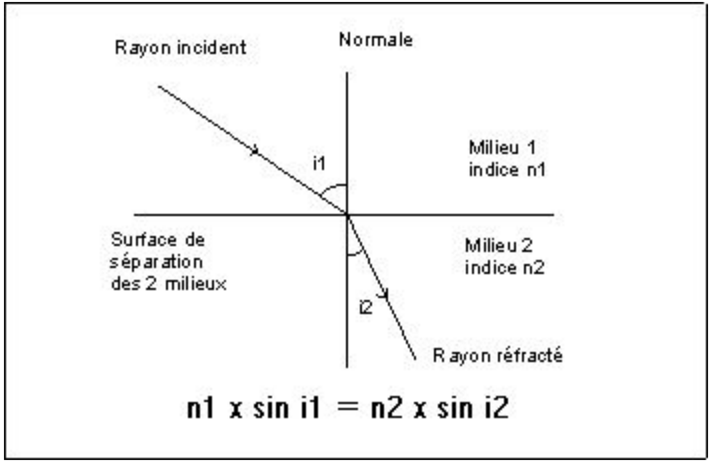
Etat des travaux: Simulation

Matrice de Radon

Raycasting

Matrice de Radon et Faisceau THz





Principe de réfraction

Raycasting et réfraction

Missions proposées

- Accélération GPU
- Généralisation 3D
- Visualisation

Formation et Profil

- 3D technique
- Traitement de l'image
- Traitement du signal

- Profil technologique
- Curiosité pour l' électronique
- Résolution de problèmes

Premiers pas dans le GPU

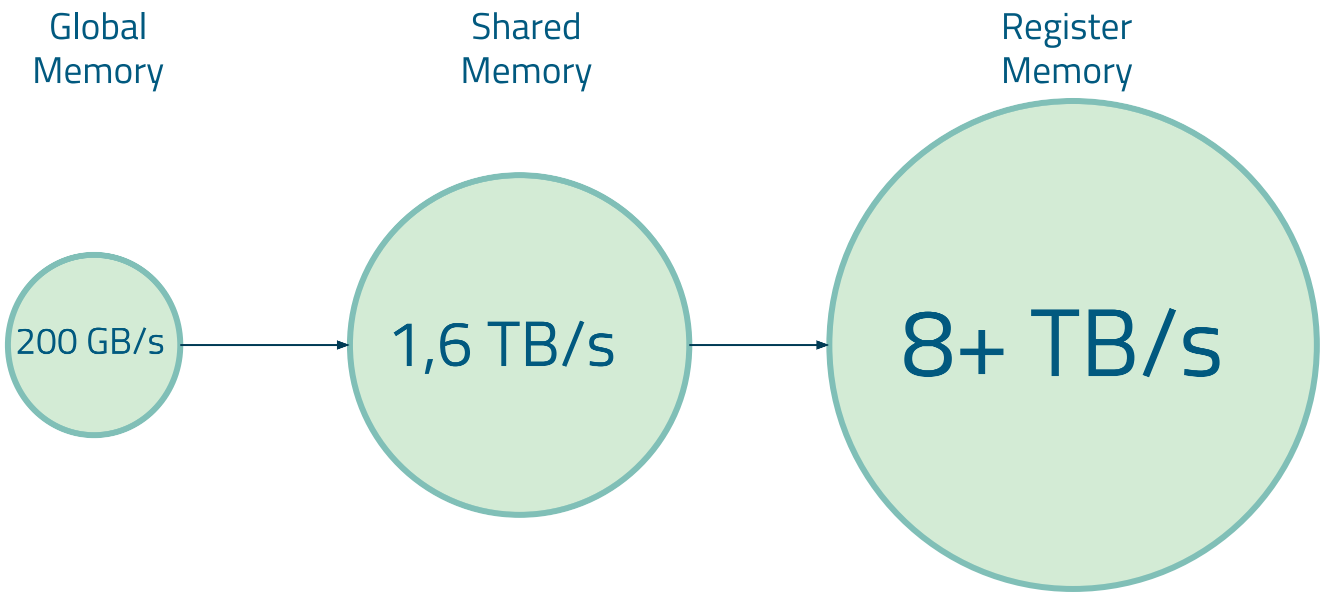
Man vs Wild

Performance optimization revolves around three basic strategies:

- Maximizing parallel execution
- Optimizing memory usage to achieve maximum memory bandwidth
- Optimizing instruction usage to achieve maximum instruction throughput

