

Rapport 5

Choisir les parametres

Algorithmes

Comme le graphe representant les images ont un sommet relie a tous les autres sommets par une arete de poids 0, alors tout arbre couvrant de poids minimal est compose uniquement de ce sommet initial et de toutes les aretes nulles. L'algorithme RSL se base sur un MST pour fournir une tournée. Ainsi RSL n'est pas pertinent pour la resolution de nos problemes. \ Nous allons donc nous concentrer sur les valeurs des parametres de notre implementation de l'algorithme HK.

HK: Construire une tournée a partir d'un 1-arbre

Dans la plupart des cas, en tout cas avec nos machines, nous n'obtenons part l'algorithme HK non pas une tournée mais un 1-arbre approchant cette tournée.

Notre strategie est donc la suivante:

- Lancer l'algorithme HK
- Atteindre un de nos criteres d'arret (a savoir: limite d'iteration, limite de temps, obtention d'une tournée)
- Si on obtient une tournée: la convertire en image reconstruite
- Sinon: (en s'inspirant de RSL)
 - Transformer le 1-arbre en arbre en enlevant le sommet initial
 - Lire cet arbre en post-ordre
 - Convertir la tournée ainsi obtenue en image

HK: Adapter nos parametres

Les parametres sur lesquels nous intervenons sont les suivant:

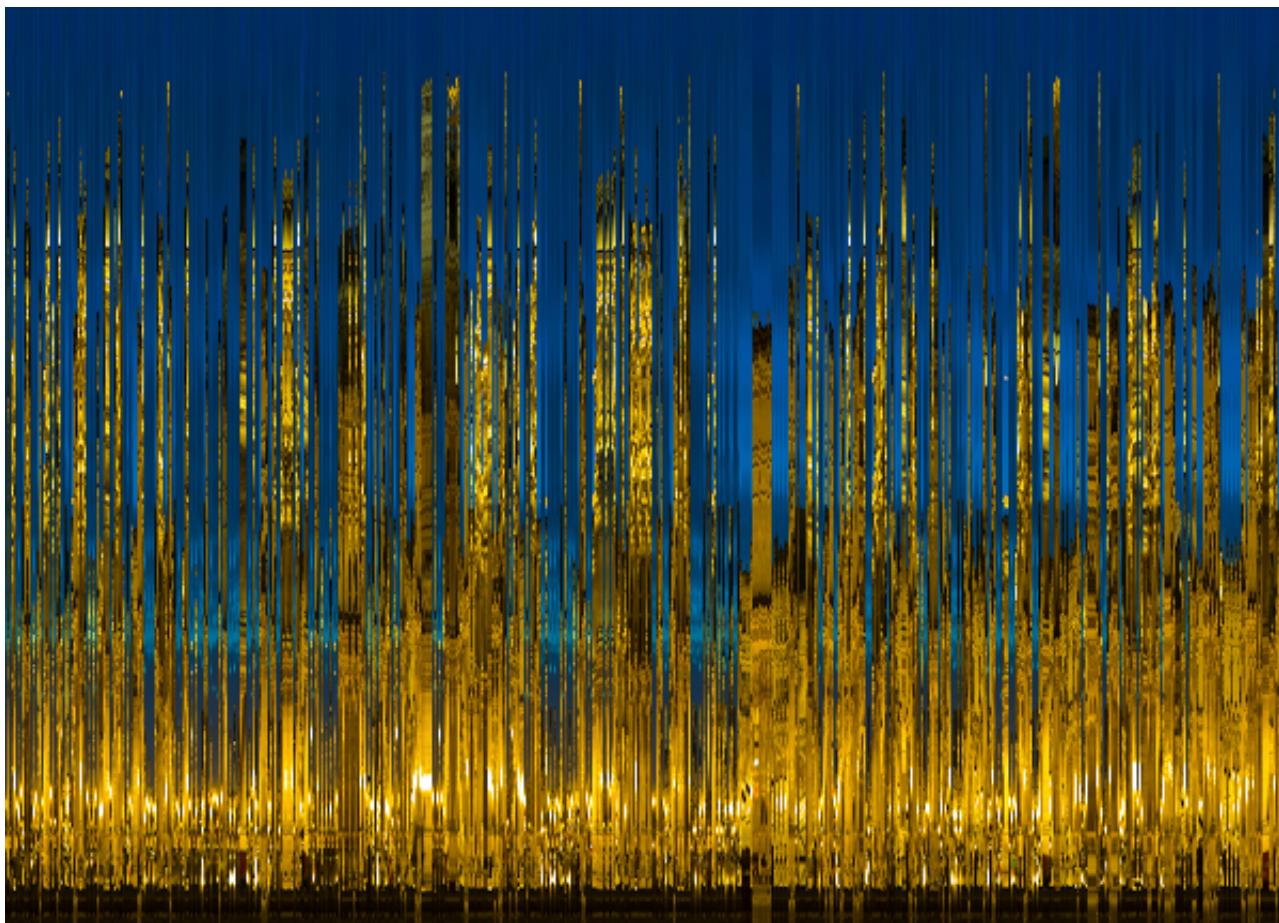
- Algorithme pour produire un MST (Prim ou Kruskal)
- Pas adaptatif ou non (True ou False)
- Time limit (par defaut 2 minutes)
- iteration limit (par defaut 10 000)
- Valeur de pas (un interval de valeurs)

De nos experiences lors des precedentes phases du projet, nous avons retenu que Kruskal etait systematiquement plus rapide que Prim. Nous nous contentons donc de cet algorithme. Pour les memes raisons, nous ne considerons que des situation ou le pas est adaptatif.

