Interpolation surfacique

Santé et prévention

Baptiste Jacquin
41107

Comment repérer une victime de nuit ?

Schéma de notre solution

Caméra de faible résolution $\longrightarrow$ Traitement du signal

- ✓ Taille et masse
- ✓ Débit de données
- ✓ Sobriété énergitique
- ✓ Coût
- ✗ Résolution

Augmente la résolution pour permettre un pilotage fluide

Problématiques

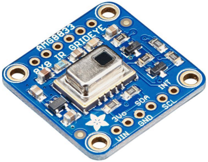
Peut-on réaliser un système adapté à notre usage ?

Comment traiter le signal de sortie de notre système ?

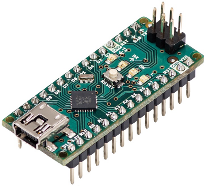
Quantification des contraintes

Contrainte	**Maximum**
Masse	250 g
Volume	$8 \times 10^{-3}\,\mathrm{m}^3$
Tension d'alimentation	10 V

Choix des composants



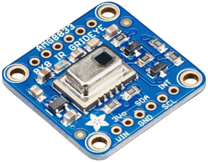
Caméra

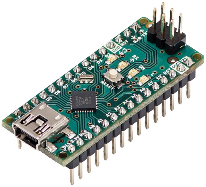


Arduino

Python

Interfaces entre les composants





Caméra $\longleftrightarrow$ (Soudage) Arduino $\longleftrightarrow$ (`serial`) Python

Système final

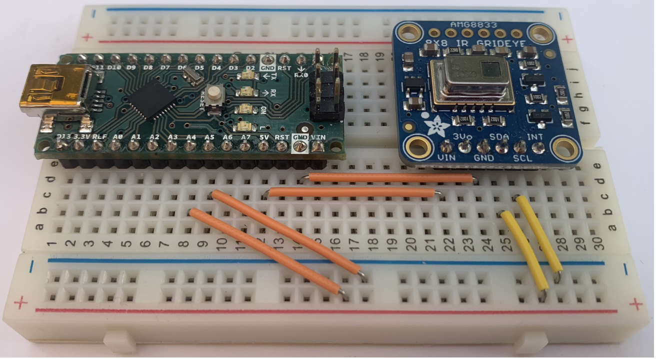
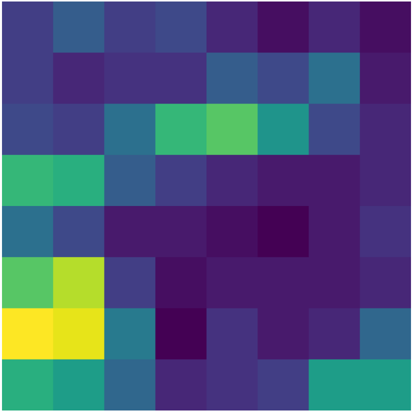
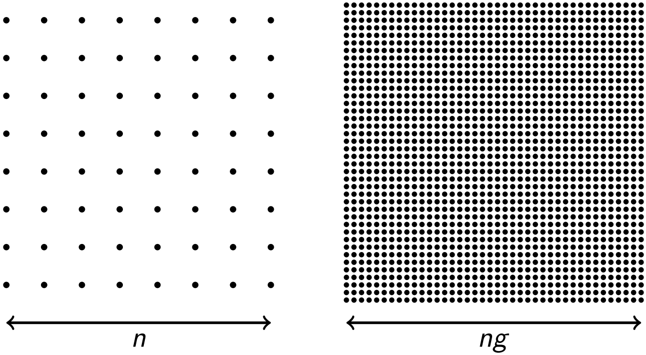


Image thermique produite



Interpolation surfacique

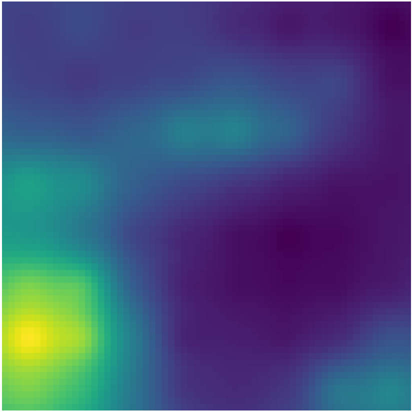


$$\mathcal{I}(\mathrm{M}) = \sum_{\mathrm{P} \in \mathcal{D}} \phi_{\mathrm{M}}(\mathrm{P}) \cdot \mathcal{I}(\mathrm{P})$$