Cuda Best Practice Guide

Utilisation de la mémoire registre



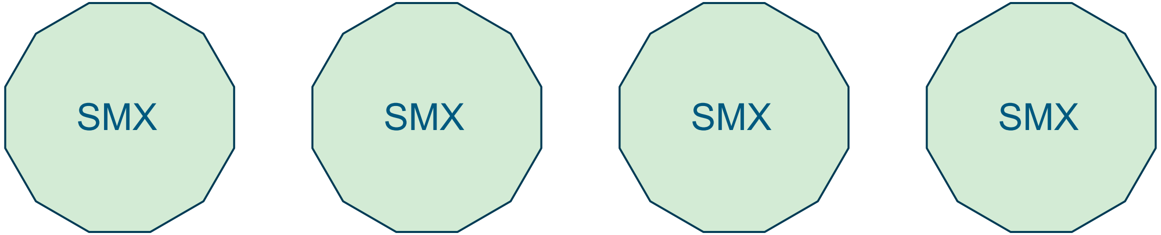
Architecture Fermi

Métriques de qualité GPU

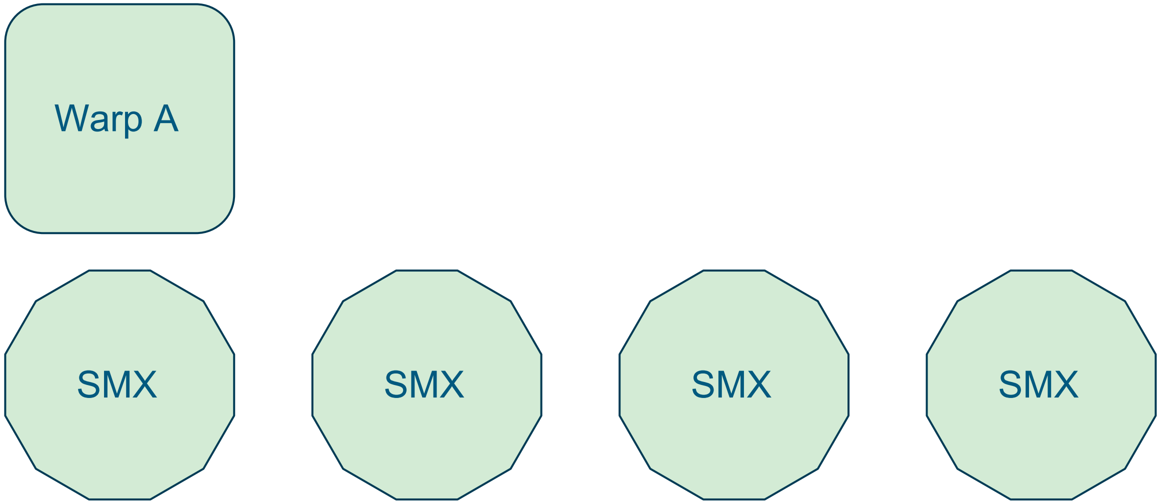
Métriques de qualité

- Occupancy
- Instruction-Level Parallelism
- Thread-Level Parallelism

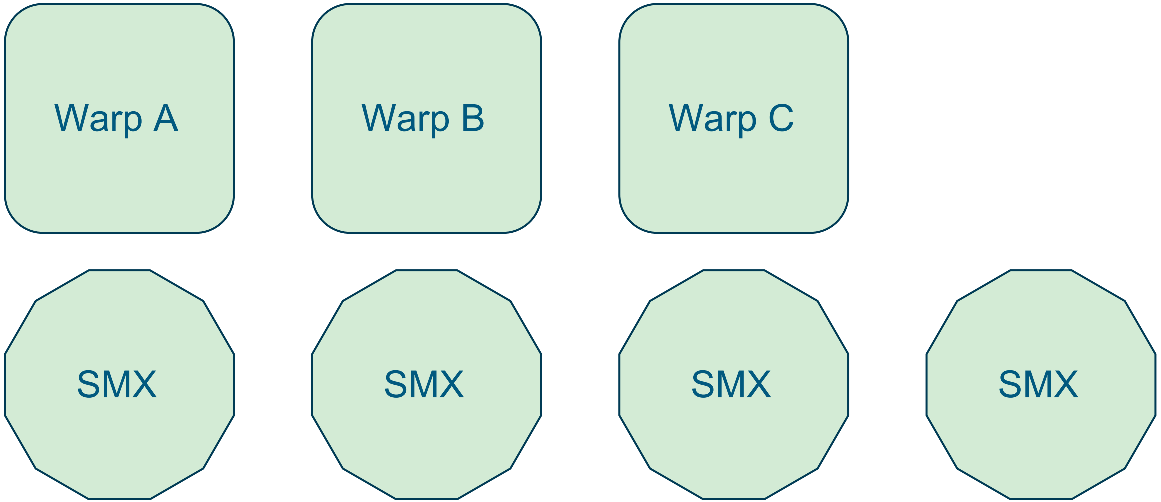
Occupancy



Occupancy : 0%



Occupancy : 25%



Occupancy : 75%

Warp A

Registres: 32/threads
Max: 64 warps/blocs
100% occupancy
33% bandwidth

Warp B

Registres: 64/threads
Max: 32 warps/blocs
50% occupancy
100% bandwidth

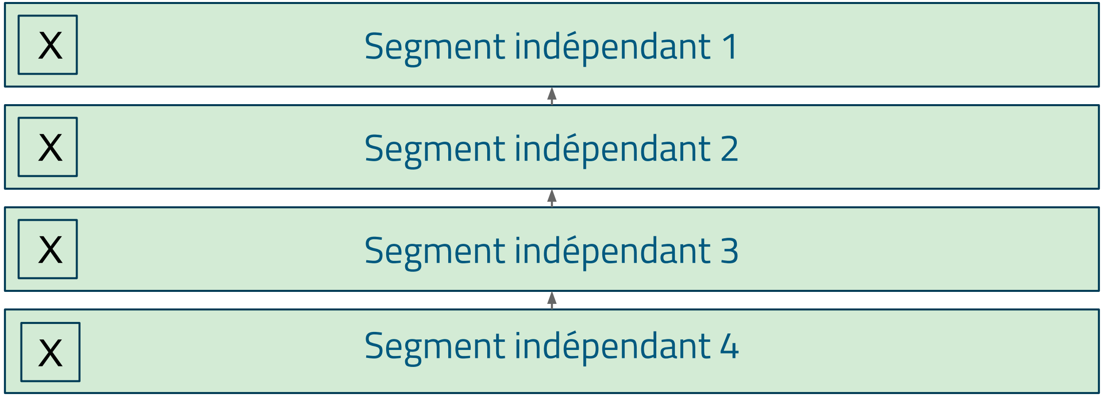
Limites de l’occupancy

Instruction-Level Parallelism

```
A = 0;
A = A + 5;          // A devient indisponible
C = A;              // Latence arithmétique

B = memory[0];      // B devient indisponible
B = B + 10;         // Latence mémoire
```