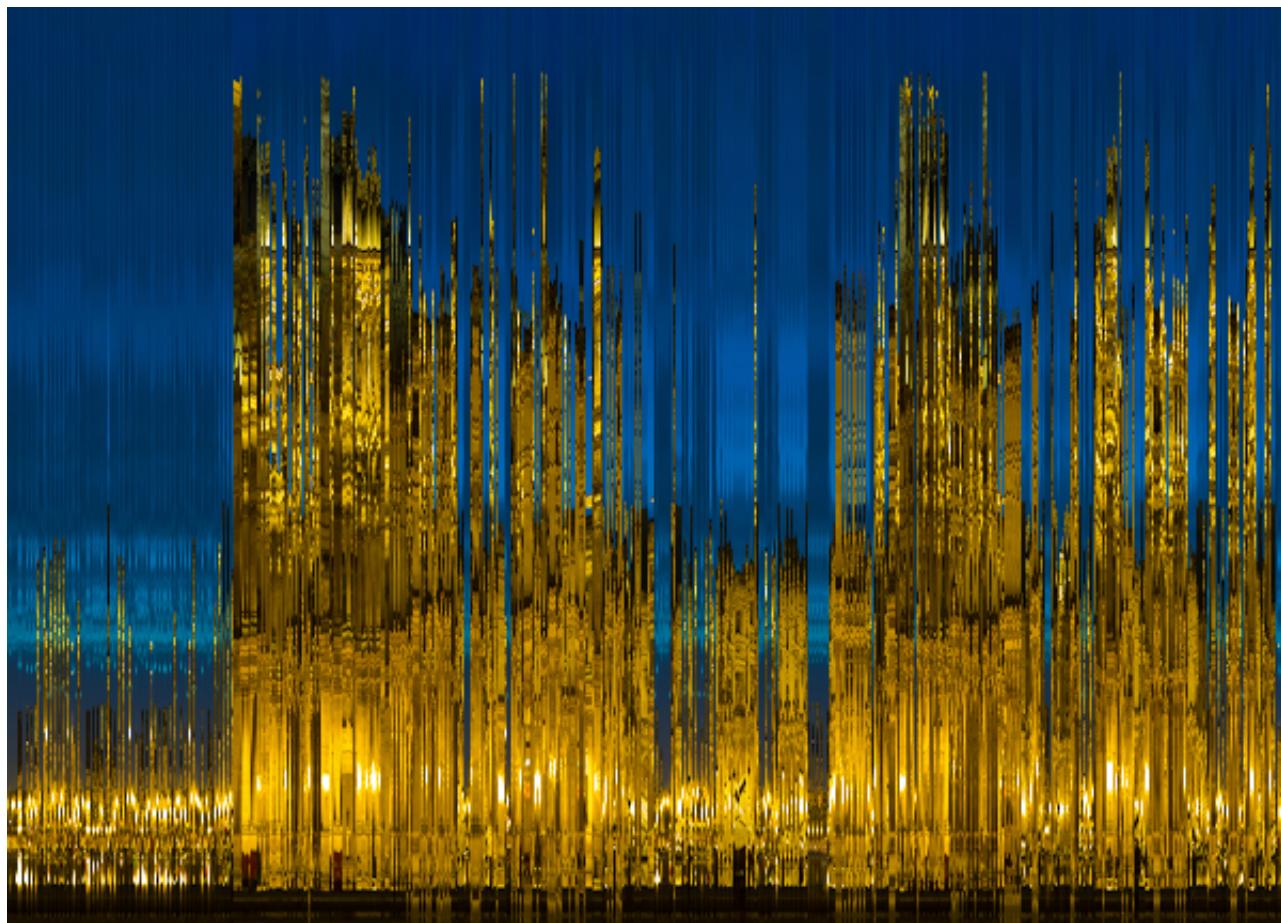
Impact du pas

Tout d'abord, considerons pour une meme image, avec une limite de temps de 2 minutes, differentes valeur de pas:

