Sciper	code décomposition (max: 2pts)	function size violation	*Comments on code decomposition*
271829	2,0		Bonne décomposition mais qui peut être améliorée en ne laissant que les déclarations de variables et appels aux fonctions principales du projet (lecture, seuillage, filtrage)...
273712	2,0		Bonne décomposition! Bonne utilisation des define! Bravo! Prochaine fois essaie de plus commenter ton code, mais c'est très bien
274888			
275185	2,0		A proper understanding of the basic principles of abstraction and reutilization led to a well-organized code When sending a vector to a function, you don't need to send the dimensions as these can be obtained through accessors. Passing arguments by reference is used when modifications are necessary, but you might want to think of constant references as well as a way to enhance performance.
275336	1,5	filtrage	Excellent decomposition, pay attention to your "filtrage" function : try to create subfunctions.
276861	2,0		Good decomposition. When passing a parameter by reference to improve performance, use const if the variable is not to be changed (e.g. display_fin_image)
279202	1,0	main	Your main is too long. Try to create more sub-functions and comment less. The code must be understandable by itself. Although, the decompositions principle seems understood.
282211	2,0		C'est un code très beau et très lisible ! Bravo !
282515	2,0		Très bonne décomposition, bravo!
282708	2,0		Rien à redire sur la décomposition du code. Bonne utilisation des passages par référence !
282712	2,0		bonne décomposition. plusieurs choix coûteux du point de vue des performances: passage par valeur de gros vector au lieu de passage par référence. de même la boucle 333-340 peut être remplacée par la définition d'une variable intermédiaire k = image_destination[i][j].
282756	2,0		Excellent décomposition du code, code extrêmement modulaire! La logique est plutôt bonne.
282844	2,0		Bonne décomposition en fonctions Bien vu de passer tes "gros" paramètres par référence constante quand tu ne les modifie pas
282988	0,5	main, filtrage	Quand elle est faite, la decomposition est très bonne. A faire partout. Le passage par référence de grands paramètres serait plus performant, et les vecteurs de vecteurs mériteraient un typedef
283389	2,0		Très bonne décomposition du code, propre et logique. La méthode main est très bien implémentée.
283391	2,0		Très bonne décomposition du code. A revoir en détail les avantages et désavantages de passer par référence ou par valeur (le premier mode permet d'avoir un programme plus prévisible, le deuxième d'obtenir de meilleures performances).
283419	0,5	main, image_finale_filtre	Bonne décomposition mais pas totalement aboutie. main mérite d'être plus concentré, la lecture des paramètres PPM pourrait être délégué à une fonction. image_finale_filtre() devrait aussi exploiter d'autres fonctions
283470	2,0		Bonne structure, plusieurs petites opérations ont été extraites dans des fonctions c'est une bonne pratique.
283792	2,0		Parfait. Le principe de décomposition est bien compris et bien appliqué !
284194	2,0		Bonne décomposition, la structure est logique et claire. Bonne utilisation des références et des références constantes.
284284	2,0		Très bon travail sur la décomposition du code. Les fonctions sont bien utilisées. Très bon usage des commentaires également : bon travail !
284564	2,0		Bonne décomposition du code ! Pense à passer tes vector par référence pour plus de performances
284867	2,0		Tu décomposes bien ton programme entre les divers fonctions. Tu pourrais utiliser un peu plus les passages par référence pour faciliter la modification des différents tableaux que tu utilises
284896	2,0		Très bonne décomposition, bravo! Par contre ce serait mieux si tu passais certains de tes arguments par référence (les struct par exemple) tu gagnerais pas mal en performance. N'hésite pas à écrire plus de commentaires dans ton prochain projet.
286317	2,0		Bonne décomposition, bravo! Mais attention au grand nombre d'arguments passés à certaines fonctions. Si tu as trop d'arguments c'est souvent le cas que tu peux mieux faire. Il y a pas trop d'intêret à passer des types primitifs (tels que int, double, etc) par référence. Ce sont plutôt les types "lourds" tels que les vecteurs ou struct que tu devrais passer par référence pur éviter la copie.
286386	2,0		La décompositio est très bonne mais peut être améliorée en rétrécissant la fonction main (création de sous-fonctions).
286509	0,0	main	Le programme consiste seulement en un main() et aucune autre fonction, difficile à lire et le code n'est pas réutilisable Cependant le code est bien commenté

286570	1,0	load_data, filtrages	La logique du programme est raisonnable, mais comme les implémentations sont énormes, il aurait été convenable de les diviser en plusieurs méthodes auxiliaires. Par exemple, définir des fonctions pour récuperer chaque entrée, et les appeler dans load_data, ou encore définir une méthode de filtrage pour un seul pixel et l'appeller dans la troisième boucle for imbriquée de filtrages.
286641	2,0		Très bonne décomposition du code, rien à redire ! Bon usage des passages par référence.
286721	2,0		Excellente décomposition du code, très logique et facile à comprendre!
286734	2,0		Bonne décomposition en fonctions, pense à ajouter des commentaires pour expliquer que fait chaque fonction et comment
286955	2,0		Code parfaitement bien décomposé ! Bravo !
286962	1,0	main	Ton main est beaucoup trop long, ta function filtrage est à la limite mais passe. BOnne utilisation des références, ton code est bien en général, mais pense à faire un main plus petit la prochain fois
287054	2,0		Very good decomposition with nice commentary, nice!
287056	2,0		Nice decomposition ! Continue
287428			
287433	2,0		La décomposition est bonne mais deux soucis majeurs : pourquoi le main n'appelle-t-il qu'une seule fonction ? Aussi, utilise des prototyes de fonctions plutôt qu'écrire le main à la fin de ton programme.
287443	2,0		La décomposition est parfaitement classique et logique: rien à reprocher, mais définir au moins quelques fonctions auxiliaires peut rendre le code plus clair et modulaire.
287459	2,0		The code seem well-organized. The different basic ways to pass parameters are properly handled As a way to be more efficient and to get used to browsing documentation: don't reinvent the wheel! What could you do instead of writing a function *reading_unused_char* ?