Latences mémoires et arithmétiques



4 segments de X instructions indépendantes,
X étant la densité arithmétique

Instruction-Level Parallelism

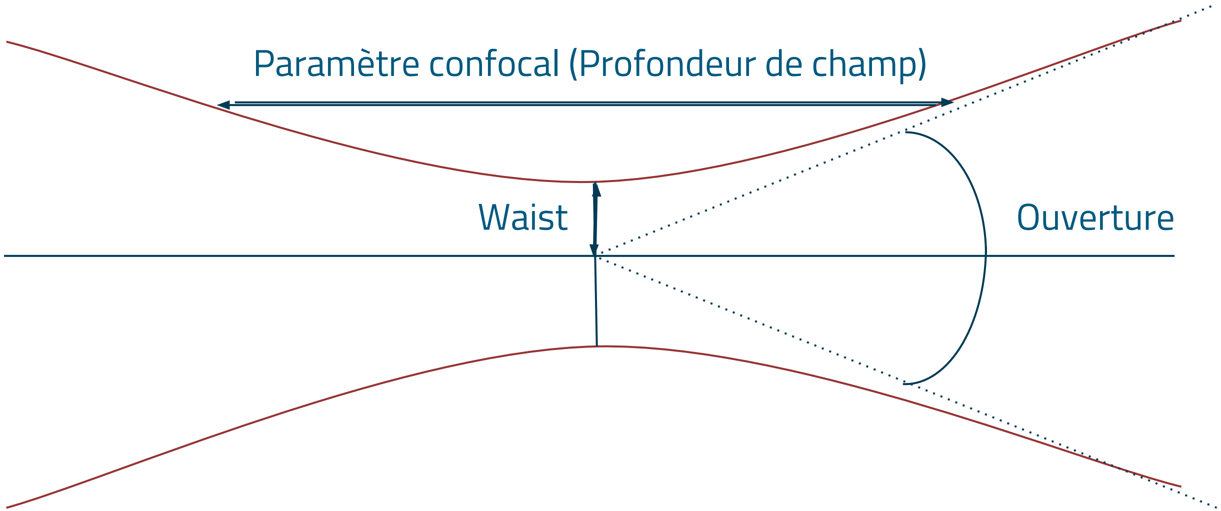
Thread-Level Parallelism

Thread-Level Parallelism

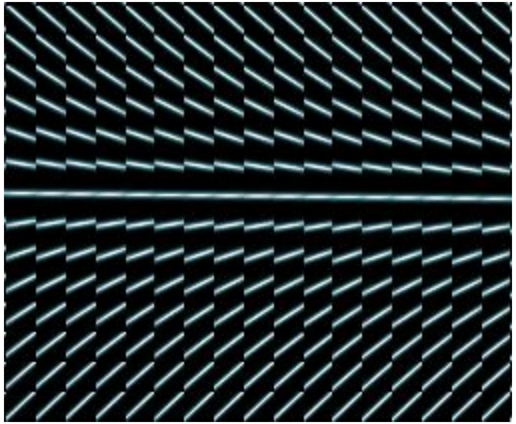
Missions principales

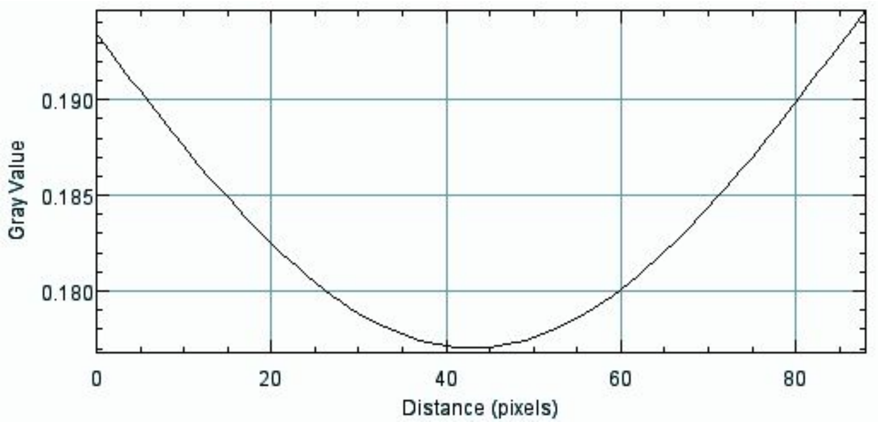
Accélération GPU

Cas simple: Matrice de Radon



Cas simple: Matrice de Radon





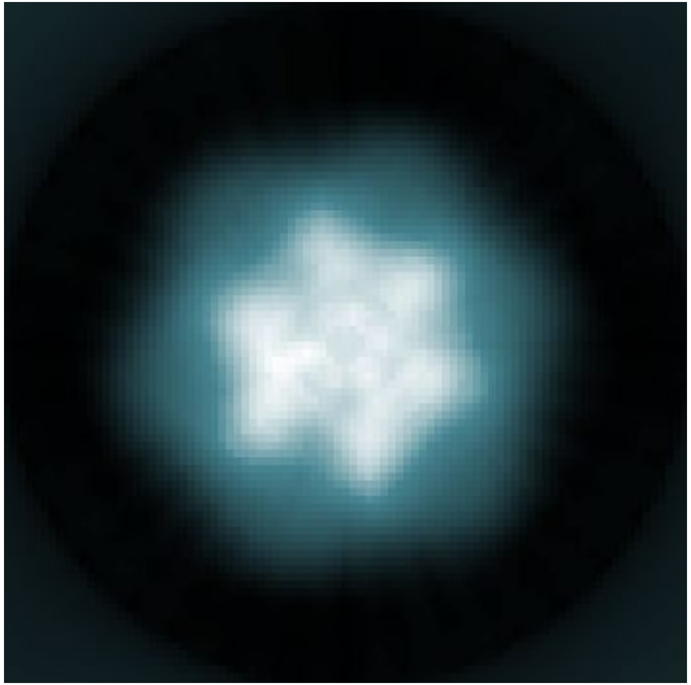