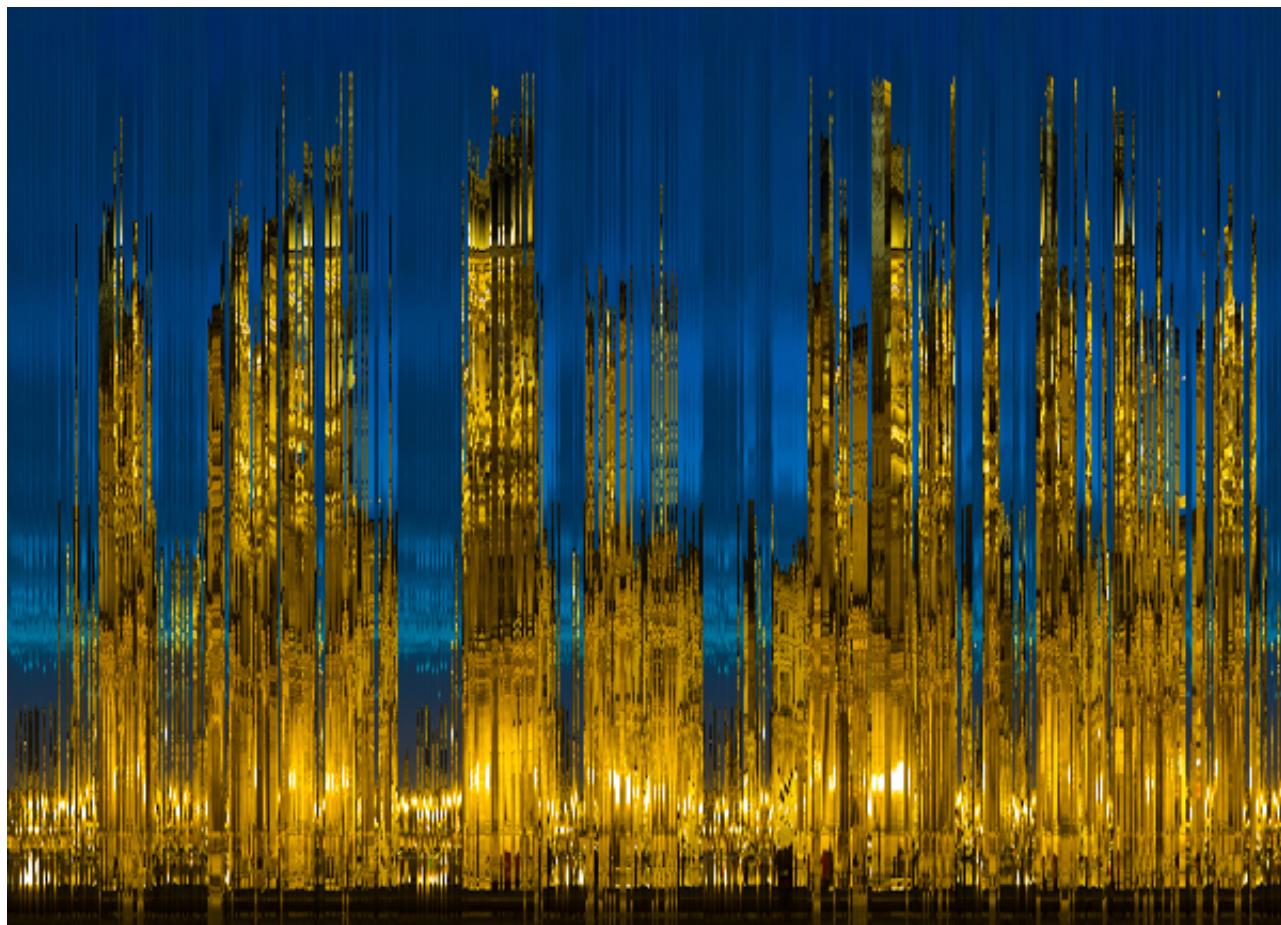
1. [0.5, 1.0], lenght = 125534.58



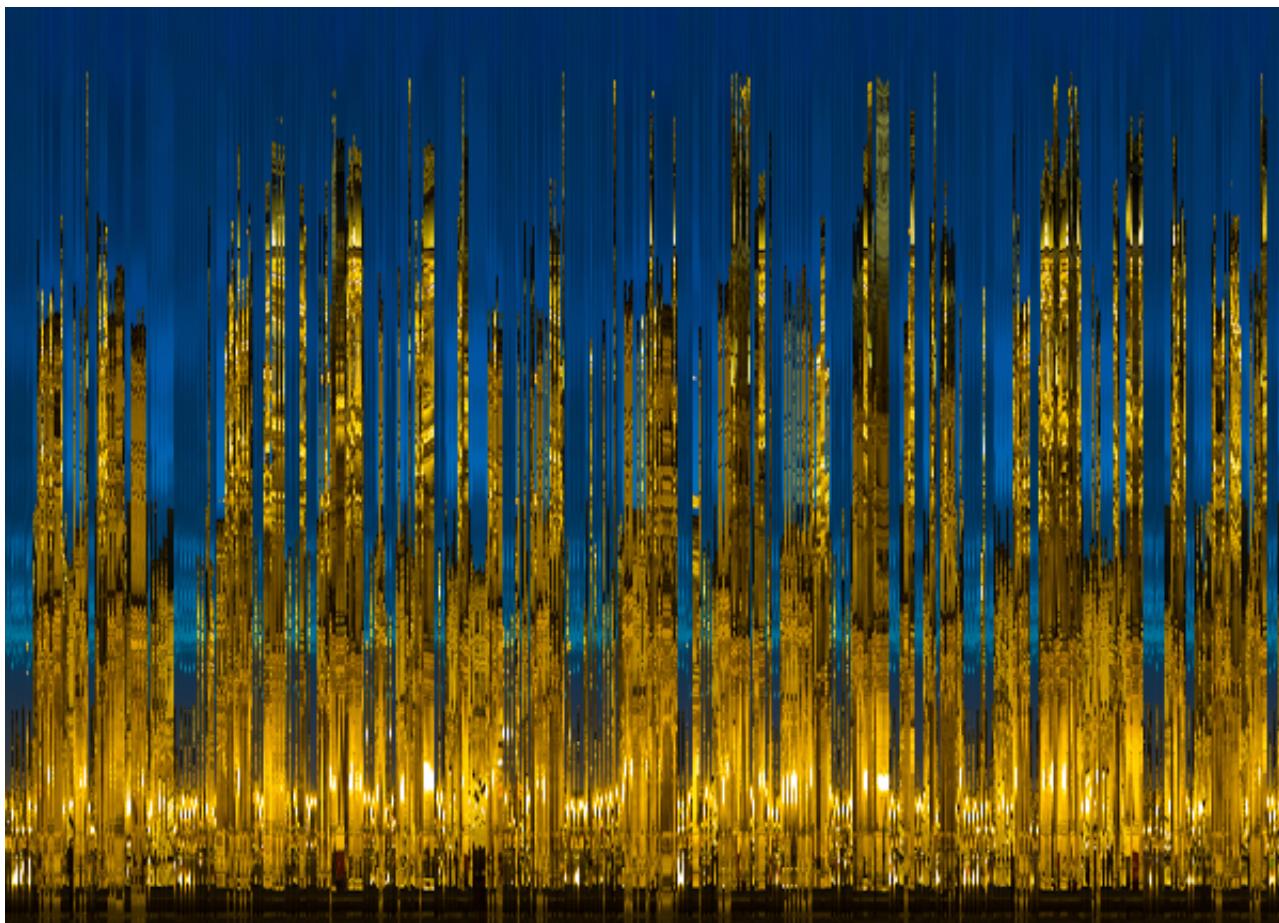
2. [0.5, 50.0], lenght = 125534.58



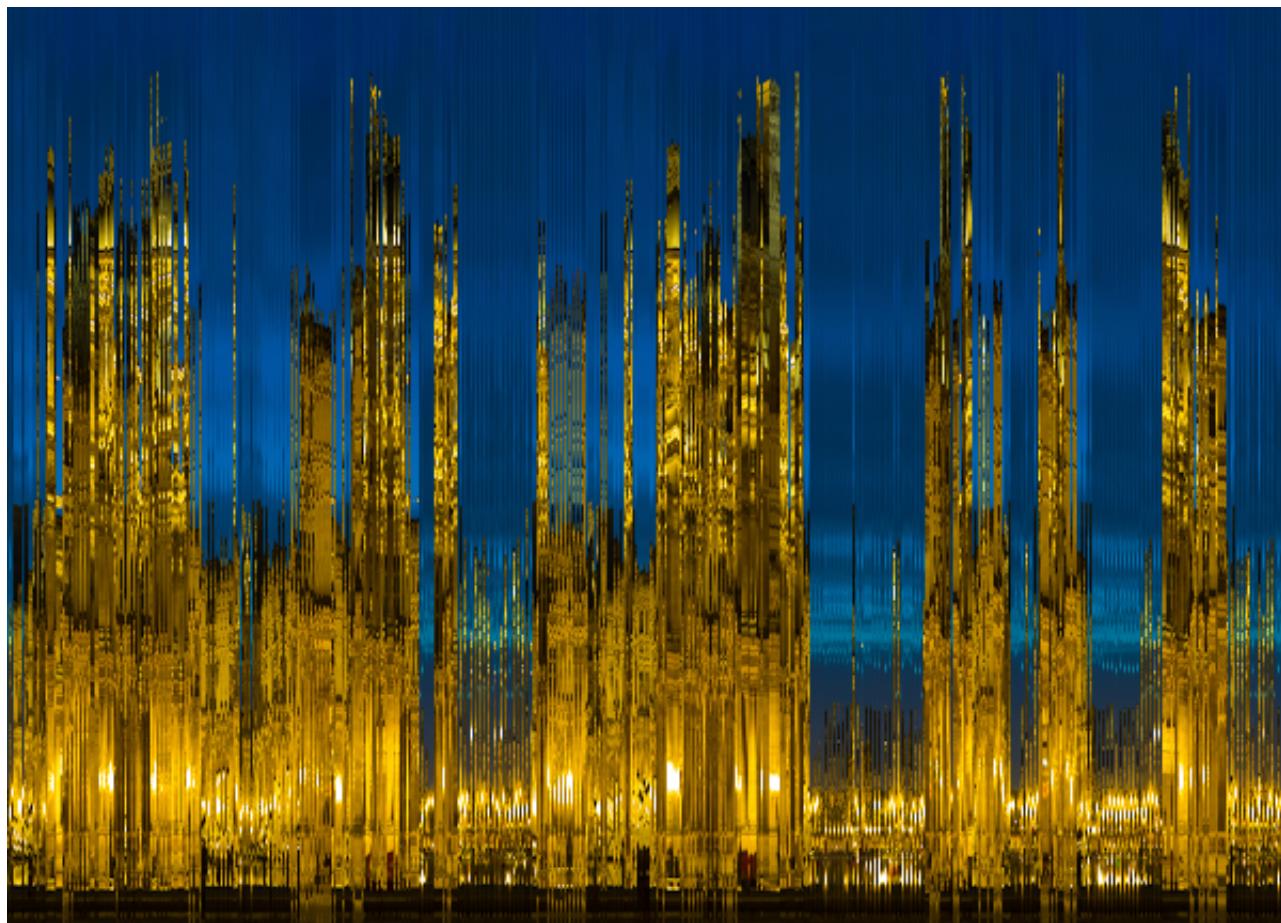
3. [0.5, 100.0], length = 125534.58



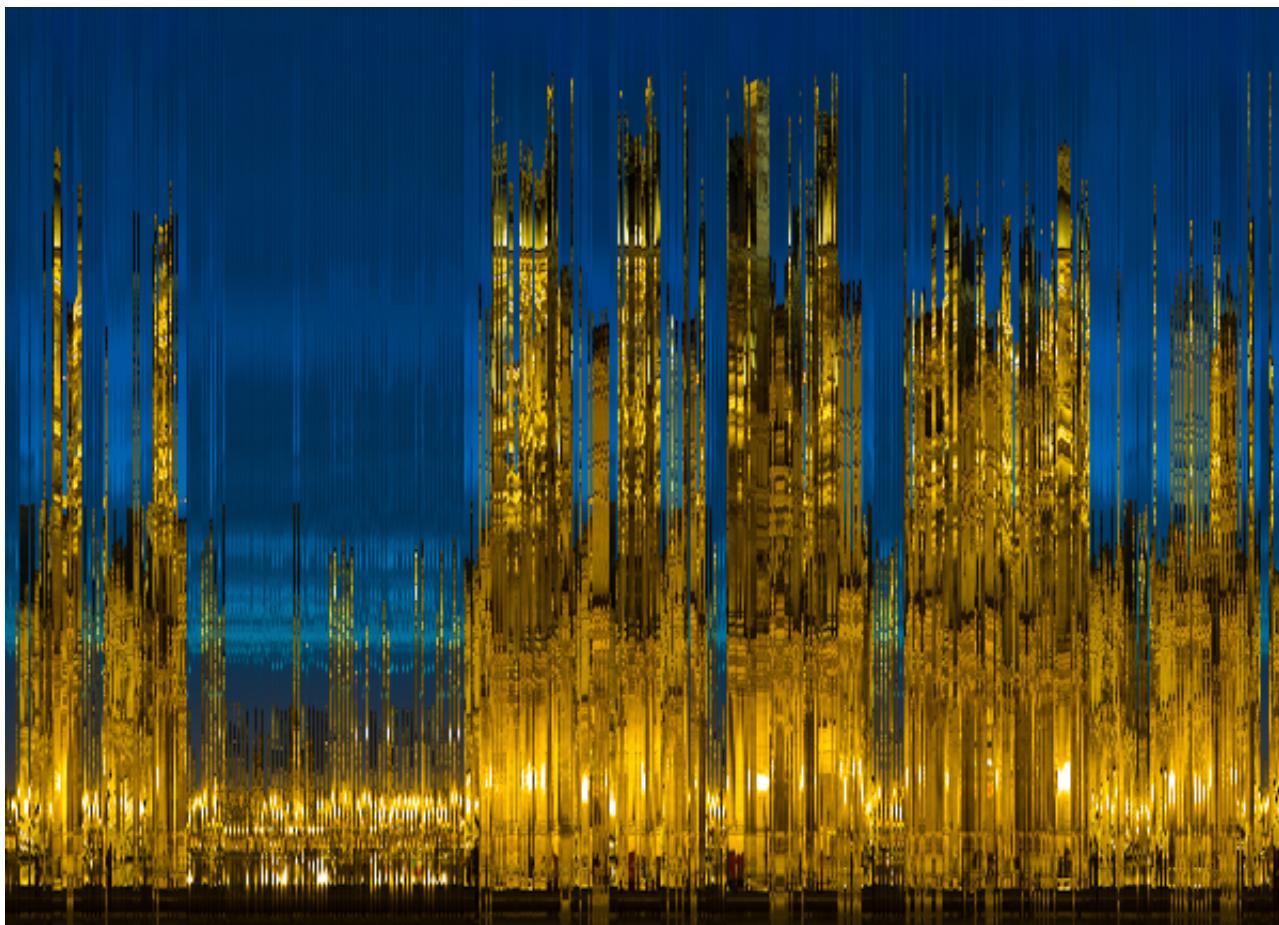