287480	2,0		Décomposition correcte, attention cependant aux nombreuses boucles imbriquées. Les vector<vector<int>> peuvent être passés par référence constante pour de meilleures performances.
287529			
287609	1,0	main	Le principe de décomposition est bien compris mais mal appliqué : chaque partie décrite par les commentaires doit faire office d'une fonction à part au nom explicite. De cette manière, le main respecterait la taille imposée.
287725	2,0		Bonne décomposition en fonctions Pense à passer tes structs par référence constante si tu ne les modifie pas, pour des raisons de performance
288166	2,0		Plusieurs fonctions pourraient être regroupées. C'est bien de diviser les différentes tâches du programme en fonctions, mais si tu te retrouves avec des fonctions de 3 lignes, il peut-être intéressant d'essayer de les regrouper. Sinon, c'est tout bon !
288252	2,0		The use of functions is relevant and the basic principles seem to be understood. There main function is kept short and clear.
288287	1,0	main	Tu aurais pû avoir une seule fonction d'affichage. La décomposition du code est correcte.Penses à utiliser les typedef, et mets des const pour les passages par référence qui ne doivent pas changer.
288312	2,0		Penses aux passages par référence. La décomposition du code fait sens et est bien claire, c'est bien.
288423	2,0		Bonne décomposition du code ! N'hésite pas à rajouter un peu plus de code dans ta fonction main, tant que tu respectes la limite de 40 lignes par fonction.
288530	2,0		Excellente structuration de main, il est très facile à comprendre ce qui se passe. Bon usage de struct et typedef, mais ces derniers pourraient être passés en paramètre par référence pour une meilleure performance.
288553	1,0	read_file, filter_step	Les fonctions read_file et filter_step doivent être davantage décomposées pour mieux respecter le principe de décomposition : crée des sous-fonctions. A part ça, le main est très bien décomposé !
288580	2,0		Bonne décomposition, bonne utilisation des références. Bravo!
288820	2,0		[warning] filtrage_un_pixel (length) Code factoring is OK ; nice use of references.
288825	2,0		Bonne décomposition, essaie de passer des arguments par référence et tu verras un gain en performance!
288838	2,0		Très bonne décomposition, bravo!
289028	2,0		Bonne décomposition, bravo!

289067	2,0		Bravo pour la decomposition. Fonction main très élégante
289121	2,0		Excellent travail sur la décomposition du code en fonctions;
289122	2,0		Bonne implémentation du principe d'abstraction ! Certaines de tes fonctions sont cependant peu justifiées, attention également à utiliser le principe de réutilisation plutôt que de créer plusieurs fonctions identiques.
289188	2,0		Penses à utiliser le passage par références. La représentation en une dimension de l'image rend l'utilisation du tableau très difficile, tu aurais pû utiliser un vector<vector<int>> par exemple. La décomposition du code est très bien !
289234	2,0		Bonne structure, même si la lecture d'entrée mériterait sa propre fonction.
289285	2,0		Bonne décomposition en fonctions pense à passer les gros argument par référence constante si tu ne les modifie pas
289292			
289304	2,0		Bravo, bonne décomposition
289324	2,0		Bonne décomposition du code, plutôt classique. L'ensemble est logique.
289596	2,0		Well decomposed code
290067	2,0		Bonne décomposition en fonctions, et utilisation des références y compris const
290470	2,0		Décomposition correcte, mais eviter les boucles for dans le main !
291182	2,0		Bonne décomposition ! Continue ainsi.
295745	2,0		Modularisation correcte, attention à la fonction filtrage qu'il aurais mieux valu décomposer un peu plus.
295758	2,0		Décomposition classique et logique du code en fonctions. Cohérent en général, aucune faute flagrante. Il aurait été possible de modulariser encore un peu plus le code, sans pour autant que cela soit absolument nécessaire. Les signatures des fonctions sont plutôt adaptées. Par contre, toutes les fonctions ont le type de retour void, il aurait été plus adapté qu'au moins quelques fonctions retournent plutôt des valeurs que d'utiliser exclusivement la mutabilité.
295762	2,0		Très bonne décomposiiton, bravo!
295784	2,0		An overall good decomposition. You might want to add more subfunctions for your main tasks when they require too much to do within a single function. For instance, affiche_ppm seem to lack clarity, probably because you tried to shrink it down to the accepted size a bit forcefully... And affiche_ppm is actually... the true name of main! As much as I appreciate very short main functions (sort of table of contents of the code), you might know that creating a function whose only role is to call another one is pretty much useless... and this also applies to main. Keep it short*and clear* . You've reached a point where your main funciton doesn't really provide any sort of insight into your code. Beyond passing by simple references, think of constant references instead of values when you don't want to change the orginal variable, to avoid significant and unnecessary copies.
295798	2.0		Bonne Décomposition
295808	2,0		Bonne décomposition et utilisation du passage par référence ; pense à utiliser des constref si tu ne modifies pas le paramètre
295810	1,0	main	Le principe de décomposition est très bien appliqué sur les dernières lignes de ton main. Il est un tout petit peu trop long, définir des sous-fonctions pour la lecture de données par exemple pourrait régler ce problème.
295813	1,0	main	Bone idée pour les fonctions de contrôle. Mais il faut plus décomposer main. Plus de structures et de typedef peuvent aider aussi.
295821	2,0		L'utilisation de références comme arguments de fonction n'est pas nécessaire si la fonction return void. Bonne décomposition du programme, quelques parties pourraient être simplifiées, mais c'est très bien.
295855	2,0		Très bonne modularisation, les noms de fonctions sont explicites bravo !
295890	2,0		Très bonne décomposition ! N'hésite pas encore plus décomposer en ne laissant que des appels de fonctions et des déclarations de variables dans le main.
295913	2,0		Bonne décomposition du code et bonne utilisation des passages par référence !

295923	2,0		Bonne structure, et toutes les fonctions rentrent dans les limites, mais les triples for-loops inpriqués pourraient être découpés pour plus de lisibilité.