Interpolation par inverse des distances

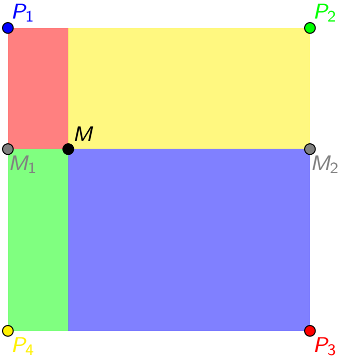
$$\phi_{\mathrm{M}}(\mathrm{P}) \propto \mathrm{distance}(\mathrm{M},\mathrm{P})^{-1}$$

$$\Theta(n^4 g^2)$$



Inverse des distances

Interpolation bilinéaire

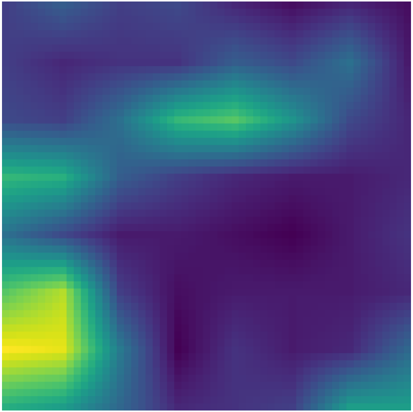


Succession de deux interpolations linéaires

$$\mathcal{I}(\mathrm{M}) = \sum_{i=1}^{4} \frac{\mathcal{A}_{\mathrm{M}}(P_i)}{\mathcal{A}} \cdot \mathcal{I}(P_i)$$

$$\Theta(n^2 g^2)$$

Deux fonctions auxiliaires
`lineaire`
`transpose`



Bilinéaire

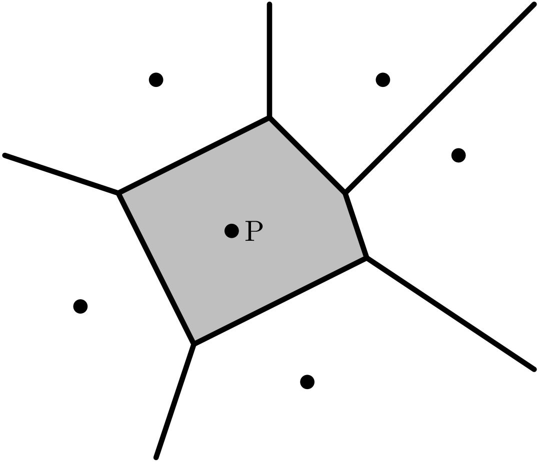
Interpolation par voisins naturels

Détermination d'un ensemble $\mathcal{V}$ de voisins naturels

$$\mathcal{I}(\mathrm{M}) = \sum_{\mathrm{P} \in \mathcal{V}} \phi_{\mathrm{M}}(\mathrm{P}) \cdot \mathcal{I}(\mathrm{P})$$

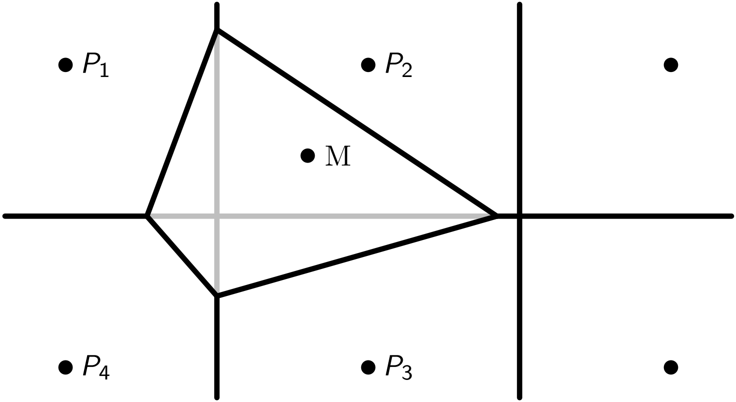
$$\Theta(n^2 g^2)$$

Diagramme de Voronoï



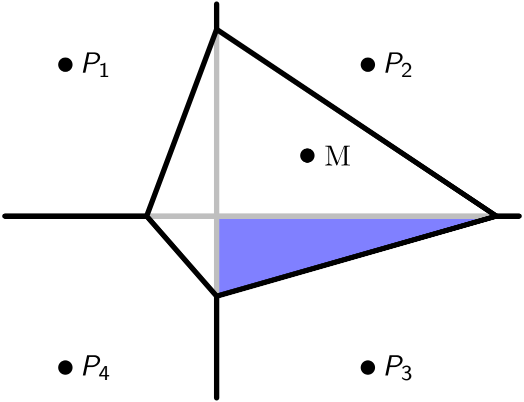
$$
\begin{aligned}
\mathrm{Vor}(\mathrm{P},\mathcal{D}) &= \{\mathrm{M}\in\mathbb{R}^2, \forall \mathrm{Q}\in\mathcal{D}, \quad \|\mathrm{M}-\mathrm{P}\| \leq \|\mathrm{M}-\mathrm{Q}\|\} \\
&= \bigcap_{\mathrm{Q}\in\mathcal{D}\setminus\{\mathrm{P}\}} \mathrm{D}(\mathrm{P},\mathrm{Q})
\end{aligned}
$$