4. [1.0, 10.0], lenght = 125534.58



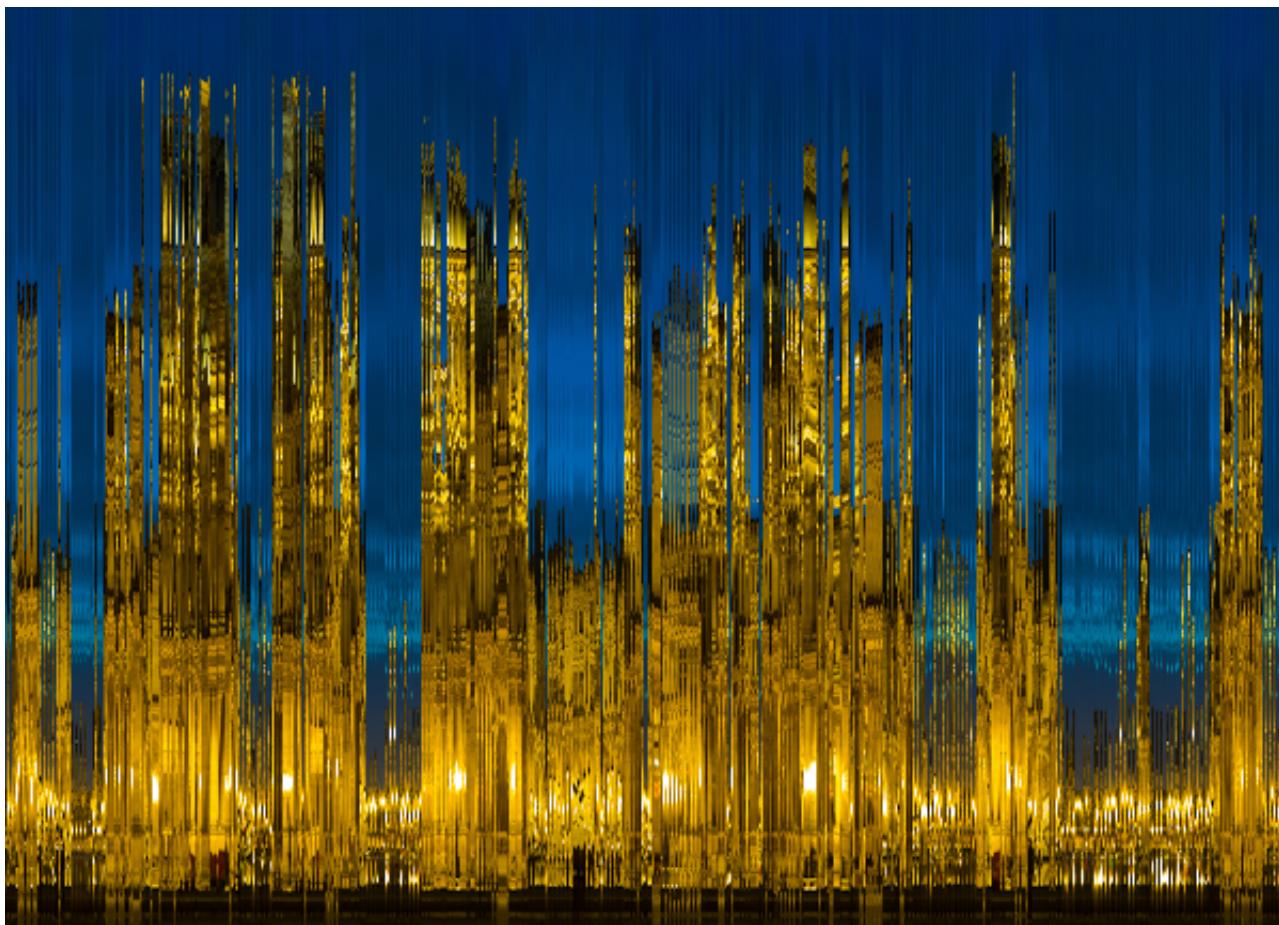
5. [20.0, 30.0], lenght = 125534.58



6. [20.0, 30.0], lenght = 125534.58, sans pas adaptatif



7. [30.0, 50.0], length = 125534.58



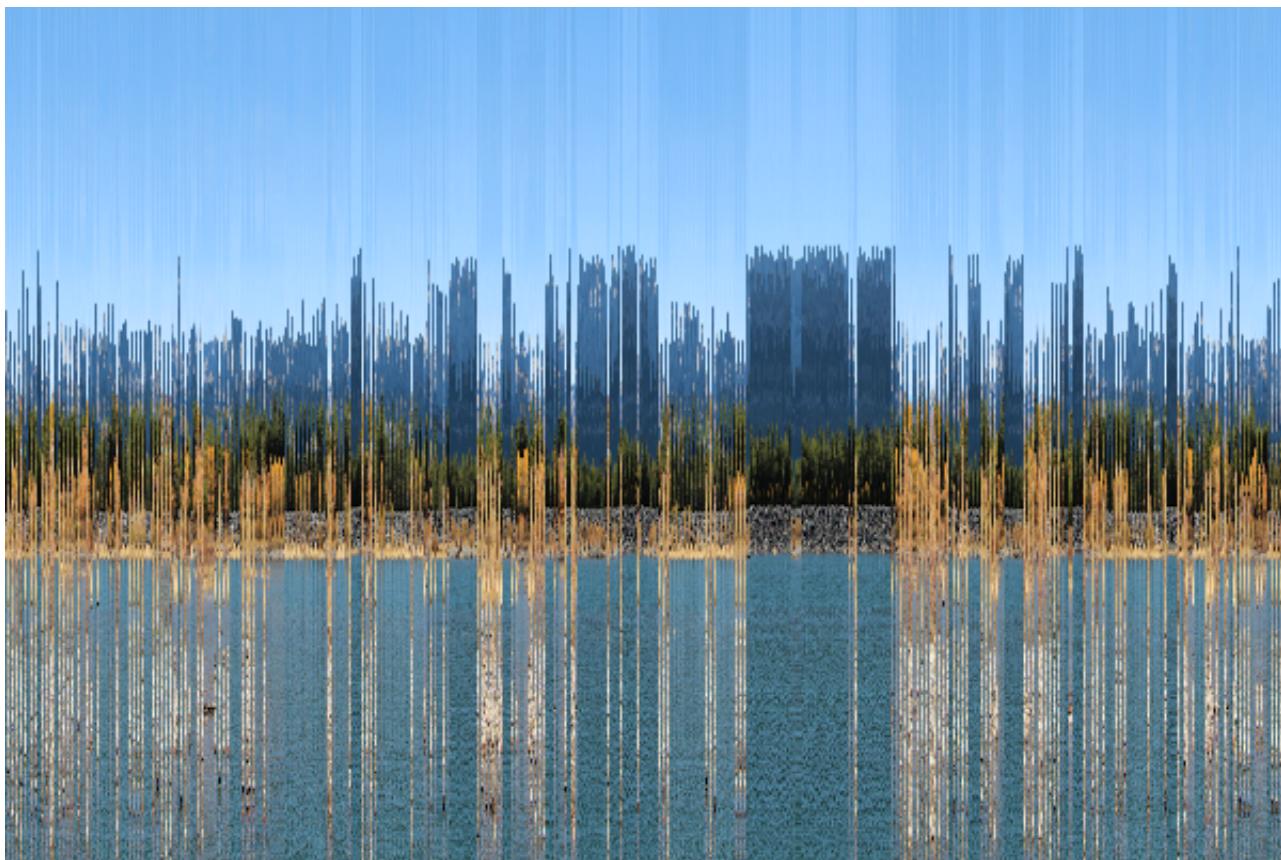
Si la longueur des tournées obtenues est toujours la même, on peut tout de même remarquer que les images 5,6 et 7 se rapprochent plus de l'image originale que les 4 premières. En particulier il semble que le pas adaptatif, s'il permet d'avoir une meilleure idée de la structure de l'image, n'aggregé pas beaucoup de colonnes voisines.