295927	2,0		Your code is properly structured and agreeable to browse Beyond using references, you could also use constant references to avoid unneccesary copies of significant amounts of data.
295954	2,0		Décomposition assez claire, le filtrage aurais cependant pu être plus décomposé.
296023	2,0		Modularisation correcte, plusieurs fonctions sont un peu longues et auraient pu être décomposées. Cela reste néanmoins une structure logique.
296026	2,0		Très bonne décomposition qui pourrait être encore mieux en regroupant certaines fonctions dans des fonctions de plus haut niveau (par exemple les fonctions saisie).
296032	2,0		Bonne décomposition, mais la lecture d'entrée mériterait sa propre fonction.
296046	2,0		Pas de prototypes de fonctions. La structure est assez logique mais attention aux nombres élevées de variables passées en paramètre et aux nombreuses boucles imbriquées.
296048	2,0		Décomposition classique, logique. Par contre même si tout à fait juste mettre la fonction main tout à la fin plutôt que tout au début est assez surprenant.
296050	1,0	main	Le principe décomposition est bien appliqué mais la fonction main est un peu trop longue. N'hésite pas à créer plus de sous-fonctions pour encore mieux décomposer le problème.
296057	2,0		Très bonne modularisation, toutes les étapes du programme sont décomposées et regroupées de façon cohérente bravo !
296092	2,0		The code is indeed subdivided into functions and the main function for instance shows the work that has been achieved regarding the basic principles described. However it might be necessary to push this further, or to find ways to increase the legibility of your code (cf. next comment section). This problem can be noticed with the *filtrage* function.
296093	1,0	main	Le code est décomposé très proprement et logiquement, vraiment dommage pour la méthode main qui dépasse d'un tout petit peu la taille maximale plus la tolérance. Il peut par contre être utile parfois de retourner des valeurs plutôt que d'avoir uniquement des fonctions à type de retour void, même si dans votre cas cela aurait été utile uniquement si vous aviez d'autres fonctions auxiliaires. Bonne utilisations des commentaires, cela rend le code bien plus agréable à lire!
296142	2,0		Très bon travail sur la décomposition du code.
296165	2,0		Bonne décomposition!
296169	2,0		Modularisation correcte, attention à bien utiliser des références constantes quand les variables passées en paramètres n'ont pas de raison d'être modifiées (fonction affichage par exemple)
296189	2,0		Très bonne utilisation des const references pour les arguments des fonctions. Décomposition du code très claire, juste ta fonction sous_filtrage qui était un peu lourde.
296202	2,0		Bonne décomposition du code (à part une fonction à la limite d'être trop longue (filtre())) Bonne idée pour l'utilisation des structs, mais est-ce que le code serait facilement adaptable si on augmentait le nombre de composantes de couleurs ?
296232	2,0		You could improve the structure of your code by going further in the creation of functions. You probably noticed the problem trying to shrink down your functions below the 40-line limit. When you have such a problem (or when it feels like it is OK only because your code is very dense – which basically amounts to the same) , consider it as a warning sign: you should maybe think of writing subfunctions, which whould also help make things clearer by helping not losing sight of the main concern of the higher level function.
296235	2,0		Bonne décomposition du code. Tu pourrais peut-être regrouper certaines fonctions qui font des choses assez similaires, pour avoir un code moins complexe.
296253	2,0		Bonne décomposition du code et bonne utilisation des passages par référence !
296351	2,0		Décomposition très claire, certaines fonctions auraient pues être combinées entre elles au lieu de les appeler toutes à la suite dans le main.
296372	2,0		Structure intéressante, des références constantes peuvent être utilisées pour améliorer les performances.
296399	2,0		Bonne décomposition du code, plutôt classique.
296442	2,0		Bonne décomposition.
296443	2,0		Bonne décomposition, mais fais gaffe à passer des arguments par référence pour améliorer les performances!
296451	2,0		Rien à redire sur la decomposition des fonctions. Un conseil : tu peux aussi utiliser des passages par référence pour éviter des returns (parfois moins pratiques).

296465	0,5	main, filtrage	Bonne idée d'utiliser des structs: plus serait mieux. Typedef est utile pour l'utilisation répétée de vecteurs de vecteurs Les fonctions sont bien structurées, ces parties pourraient être des fonctions.
296487	0,0	main,lire_image_ppm,filtrage	Il y a plusieurs problèmes au niveau de la décomposition en fonctions comme leurs tailles ou leur pertinence
296573	2,0		Bonne décomposition du code. Il aurait été possible de modulariser un tout petit peu plus.
296575			
296642	2,0		Très bonne utilisation des fonctions ! On aurait pu imaginer en utiliser aussi pour la lecture des paramètres
296643	1,0	main	Bonne décomposition, mais la fonction main reste bcp trop longue
296644	2,0		Très bonne décomposition, bravo!
296759	2,0		Très bonne décomposition, la structure est logique et les noms de fonctions sont clairs. Les Images peuvent être passées par référenceconstante pour améliorer les performances.
296815	2,0		Bonne décomposition du code, plutôt classique.
296816	2,0		Excellente décomposition du code!
296820	0,0	main, filtrage, error_analyser	Le code doit être mieux divisé. Les fonctions longues sont bien structurées, il suffirait de diviser les blocs. Pour la lisibilité, un typedef pourrait remplacer les vecteurs de vecteurs. De plus, passer des vecteurs par référence pourrait être beaucoup plus performant.
296827	2,0		Très bonne décomposition, bravo!
296833	2,0		Bonne décomposition mais qui peut être améliorée en ne laissant que les déclarations et les appels de fonctions de haut niveau dans le min.
296853	2,0		Très bonne décomposition ! Elle pourrait être améliorée en regroupant toute la partie lecture de donnée dans une fonction dédiée. Continue ainsi.
296867	2,0		Bonne décomposition, bravo!
296935	2,0		Code decomposition seems overall good, but don't hesitate to create subfunctions when things get dangerously long. For filtering for instance, if it was presented in a lighter way, it would be too long. About the C-style, as an example: wouldn't it be clearer, shorter and safer that*colorsUsed* be a multidimensionnal *array* , passed by constant reference whenever neccesary?