Cas simple: Matrice de Radon

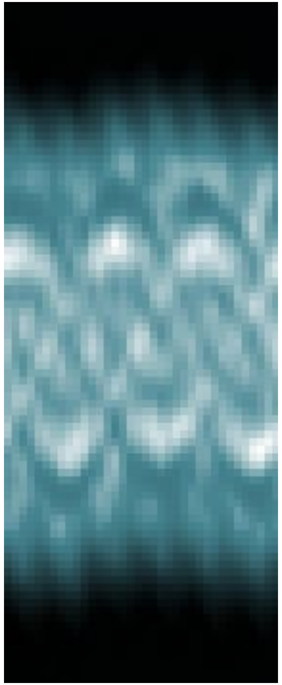
- Excellente densité arithmétique
- Pas de dépendance mémoire
- Pas d'effet de queue
- Occupancy > 95%

100x

De 11.2 secondes à 0.1 seconde

Cas complexe: Reconstruction





Cas complexe: Reconstruction

IMAGE

MATRICE DE RADON

SINOGRAMME

Cas complexe: Reconstruction

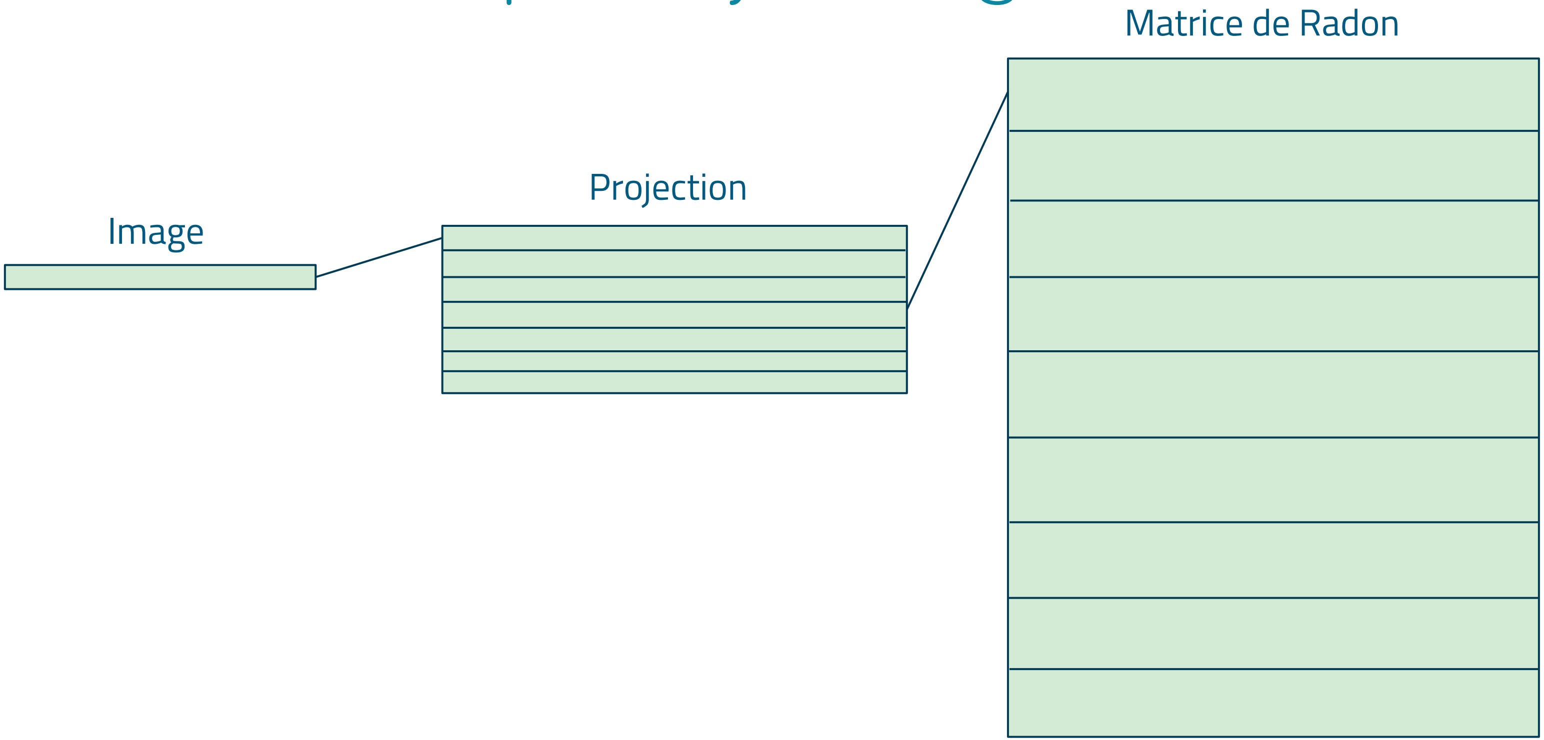
- Faible densité arithmétique
- Énorme dépendance mémoire/arithmétique
- Pas d’effet de queue
- Occupancy > 90%