Détermination des voisins $\mathcal{V}$



$$\mathcal{V}(\mathrm{M}) = \{\text{Points limitrophes de } \mathrm{Vor}(\mathrm{M}, \mathcal{D} \cup \{\mathrm{M}\})\}$$
$$\simeq \{P_1, P_2, P_3, P_4\}$$

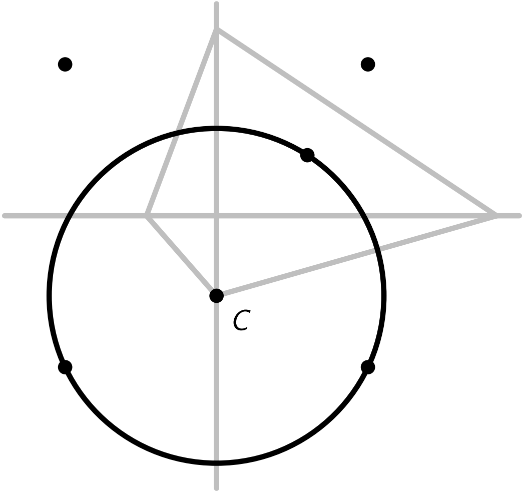
Pondération de Sibson et de Laplace



$$\phi_{\mathrm{M}}(\mathrm{P}_3) \propto \textcolor{blue}{\mathcal{A}}(\mathrm{Vor}(P_3, \mathcal{D}) \cap \mathrm{Vor}(\mathrm{M}, \mathcal{D} \cup \{\mathrm{M}\})) \qquad \text{(Sibson)}$$

$$\propto \frac{\text{Longueur de l'interface}}{\text{Distance entre les points}} \qquad \text{(Laplace)}$$

Stratégie d'implémentation



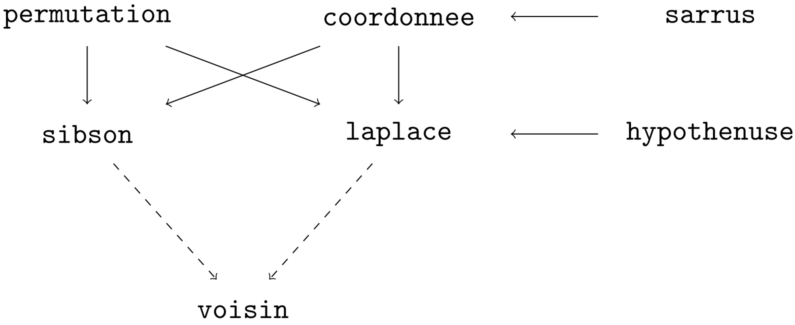
Voisins naturels

Pondération périodique

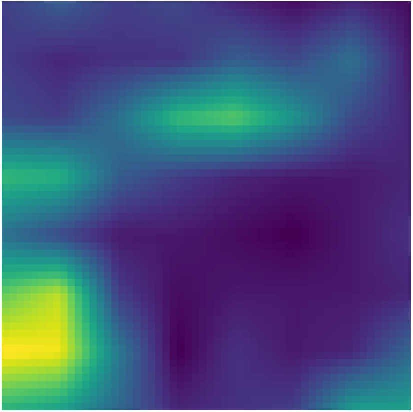
Symétries axiales

Centre du cercle circonscrit

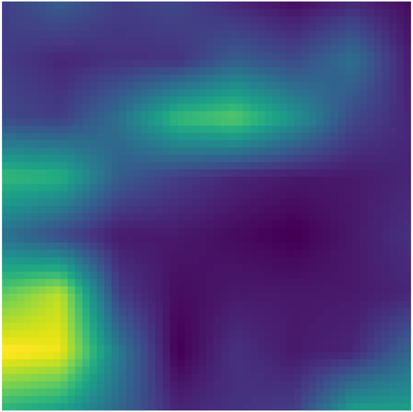
Organigramme de l'implémentation



Résultats



Pondération de Sibson



Pondération de Laplace

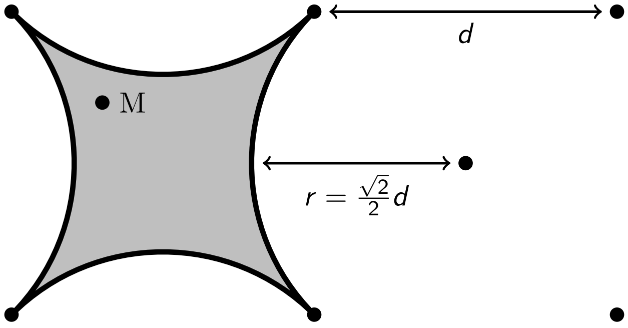
Conclusion

Prototype fonctionnel correspondant aux contraintes
Communication logicielle établie