296952	2,0		Bonne utilisation du principe d'abstraction ! Petit conseil : n'hésite pas à utiliser le passage par référence, souvent plus efficace qu'un passage par valeur, notament par exemple pour les vector et tableaux
296957	2,0		Dans l'ensemble le découpage des fonctions est très bon. Bonne utilisation du principe d'abstraction
296958	2,0		Excellent travail : très bonne utilisation des fonctions, typedef et références. Bravo !
296960	2,0		Bon usage de typedef. Une décomposition en nouvelles fonctions est préférable à trop de for-loops imbriqués. Le passage par référence de vecteurs serait plus performant.
297072	2,0		Good code decomposition You could pass large function parameters by reference to avoid copying them, even if you don't plan on modifying them (in which case use a const ref)
297075	2,0		Bonne décomposition, bravo!
297096			
297097	2,0		Code très bien décomposé :)
297151	1,5	filtrage	Code dans l'ensemble bien décomposé, tu aurais pu réduire le filtrage d'encore un niveau d'abstraction. Attention aussi, certaines de tes fonctions sont identiques : pense bien au principe de réutilisation.
297176	2,0		Bonne application du principe d'abstraction via les fonctions, beau travail.
297202	2,0		Penses à utiliser les passages par référence. Les fonctions plus_petite_diff_seuils et lecture_seuils peuvent être regroupées pour l'efficacité. bords_noirs réécrit aux mêmes endroits et parcourt toute l'image, donc assez inefficace. En général, fais attention à la complexité de tes fonctions ; tu gagnerais beaucoup en performances. Mis à part ça, le code est bien décomposé et structuré, c'est très bien.
297250	2,0		Attention à décomposer davantage tes fonctions, ta fonction "filtering" est beaucoup trop compacte. C'est à la fois source de bugs et plus difficile à tester.

297284	2,0		Bonne décomposition du code dans l'ensemble, même si tu aurais pu utiliser plus de passages par référence pour éviter de passer des gros tableaux entre plusieurs fonctions.
297290	2,0		Très bon découpage!
297768	2,0		La longueur des fonctions est correcte, mais dans la plupart des cas au dépend du respect du style et de sa clarté. Plutôt que de fermer toutes tes accolades sur la même ligne, utilise le principe d'abstraction pour séparer tes longues fonctions en plusieurs. Egalement, il serait mieux que tu utilises des prototypes de fonctions et que tu laisses le main tout en haut.
297797	2,0		The factoring is clear. Think of using constant references when passing arguments you don't want to create a resource-consuming copy for but that also shouldn't be modified.
297846	2,0		Bonne décomposition, bravo!
298006	2,0		Très bonne décomposition ! Le concept est bien compris et bien appliqué, continue !
298011	2,0		Très bonne décomposition qui est bien comprise au vu des commentaires mais chaque partie qui fait plus de deux lignes peut être remplacée par une fonction de plus haut niveau.
298386	2,0		Parfait ! Le principe de décomposition est bien compris et bien appliqué
299066	1,5	filtrage_fonc	Bonne décomposition dans l'ensemble, il manque juste une fonction d'affichage.
299245	2,0		Parfait, rien à redire ! Best décomposition so far.
299248	2,0		Très bonne modularisation, qui suit bien la logique du programme.
299274	2,0		Très bonne décomposition avec des noms de fonctions clairs. Tu pourrais l'améliorer en regroupant toutes les fonctions saisie ensemble par exemple.
299290	2,0		Des choses très interessantes, on voit que tu as bien étudié le problème. Par contre, du coup tu as moins pensé à l'efficacité de ton code : devoir passer de vector à vector<vector> est couteux, certains for peuvent être simplifiés. A part ça, la struct entrees est une très bonne idee.
299391	2,0		Bonne décomposition du code. La manière dont tu utilises et transmets un tableau unique pour beaucoup de fonctions par un passage par référence est clair et te permet d'éviter des "return" peu pratiques ou une multitude de tableaux. C'est bien !
299450	2,0		Bonne décomposition, bravo! Tu pourrais gagner en performance si tu passais aussi tes vector par reference.
299479	2,0		Très bonne décomposition, la fonction main pourrait quand même être clarifiée en ne laissant que des déclarations et appels de fonctions de haut niveau.
299492	2,0		Bonne décomposition, bravo!
299496	2,0		Parfait et très bien décomposé
299506	1,0	main	The principle of abstraction doesn't tell you to create sections within the main function with comments.. It tells you to create functions reflecting the different levels of abstraction your worked with. In this way *main* should be at the highest level, thus bearing a certain resemblance to a simple table of contents.
299554	2,0		L'utilisation des structures et le passage par références dans tes fonctions sont très bien ! Attention à garder tes structs plus légères et logiques. N'hésites pas à décomposer un peu plus tes fonctions pour éviter la duplication de code.
299561	2,0		Le code est assez bien décomposé. Bravo.
299645			
299667	2,0		Le principe de décomposition est bien respecté. Il pourrait l'être encore mieux en ne laissant que des déclarations de variables et des appels de fonctions de haut niveau. dans le main.
299689	2,0		Très bonne décomposition, bravo! ä certains endroits tu pourrais passer des arguments par référence pour éviter la copie (surtout quand tu apsses des struct)
299724	2,0		La fonction entree_entier est inutile. Sinon, le passage par const références est très bien, et la struct Couleur clarifie très bien !
299882	2,0		Utilises le passage par référence constante aussi. La décomposition de ton code est très bien, tu as pensé aux cas limites et la recherche dichotomique est super. C'est du très bon code, bravo !
299885	2,0		Bonne décomposition!
299892	2,0		Très bonne décomposition, bravo!
299894	2,0		Très proprement décomposé. Attention d'appliquer le passage par référence de vecteurs partout pour une meilleure performance.