10x

De 12 minutes à >1 minute.

Simulation par Raycasting discret

Simulation par Raycasting discret



Simulation par Raycasting discret

- Densité arithmétique faible
- Dépendance mémoire présente
- Effet de queue important
- Occupancy proche de 50%

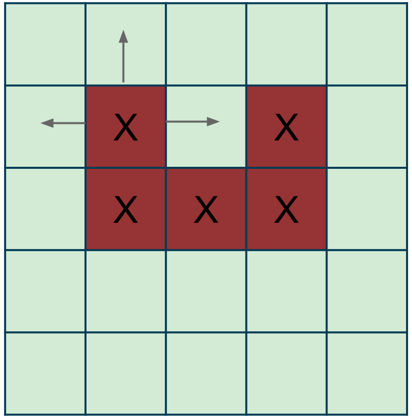
30x

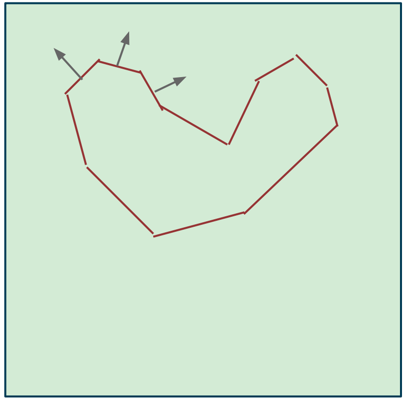
De 90 secondes à 3 secondes.