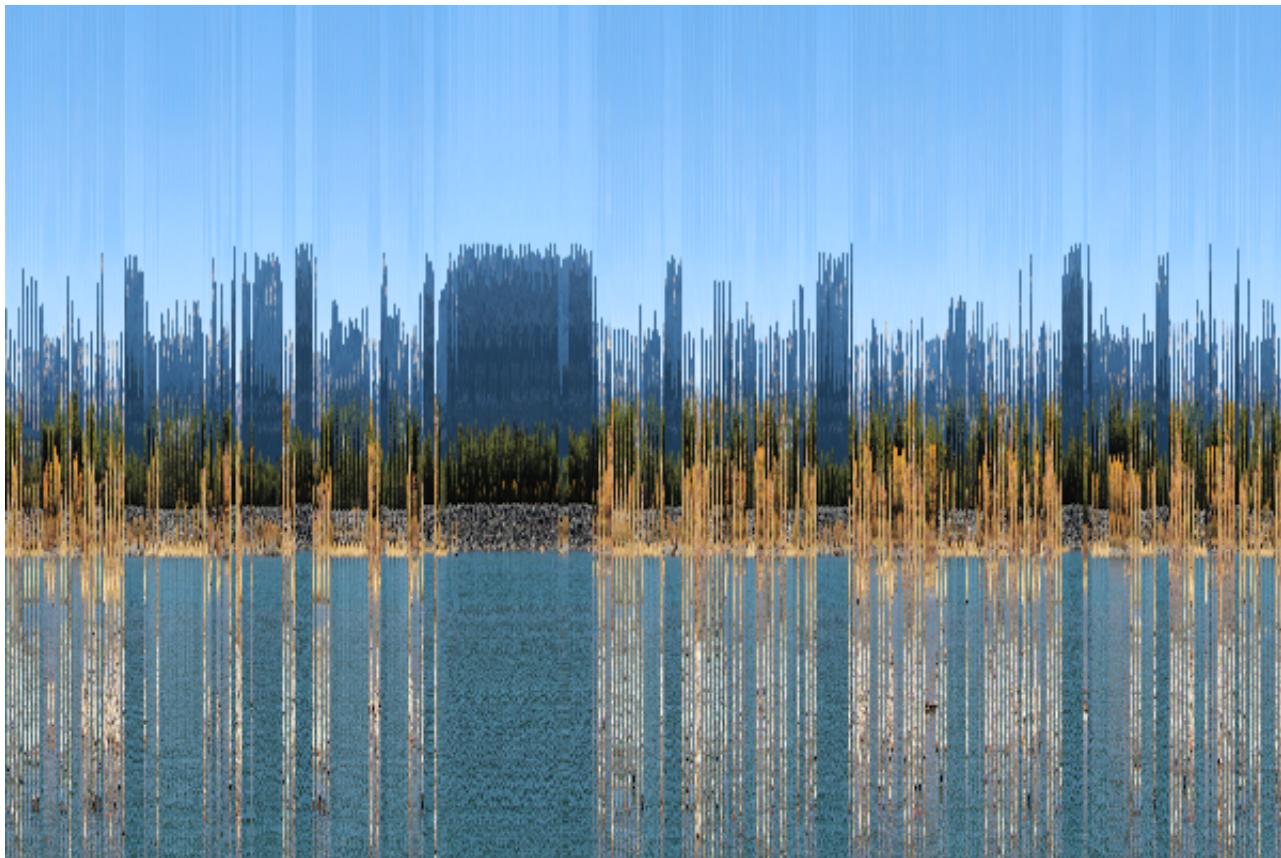
Impact du temps

Tout parametre egal par ailleurs, nous avons voulu etablir si la duree d'execution permettait de produire des solutions plus satisfaisantes. Il nous semble (et nous arrivons a la meme conclusions sur plusieurs instances) que la solution trouvee en 2 minutes n'est pas si differente de celles trouvees en 5, 10 ou 15 min.