299895	2,0		Evites de trop modulariser : saisir_entier est inutile car équivalente à "cin" qui est déjà une seule ligne. Penses à pasage par référence pour les performances. Le tableau frequences apparitions est assez lourd, le traitement pourrait se faire directement.Excellente décomposition, très lisible. test_validite très bien.
299964	2,0		Bonne structure générale, mais les fonctions pourraient être plus découpées. Par exemple, la lecture d'entrées mériterait sa propre fonction. Passer les structs et vecteurs par référence dans les fonctions peut aussi aider la performance.
299971	2,0		Très bonne décomposition!
300038	1,0	main	La structuration par fonction est clairement à revoir. Mais au moins, tu as utilisé les fonctions déjà créées et as réussi à séparer par foncionnalité ton codes avec des retours à la lignes.
300048	2,0		Le main est tout juste, essaye de le rendre plus court en definissant une sous-fonction dédiée à la lecture des données par exemple. Bon respect de la décomposition sinon.
300089	2,0		Très bonne décomposition! Bravo! Tu n'as pas besoin de passer les types primitifs tels que int, double, etc par référence. TU veux passer par référence les containeurs ou des struct ou objets que tu as crés pour éviter la copie. Dans le cas des types primitifs cela n'est pas tant utile que ça, mais c'est bien d'y avoir réfléchi.
300093	2,0		Bon choix de forme pour les structs et les typedef. Pense à utiliser des noms de fonctions plus descriptifs que lecture_1 lecture_2, ...
300114	2,0		Bien décomposé. Bravo. Pense peut-être à faire des fonctions avec des passages par références ou plus de foncions tout simplement plutôt que de renvoyer des structures. Pour un gros projet, cela posera problème.
300131	2,0		Bon découpage du code en fonctions. Pense à passer tes vector par référence plutôt que par valeur, c'est plus efficace
300166	0,0	main, Affichage,filtrage	Il y a un effort de décomposition mais hélas cela ne marche pas. Attention aux longueurs de fonctions !
300247	1,0	main	Ta fonction main dépasse très largement la limite autorisée ! L'utilité des fonctions est justement de pouvoir diviser ton code pour éviter une situation de ce style. Plusieurs parties de ta main seraient pertinentes à intègrer dans des fonctions dédiées (les tests d'erreurs par exemple)
300253	2,0		Bonne décomposition du code bravo, attention tout de même aux fonctions comportant beaucoup d'arguments
300284	2,0		Très bonne décomposition du code, logique et intéressante même. Utiliser le main peut être une bonne idée (il est dans ce cas-là superflu).
300411	2,0		Trés bonne décomposition ! Continue ainsi.
300443	2,0		Excellent usage de constantes et structures!
300456	2,0		Bonne décomposition en fonctions Pense à passer tes gros paramètres (p.ex. les vectors) par référence constante si tu ne les modifie pas, pour des raisons de performance
300540	2,0		The code is well organized. Nice use of const& Is the use of Dimension necessary? Don't forget the accessor *size* ().
300556	2,0		The code is properly organized. Some improvements could be made through the use of references.
300558	2,0		The main shows instantly that you really worked towards clarity with a relevant use of functions Think of using const& to prevent unnecessary copies.
300559	0,5	sortir entrer filtrer	You could have added some instructions in the main function, especially as some other functions are too long You created some high-level functions, but you should have taken the code factoring further, writing some more subfunctions. Think of passing parameters by (constant) reference (beyond the single bool&), especially to save resources.
300594	2,0		Très bien et très clairement décomposé ! bravo !
300615	2,0		Penses aux passages par référence. L'utilisation des define rend le code clair, l'array<int, 3> est une bonne idée. La décomposition du code est très bien, bravo.
300620	2,0		Très bonne utilisation du principe d'abstraction ! Petit conseil : passer tes vector par référence est plus efficace que par valeur
300649	2,0		Très bon travail sur les fonctions : les principes d'abstraction et de réutilisation sont bien implémentés. N'hésite pas à passer tes vector en paramètres via des références, c'est plus eficace
300664	2,0		
300669	2,0		Bonne décomposition en fonctions Par contre ce n'est pas vraiment une bonne pratique de passer des paramètres qui ne servent pas à la fonction
300780	1,0	main	Correctement décomposé. Hélas pour main, il y a deux lignes de trop.
300782	2,0		Bravo ! Ton code est décomposé de manière juste. Une tâche vaut une fonction et ce sans superficialité.

300803	2,0		Très bonne décomposition bravo! Pense à passer tes arguments par référence pour améliorer les performances. Code bien commenté.
300840	2,0		Penses aux passages par référence constante. La struct est une bonne idée, attention toute fois à ne pas mettre des choses dedans qui n'ont rien à y faire, comme ton copie1dim.
300845	2,0		Excellente décomposition ! Le code pourrait être encore plus clair et compact en regroupant les acquisitions par exemple.
300882	2,0		Penses à utiliser les passages par référence. Mis à part ça, la recursion est très interessante. Très bien !
300883	1,5	filtrage	You did divide your code into some functions but you could have gone further, especially with *filtrage* . I wouldn't have put const& all the time for parameters... but I believe this is open to debate.
300886	1,0	main	On voit que tu as bien commencé à séparer en fonction mais tu aurais dû continuer. Code agérable à lire sinon.
300899	2,0		Bonne décomposition
300910	1,0	main	Il manque encore du travail de décomposition.
300930	2,0		Bonne modularisation, qui suit bien la logique du programme.
301001	2,0		Bonne décomposition du code. N'hésite pas à plus utiliser le passage par référence pour éviter de transférer des gros tableaux entre les différentes fonctions.
301030	2,0		Bonne décomposition, mais pense à être plus explicite dans tes noms de fonctions pour que le code soit plus clair : par exemple entre filtrage et filtrage_total
301033	2,0		Très bonne utilisation des fonctions, bravo ! Toutefois, je note plusieurs fonctions identiques : pense bien au principe de réutilisation
301061	2,0		Penses aux passages par référence, et n'hésites pas à utiliser des typedef pour la clareté. Tu aurais pu décomposer plus ton code pour améliorer la lisibilité (éviter les for imbriqués)
301070	2,0		Bonne décomposition du code, mais n'hésite pas à mettre plus de code dans ta fonction main.