Missions principales

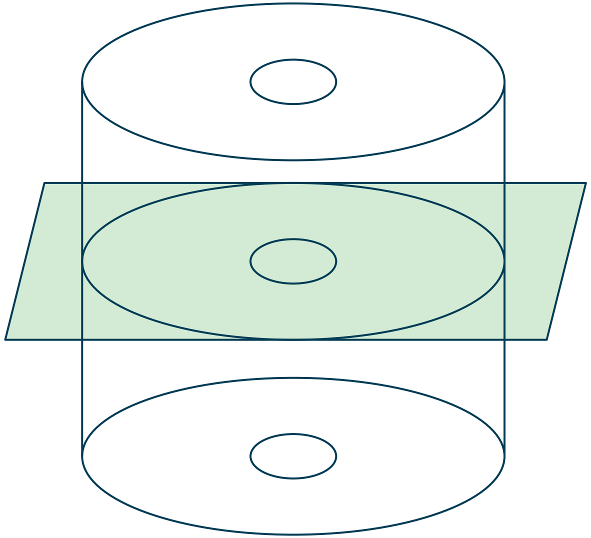
Généralisation à la 3D continue

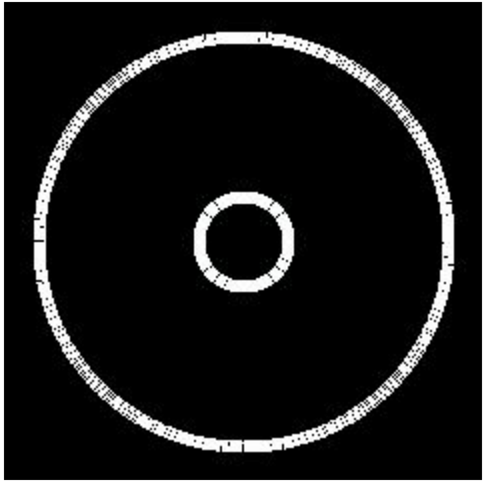
Discret / Continu



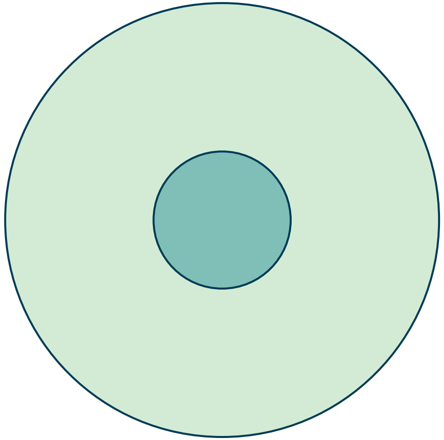


Colliers collisionnels



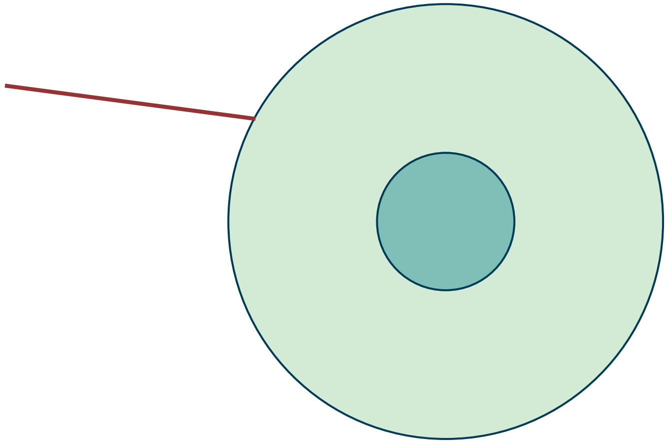


Gestion des matériaux



Gestion des matériaux

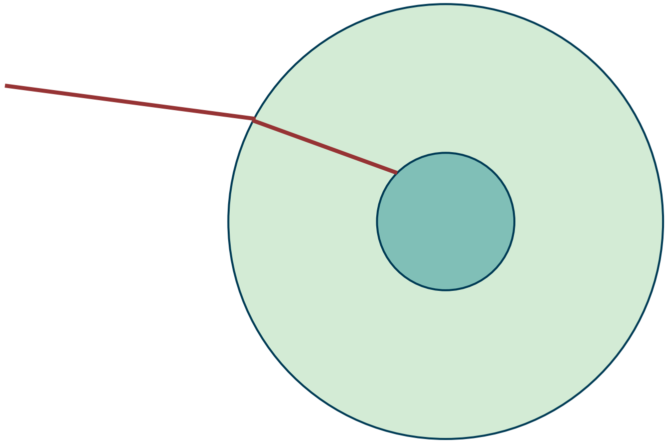
1



Gestion des matériaux

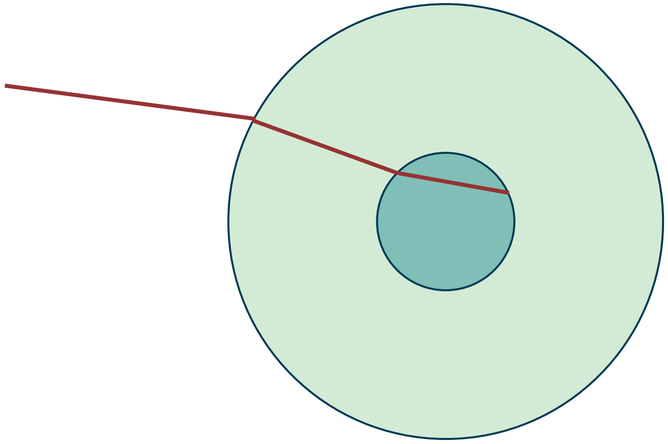
1

2

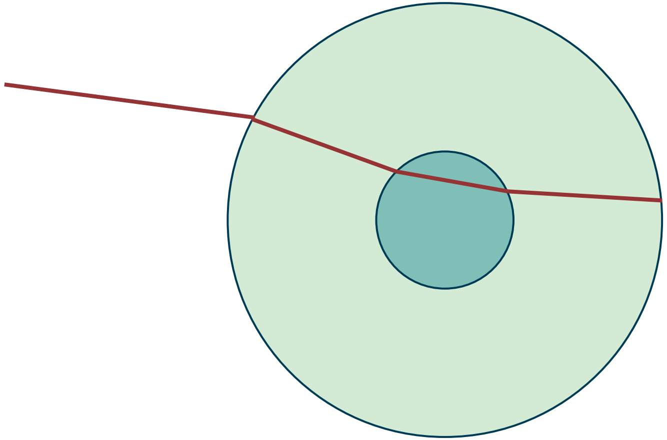


Gestion des matériaux

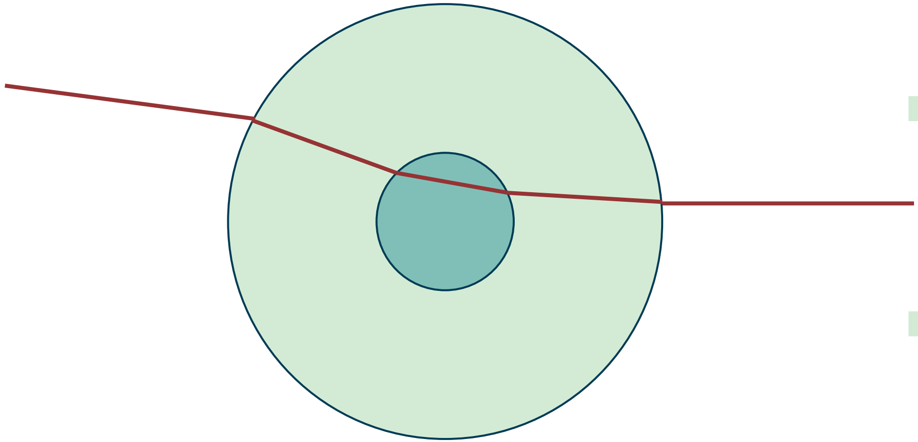
1

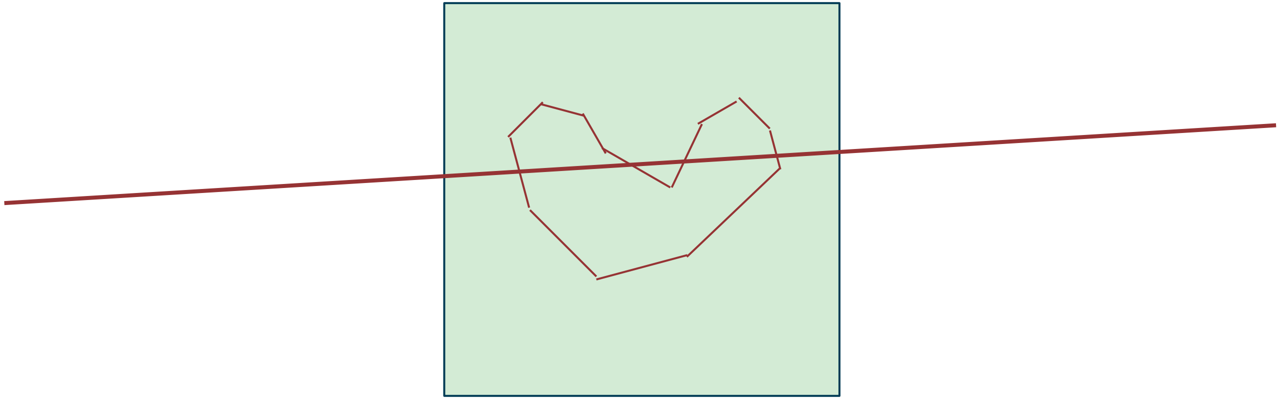