Trois interpolations différentes implémentées
Analyse de complexité et du rendu

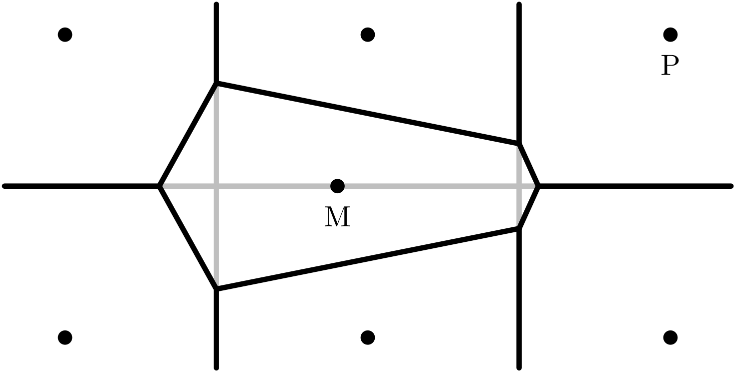
Affichage avec `PyQtGraph`
Interpolation temporelle

Annexe - Domaine de validité



$$
\begin{aligned}
\mathcal{A} &= 2 \cdot \mathcal{A}_{\text{carré}} - \mathcal{A}_{\text{disque}} \\
&= 2d^2 - \pi r^2 \\
&= (2 - \frac{\pi}{2}) \cdot \mathcal{A}_{\text{carré}}
\end{aligned}
$$

Annexe - Erreur d'approximation



$$\begin{aligned} \phi_{\mathrm{M}}(\mathrm{P}) &\approx 0.070 && \text{(Sibson)} \\ &\approx 0.025 && \text{(Laplace)} \\ &\ll 1 \end{aligned}$$

Annexe - Conditions de cocyclicité

$$M, A, B, C \text{ cocycliques}$$

$$\begin{aligned}
&\iff \exists (O, R), \forall P \in \{M, A, B, C\}, \quad (x_P - x_O)^2 + (y_P - y_O)^2 = R^2 \\
&\iff \exists (\alpha, \beta, \gamma), \forall P \in \{M, A, B, C\}, \quad x_P^2 + y_P^2 + \alpha x_P + \beta y_P + \gamma = 0
\end{aligned}$$

$$\iff \exists (\alpha, \beta, \gamma), \quad \underbrace{\begin{pmatrix} x_M^2 + y_M^2 & x_M & y_M & 1 \\ x_A^2 + y_A^2 & x_A & y_A & 1 \\ x_B^2 + y_B^2 & x_B & y_B & 1 \\ x_C^2 + y_C^2 & x_C & y_C & 1 \end{pmatrix}}_{N} \begin{pmatrix} 1 \\ \alpha \\ \beta \\ \gamma \end{pmatrix} = 0$$

$$\begin{aligned}
&\iff \det(N) = 0 \\
&\iff (x_M^2 + y_M^2)\Delta - \underbrace{\begin{vmatrix} x_A^2 + y_A^2 & y_A & 1 \\ x_B^2 + y_B^2 & y_B & 1 \\ x_C^2 + y_C^2 & y_C & 1 \end{vmatrix}}_{\alpha\Delta} x_M + \underbrace{\begin{vmatrix} x_A^2 + y_A^2 & x_A & 1 \\ x_B^2 + y_B^2 & x_B & 1 \\ x_C^2 + y_C^2 & x_C & 1 \end{vmatrix}}_{\beta\Delta} y_M - \tilde{\Delta} = 0
\end{aligned}$$

Annexe - Coordonnées du centre d'un cercle circonscrit

$$x_O = \frac{\alpha}{-2} = \frac{1}{2\Delta} \begin{vmatrix} x_A^2 + y_A^2 & y_A & 1 \\ x_B^2 + y_B^2 & y_B & 1 \\ x_C^2 + y_C^2 & y_C & 1 \end{vmatrix}$$

$$y_O = \frac{\beta}{-2} = \frac{-1}{2\Delta} \begin{vmatrix} x_A^2 + y_A^2 & x_A & 1 \\ x_B^2 + y_B^2 & x_B & 1 \\ x_C^2 + y_C^2 & x_C & 1 \end{vmatrix}$$

Annexe - Affichage avec PyQtGraph