301107	2,0		Bravo, Bonne décomposition
301221	2,0		Bonne décomposition, bravo!
301225	2,0		Bonne décomposition, bravo!
301226			
301265	2,0		Très bonne décomposition, continue ainsi !
301266	1,0	main	La fonction main est beaucoup trop longue et on n'arrive malheureusement pas à comprendre les différentes étapes de la résolution sans se plonger dans la technicité du code. Essaye de laisser dans le main seulement tes déclarations de variable et tes appels de fonctions de haut niveau. Les détails doivent rester dans les fonctions de plus bas niveau.
301268	2,0		Bonne décomposition du code, mais c'est une bonne idée de mettre des noms évocatifs aux fonctions plutôt que des acronymes. Il en va de même le nom des variables et des arguments. Remplacer par exemple FLT par filter.
301271	1,0	main	La décomposition du main est d'une certaine manière respectée mais chacune des sous-parties décrites en commentaires doivent être dans des fonctions autre que main. Main ne doit contenir que les déclarations de variables et les appels de fonction de haut niveau.
301272	2,0		Excellente décomposition, Bravo !
301291	2,0		Bonne Décomposition, la fonction main est très claire
301295	1,0	main	Beaucoup trop de code dans ta fonction main ! Il ne faut pas hésite à distribuer les opérations de ton programme sur plusieurs fonctions différentes. Sinon, bon usage des commentaires.
301300	2,0		Bonne décomposition
301303	0,0	main	Le code se trouve exclusivement dans la méthode main. C'est dommage, la logique était pourtant assez aisément modularisable dans le code, surtout vu l'organisation de la méthode main par étapes grâce aux commentaires, qui sont d'ailleurs utilisés de manière admirable.
301305	2,0		Très bonne décomposition, continue ainsi.
301319	2,0		La décomposition est correcte, logique. Attention, la fonction initialisationSeuils est trop longue, presque plus de 44 lignes, il aurait fallu la diviser en plusieurs méthodes auxiliaires.
301327			

301358	2,0		Penses à utiliser des passages par référence. La décomposition est très propre, la récursion dans filter est très interessante. Penses aux performances (combiner des boucles, ne pas recréer des tableaux à chaque filtrage)
301366	2,0		Logique du code intéressante, avec une bonne modularisation et l'utilisation de nombreuses structures.
301367			
301369	2,0		Bonne décomposition! Bravo!
301395	2,0		Très bonne décomposition bravo!
301403	2,0		les enums sont une bonne idée, struct aussi et les typedef sont très utiles. Tu utilises bien le passage par const reference. Ton algorithme pour le filtrage est très interessant, on voit une bonne compréhension du probleme. C'est très bien !
301407	2,0		Bonne décomposition
301411	2,0		Bonne décomposition du code, propre et logique, assez classique.
301412			
301418	2,0		Bonne décomposition, bravo!
301437	2,0		Beyond the use of simple references, think of constant references to avoid unnecessary copies The principles of abstraction and reutilization seem to be properly implemented.
301438	2,0		Très bonne décomposition, bravo!
301446	2,0		Bonne décomposition bravo!
301450	2,0		Très bon travail !
301452	2,0		Structure logique. Penser à utiliser des références constantes afin d'optimiser l'éxécution.
301458	2,0		decomposition assez logique. Ce qui l'est moins est le choix de passer par valeur des gros tableaux alors qu'un passage par référence constante serait plus efficace. Inutile de déclarer les vectors avec des dimensions nulles (lignes 52,53). Enfin, pourquoi choisir le type double pour les indices ?
301464	2,0		Excellent travail au niveau de l'utilisation du principe d'abstraction.
301469	2,0		Tu as fait une décomposition à l'extrême avec plein de petites fonctions mais c'est un code très bien conçu et très homogène ! Vraiment bravo !
301482	1,0	main	Attention, le principe de décomposition est bien compris mais pas appliqué : chaque partie de ton main doit être mise dans une fonction séparée. De cette manière, en lisant le main, nous saurions la logique du code sans devoir rentre dans le détail des algorithmes : main est un SOMMAIRE.
301486	1,0	main	Tu as bien décomposé ton code. Il reste toutefois la fonction main qui est trop grande.
301491	2,0		
301494	2,0		Bonne décomposition, bravo!
301496	0,5	prendreValeur main	The main function should be far shorter and clearer. Ideally, just design it so it looks like a table of contents of your program, so it provides the highest level of abstraction. By reading it, one would thus know instantly what the main steps are. Do not hesitate to create subfunctions when a given function gets too long: it could be a warning sign it will also lack clarity if you leave it this way.
301548	1,0	main	Il faut plus décomposer le code en fonctions. tu pourrais aussi passer tes vectors par référence constante à tes fonctions pour des raisons de performance
301560	2,0		Utilises des const pour les passages par référence quand tu ne veux pas que la fonction modifie l'entrée. La décomposition est bien dans l'ensemble, l'utilisation de la struct est une bonne idée. Attention à bien comprendre ce qu'un typedef est, et ce qu'un constexpr est.
301578	2,0		Bonne décomposition, bravo!
301581	2,0		Bonne décomposition, Bravo!

301595	1,5	filtrage	Dans l'ensemble, la décomposition est bonne.
301602	2,0		La décomposition est absolument excellente, le code est d'une clarté parfaite, et extrêmement modulaire en plus. Bravo!
301607	2,0		Excellente décomposition, le principe est compris et bien appliqué !
301633	2,0		Bonne décomposition du code, plutôt classique.
301641	2,0		Bien décomposé, main très compact. Au lieu de passer plusieurs valeurs d'une structure en paramètre, il est plus propre d'y mettre toute la structure. Une fonction interne est aussi préférable à 6 control statements imbriqués.
301653	1,5	seuillage	Modularisation correcte, pense à faire des "sous-fonctions".