Gestion des matériaux



Gestion des matériaux





Collision segment à segment

Simulation par Raycasting continu

- Densité arithmétique faible
- Dépendance mémoire présente
- Effet de queue important
- Très gros problème de branchements
- Occupancy proche de 10%

6x

De 3 secondes à >0.5 seconde.

180x

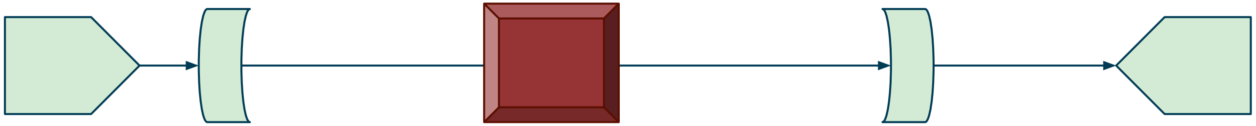
Comparé à l’état initial de l’algorithme

Missions secondaires

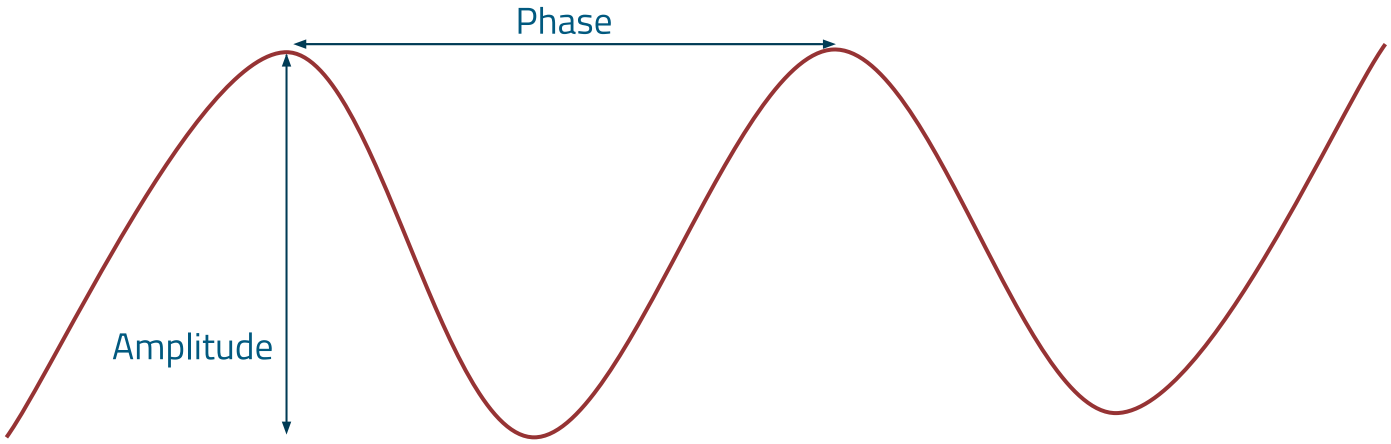
Mission proposées

- Accélération GPU
- Généralisation 3D
- Banc d'analyse vectoriel
- Visualisation 3D
- Communication