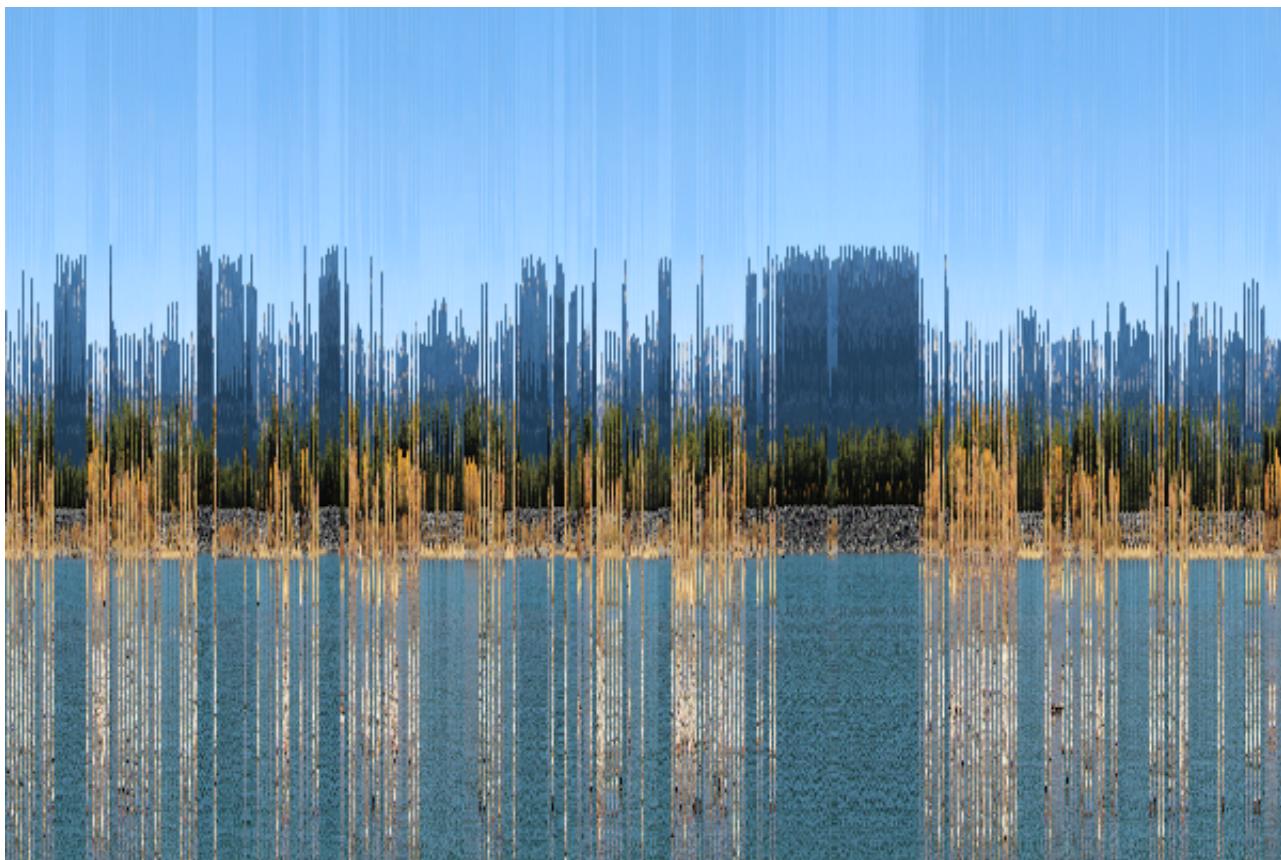
1. 2min, lenght = 118111.05



2. 10 min, lenght = 118111.05



3. 20 min, lenght = 118111.05



Originale:

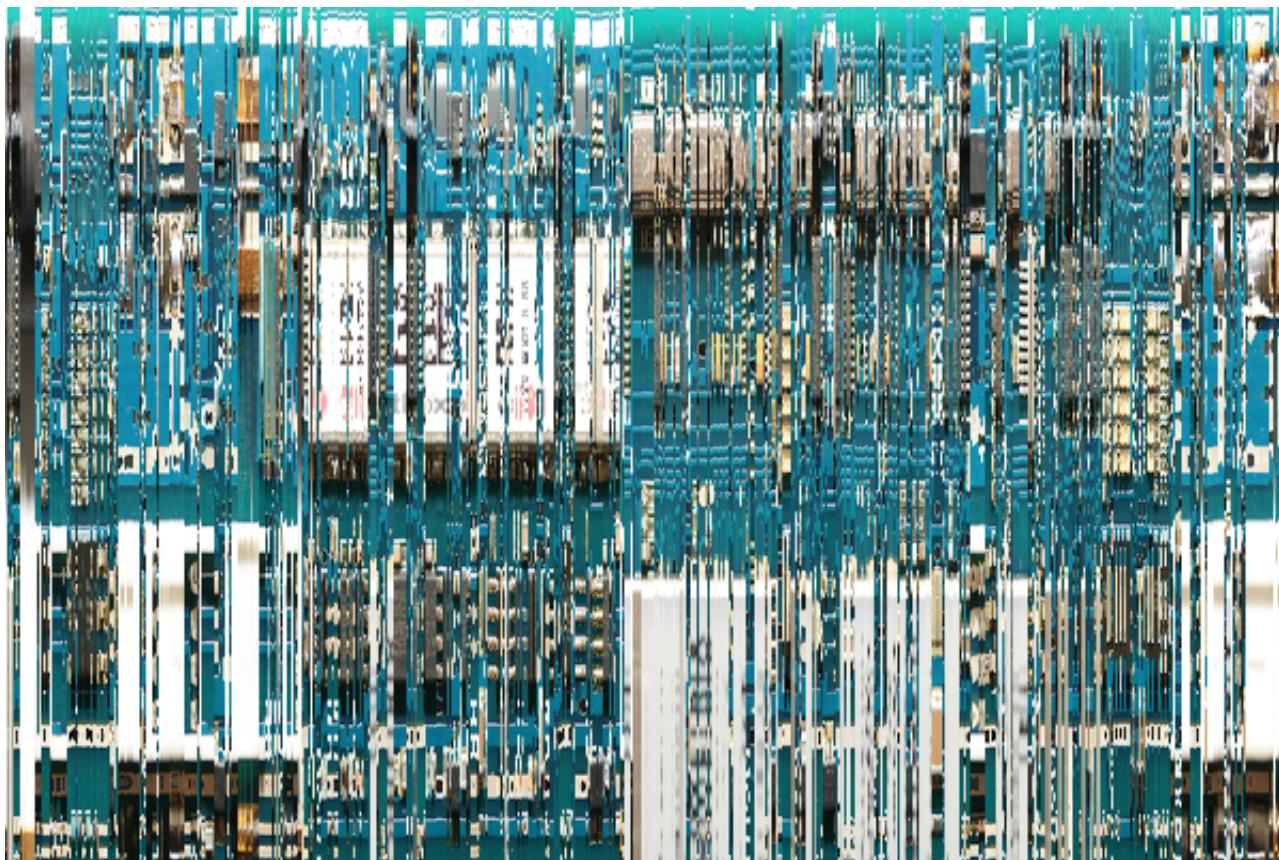


L'amélioration ne nous semble pas valoir l'investissement en ressource.