301688	1,0		Tu pourrais mieux décomposer ton main, t'as des check d'erreur que tu pourrais mettre dans une fonction et réduire la taille de ton main. Il est un peu trop long et tu perds des points pour ça. A part cela ton code est bien décomposé et ton define du vector est intéréssant. Tu pourrais aussi passer des arguments par référence pour améliorer les performances de ton programme (cela éviterait la copie de tes struct dans les fonctions qui les appellent). Essaie aussi de print les lignes la prochaine fois.
301689	2,0		Très bonne décomposition, bravo! Bonne utilisation des references et des define.
301707	2,0		Très bonne décomposition en fonctions, bonne utilisation des références et constref
301708	2,0		"Bonne décomposition en fonctions Bien vu de passer tes ""gros"" paramètres par référence constante quand tu ne les modifie pas"
301711	0,0		
301718	2,0		Rien à redire au découpage, main est particulièrement clair. Excellent usage de struct.
301720	2,0		Bonne structure solide. Les vecteurs de vecteurs seraient plus lisibles sous forme d'un nouveau type, et les structures ed contrôle à cinq niveaux sous forme de fonction.
301750	2,0		Très bonne décomposition, continue ainsi ! Tu pourrais même regrouper les fonctions de lectures entre elles.
301766	2,0		Bonne décomposition mais essaye de le rendre encore plus compacte et explicite en ne laissant que des appels de fonctions de haut niveau et des déclaration de variables !
301786	2,0		Bonne décomposition!
301829	2,0		Tu as clairement décomposé tes fonctions !
301834	1,0	main	Tu aurais pu tout de même faire des fonctions. Sinon, tu as bien séparés par des retours à la lignes tes fonctionnalités.
301836	2,0		Bonne décompostion bravo
301837	2,0		Bonne décompostion bravo
301844	2,0		Bonne division des différentes tâches du programme en fonctions. N'hésite pas à inclure plus de code dans ta fonctior main.
301848	1,5	filtrage	La décomposition est encore un peu basique, beaucoup de boucles for imbriquées, beaucoup de fonctions plutôt longues.
301850	2,0		Bonne décomposition en fonctions, pense à ajouter plus de commentaires pour expliquer que fait chaque fonction et comment
301863	2,0		Penser aux passages par référence. La struct Format est une très bonne idée !! N'hésites pas à décomposer ton code en plus de fonctions, pour clarifier et modulariser.
301873	2,0		Excellente décomposition !
301881	2,0		Excellent decomposition ! The main is short and easy to understand. Continue !
301900	2,0		bonne décomposition mais 3 paramètres inutiles pour entree_image() ; ils devraient être déclarés localement à la fonction.
301924	1,0	main	Your main is too long !! Create as much subfonctions as you need to respect the decomposition principle.
301927	2,0		Excellente structure, et les zones délimitées la rendent agréable à lire. Seul détail, vu que tu utilises plusieurs fois des vecteurs de vecteurs, pense à typedef pour les remplacer.
301940	1,0	read_thr,ope n_ppm	On se retourve facilement dans ton code mais tu as plusieurs fonctions trop longues. Pense à mieux décomposer
301949	2,0		Très bonne décomposition en fonctions, bonne utilisation des constref, bonne utilisation des typedef, structs et constexpr :)

301950	2,0		Bonne décomposition en fonctions Passer les gros paramètres par référence est bien, mais passer des types primitifs par réf est plus lent que par valeur (puisqu'il faut copier un pointeur (32/64bits)**et** déréférencer le paramètre). De plus, lorsque tu passes un paramètre qui n'est pas modifié dans la fonction par référence, passe le par référence constante
301956	2,0		Les passages par valeur des vecteurs dans les fonctions peut être moins efficace qu'un passage par const référence. Tu crées aussi une nouvelle image à chaque filtrage, ce qui sera moins efficace que d'utiliser uniquement deux images. Mis à part ça et l'utilisation de magit numbers, très bonne décomposition et structure du code.
301959	2,0		Excellent découpage très compacte de fonctions!
301964	2,0		Good code factoring!
301965	0,0		Tu n'utilises pas de fonctions dans ton code : elles sont pourtant nécessaires à sa clarté et à l'application des principes d'abstraction et de réutilisation
301994	2,0		Bonne décomposition des fonctions, mais n'hésite pas à mettre plus de commentaires pour expliciter ce que font certaines fonctions
301998	2,0		Bonne décomposition du code, et bon usage des passages par référence.
302017	2,0		Parfait ! Rien à redire.
302031	2,0		Excellente application de typedef et bon choix de noms, mais quand une fonction atteint 7 couches de profondeurs, une fonction interne est recommandée. Le passage par référence est aussi plus important pour les vecteurs que les int.
302035	1,5	filtrage	Dans l'ensemble, la décomposition est bonne.
302038	2,0		Penses à utiliser les passages par référence. Sinon la structure et la décomposition du code sont très bien !
302041	2,0		Code factoring looks good. In bord_noir, do you really need to hav nbL and nbC as parameters? Can't you retrieve these values with the accessor size()? Use of regular references is a good thing, but when the function is not meant to change the actual value of the parameter, it is better to use const& (e.g. in afficher_PPM)
302072	0,0		
302076	2,0		Ton code est décomposé avec un nombre cohérent de fonctions: il n'y a pas de tâches sans foncion, Peut-être pense à un peu plus clairement décomposer encore tes fonctions pour éviter d'être à la limite du nombre maximal de lignes.
302211	2,0		Très bonne décomposition, bravo. Cependant regarde dans la documentation de std (google), tu auras des fonctions tels que std::stoi qui font la conversion string to int. Tu as bien utilisé des références. et bien commenté ton code Joli!
302276	2,0		Bonne modularisation, qui suit bien la logique du programme. Les références constantes sont utilisées à bon escient.
302282	1,0	get_data_from_file, image_filtering	Deux de tes fonctions sont trop longues ! Cette règle ne s'applique pas seulement à la fonction main. Sinon, bonne décomposition du code et usage astucieux des passages par référence.