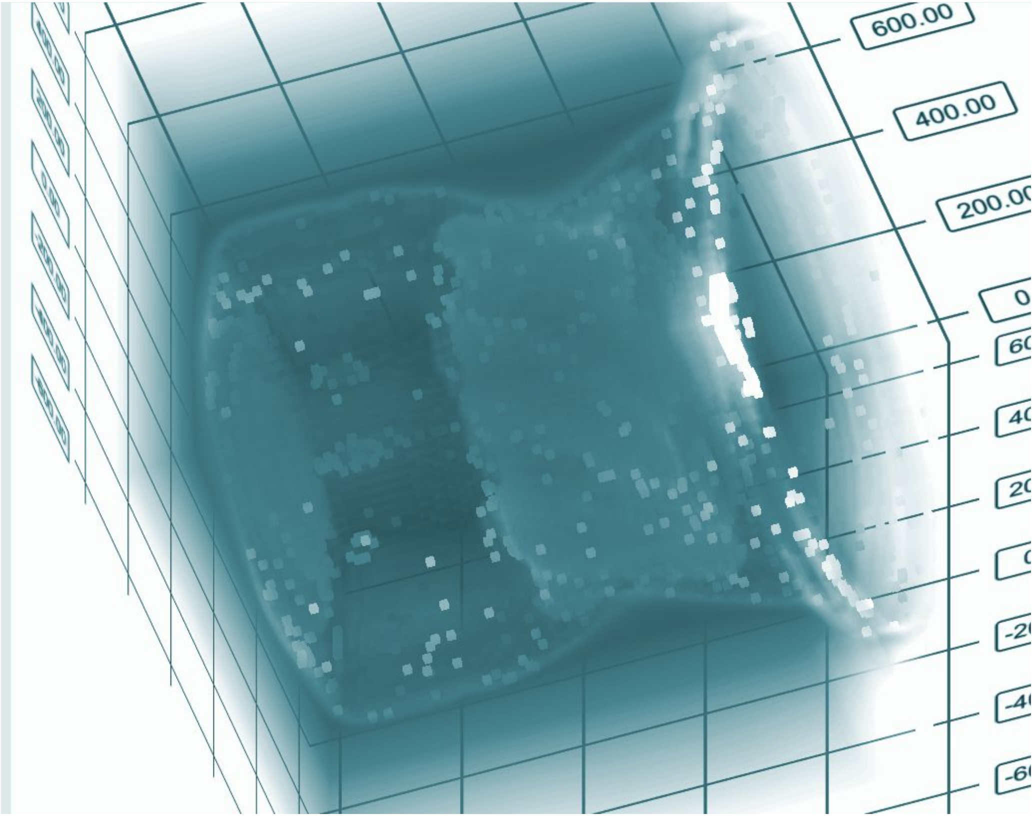
Banc d'analyse vectoriel



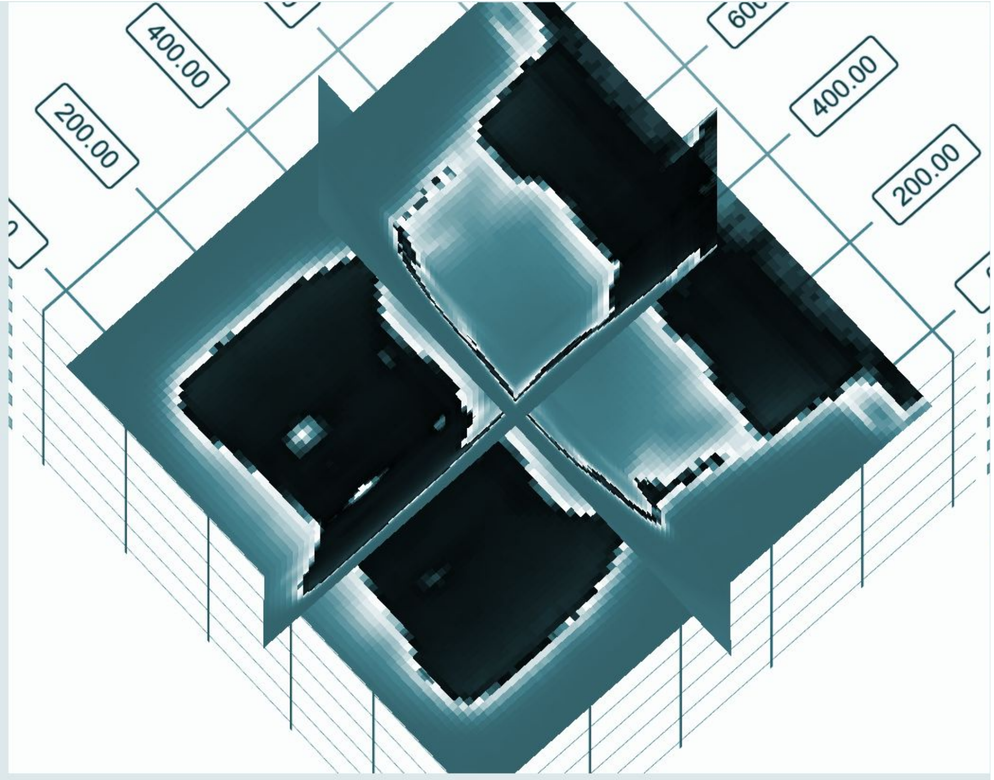
Banc d'analyse vectoriel



Visualisation 3D



Visualisation 3D



Communication

```matlab
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```

```cpp
float * buffer;
try
{
    buffer = new float[mdata.Largeur*mdata.Largeur*
}
catch (std::bad_alloc& ba)
{
    std::cerr << "bad_alloc caught: " << ba.what()
}
buffer[mdata.Largeur*mdata.Largeur*mdata.Largeur*md
```

```scilab
//parameters of figure
//build figure
img = imwrite(A, 'figure.bmp');
B = imread('figure.bmp');
C = edge(B, 'canny'); //finding edge
[xi,yi] = find(C);
rx = x(xi);
ry = y(yi);
angle = atan(ry,rx)*180/%pi; //getting polar angle
D = cat(1,angle,rx,ry); //concatenate these values
[E,k] = gsort(D(1,:),'g','i');
last = D(:,k);
Isum = 0;
//calculating for area of the figure
for i = 1:length(last(1,:)),
    if i == length(last(1,:)) then
        Ipie = 0.5*(last(2,i)*last(3,1)-last(2,1)*last(3,i));
    else i!=length(last(1,:));
    Ipie = 0.5*(last(2,i)*last(3,i+1)-last(2,i+1)*last(3,i));
Isum = Isum + Ipie;
    end,
end
```

Conclusion

Merci! ☻

Des questions?