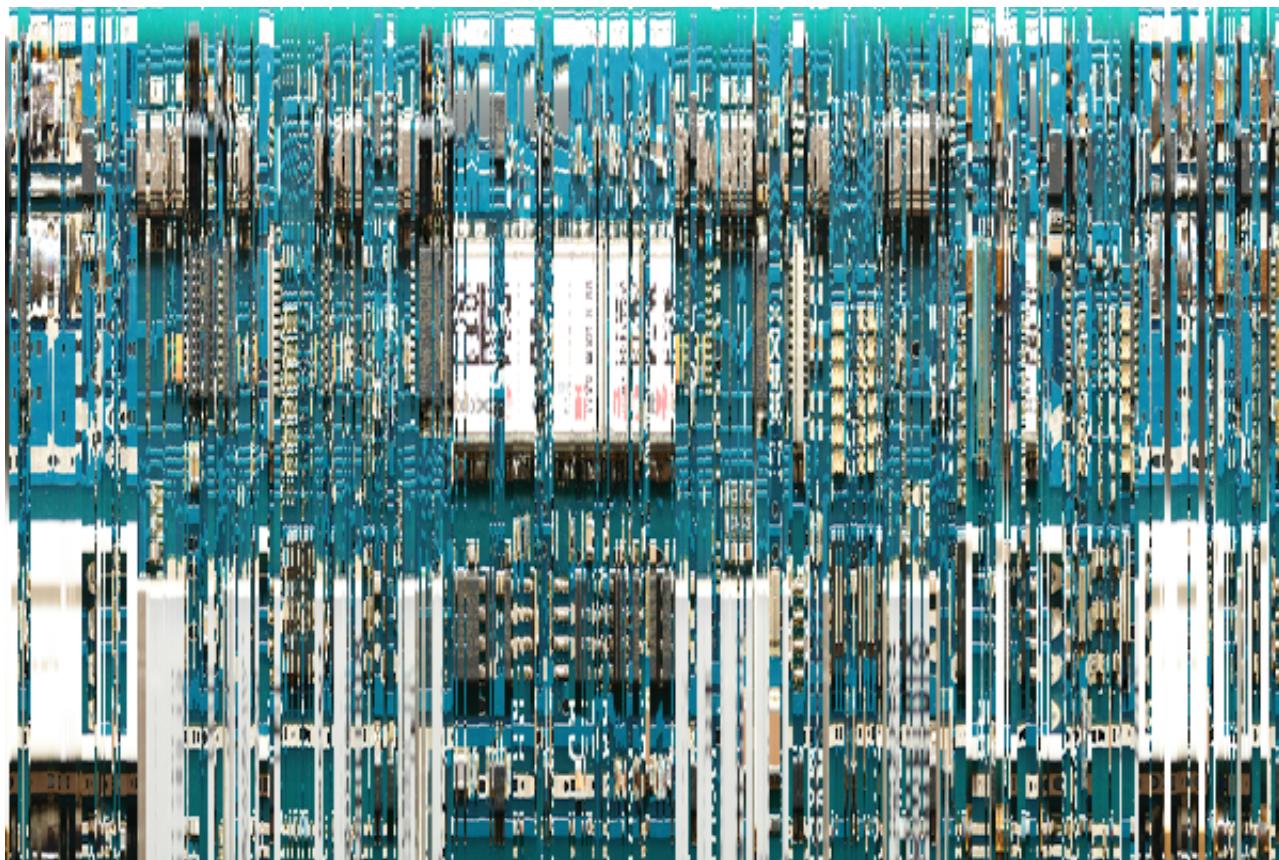
Pas adaptable

Enfin, nous comparons nos résultats avec et sans le pas adaptable sur quelques instances avec une limite de temps de 2min, un pas de [1.0, 2.0] et l'algorithme de Kruskal pour les MST:

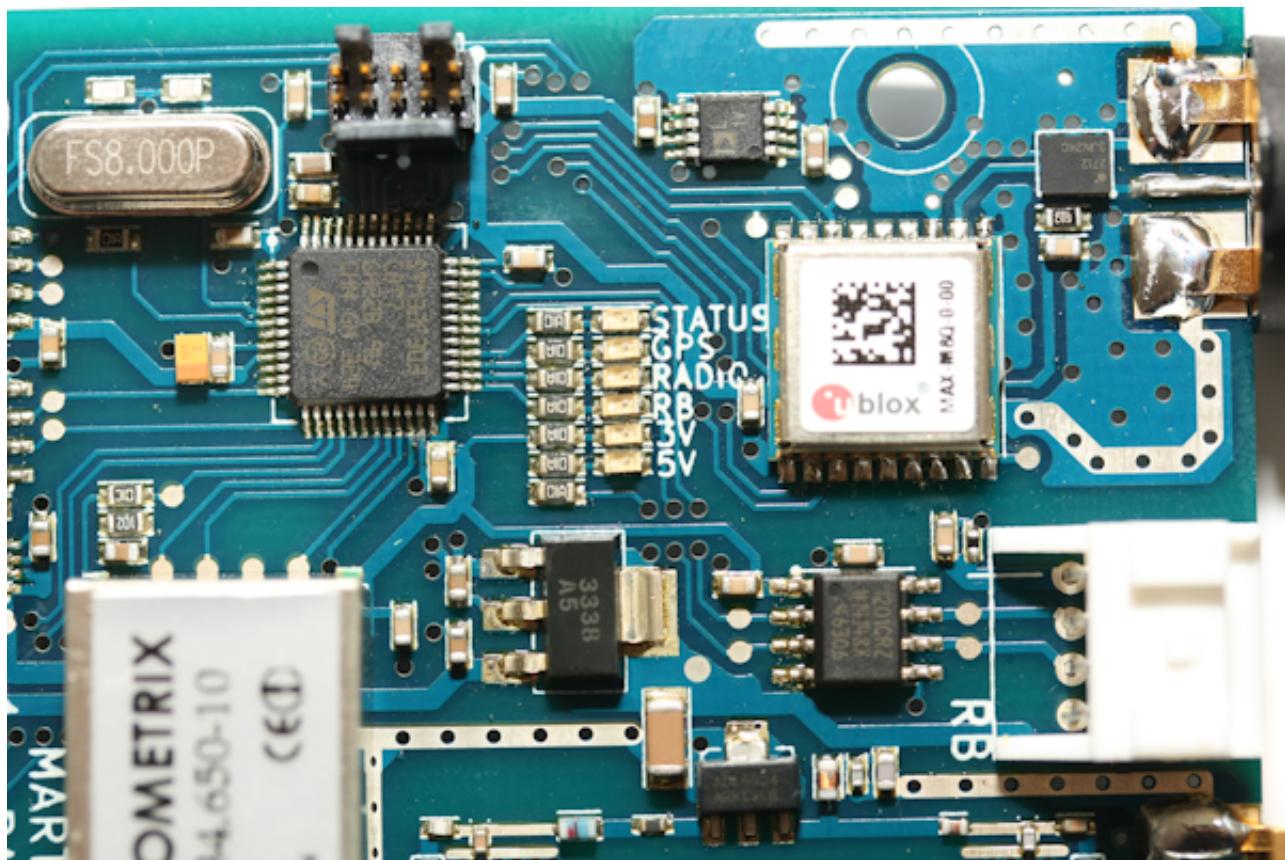
- Sans le pas adaptable



- Avec le pas adaptable



- Image originale:



On observe que bien que le pas adaptable donne des "clusters" de colonnes plus larges, les coutours des éléments principaux restent bien visible avec ou sans le pas adaptable.