302289	2,0		Bon travail sur la décomposition du code en fonctions. Tu peux passer tes vector par référence plutôt que par valeur pour plus d'efficacité, et éventuellement donner des noms plus explicites à certain paramètres
302297	2,0		Très bonne décomposition ! Continue ainsi.
302327	2,0		Très bonne décomposition, bonne utilisation des références constantes.
302360	2,0		Bonne décomposition du code et bon usage des structures !
302362	2,0		La limite de 40 lignes est bien respectée, mais souvent frôlée, utiliser plus souvent des structs et sous-fonctions pour éviter les longues listes de paramètres et les for-loops trop profonds. Passage par référence à faire plus souvent.
302365	2,0		Penses aux passages paré frérence. La struct est une bonne idee, et la modularisation de insert_couleur est très bien. Une bonne décomposition du code
302376	2,0		Bonne décomposition du code ; bien vu de passer les gros paramètres par référence
302394	2,0		Très bonne décomposition en fonctions, bonne utilisation des constref, bonne utilisation des typedef, structs et constexpr :)

302416	2,0		Ce code est extrêmement bien décomposé ! Bravo !
302508	2,0		Bonne décomposition, cherche à mettre plus de commentaires dans ton code
302520	2,0		Bonne décomposition, bravo!
302539			
302546	2,0		Good code factoring. In this program, in all likelihood, the main is at the highest level of abstraction and should thereby be rather self-explanatory, which should significantly lessen the need for comments. @(119, 123, 133) don't seem essential....
302551	2,0		Modularisation correcte mais certaines fonctions auraient du être décomposées ("recuperation_image_et_seuillage()")
302565	2,0		Code très logique, très bonne décomposition du code (à part le main qui aurait du être utilisé).
302576	2,0		La décomposition de ton code semble assez bien pensée. N'hésite pas à rajouter cependant des commentaires pour donner un indice sur le but et le fonctionnement de divers fonctions, surtout quand le nom n'est pas forcément très explicite.
302583	2,0		Très bonne décomposition. Continue ainsi
302584	2,0		Excellente décomposition : le main est clair et compréhensible. Continue ainsi !
302659	2,0		Rien à redire là-dessus, ton code est bien divisé entre les fonctions et les noms de ces dernières sont bien clairs et explicites.
302674	2,0		Ton code est correctement décomposé et est particulièrement court ! Bravo !
302680	2,0		Très bien ! Bravo ! Tu as bien décomposé ton code en fonctions. A la limite, même trop: il est inutile de faire une fonction pour simplement lire un nombre ou un string.
302699	2,0		Bonne décomposition, bravo!
302703	2,0		Bonne décomposition, bravo!
302747	2,0		Bonne décomposition
302759	2,0		Bonne décomposition du code, classique. Aurait pu être un peu plus modulaire, mais déjà la méthode main est bien divisée en méthodes d'entrée auxiliaires.
302829	2,0		Bonne décomposition, bonne utilisation des arguments par référence, bravo!
302833	2,0		Bonne décomposition, bravo!
302875	2,0		Structure logique. Penser à utiliser des références constantes afin d'optimiser l'éxécution.
302888	2,0		Bonne décomposition bravo
302896	2,0		Structure claire, les fonctions pourraient être plus specialisées.
302909	2,0		Bonne décomposition en fonctions Pense à passer tes gros paramètres par référence (constante ou non) pour éviter qu'ils soient copiés lorsque passés en paramètre
302910	2,0		Bonne décomposition en fonctions et passages par référence Pense à mettre le modificateur const sur les références que tu ne modifies pas dans la fonction
302917	1,0	main	Bonne décomposition mais le main est un peu trop long : essaye de moins espacer les lignes et créer des sous-fonctions s'il le faut (ou regrouper plusieurs fonctions dans une fonction de plus haut niveau).
302953	2,0		Bonne décomposition du code de manière générale, cela permet d'efficacement comprendre comment ton code est conçu. Fais attention cependant à ne pas créer de fonctions superflues : par exemple, saisie_nb_col et saisie_nb_lig sont exactement les mêmes ! Tu pourrais soit les rassembler soit appller à deux reprises une même fonction
303000	2,0		Excellente décomposition ! Cependant les commentaires pourraients être remplacées par des fonctions de plus haut niveau qui regroupent les sous-fonctions de chaque partie et qui rendraient ton main plus court et explicite intrinsèquement.
303003	2,0		Main devrait être plus compact. Par exemple, la lecture d'entrées mérite sa propre fonction. Passer les arguments d'une fonction par valeur peut aussi aider à la performance.

303011	2,0		Bonne décomposition du code Pense à passer tes gros paramètres par référence (constante ou non) pour éviter qu'ils soient copiés lorsque passés en paramètre
303028	2,0		Très bonne décomposition, continue ainsi.
303039	2,0		Excellente décomposition ! Continue ainsi.
303084	2,0		La struct Info est une bonne idée, difficile à justifier par contre. Attention à n'utiliser les double que pour les valeurs à virgule. Essaie de garder une vision globale du code pour voir ce qui peut être regroupé, ce qui est inutile. L'utilisation des reshape est dangereuse car difficile à suivre, tout comme l'image en nbC*PXL.
303114	1,0	main	You did use some functions but you should have gone further It is clear that the main function is way too long. Ideally, it would just reads as a table of contents and be a way to have a quick glimpse at the whole process. The comments seem relevant the way things are now (still in the main function), but you should probably reorganize the code so they become unnecessary through a rather self-explanatory main function. Nice use of const&! :)
303153	2,0		Simplement excellent !
303778	1,0	main	Ta fonction main dépasse légèrement la limite autorisée. Tu pourrais enlever plusieurs sauts à la ligne inutiles et exporter quelques tâches (ex: tests pour les fonctions d'erreurs). Le reste de ta décomposition du code me semble sinon assez bien pensée.
304034	1,0	main	Le principe de décomposition est bien compris mais mal appliqué, essaie de définir des sous-fonctions pour chaque sous problème identifié. De cette manière, le main respecterait la taille imposée.
304148	2,0		
304289			