25-Suivi de la position d'une couleur dans un film

Suivi des couleurs

Il existe de nombreuses façons d'analyser le contenu d'une matrice. Dans ce chapitre du didacticiel, nous présentons un moyen très simple d'examiner le contenu en couleurs d'une image. Nous alons étudier le problème de la recherche d'une couleur particulière (ou d'une gamme de couleurs) d'une image, puis le suivi de cette couleur lorsque sa position change d'une image vidéo à l'autre. Cette technique est utile pour obtenir des informations sur le mouvement d'un objet particulier dans une vidéo ou pour suivre un geste physique. De manière plus générale, cette technique est utile pour trouver l'emplacement d'une valeur numérique particulière (ou d'une plage de valeurs) dans une matrice de données.

jit.findbounds

L'objet que nous allons utiliser pour trouver une couleur particulière dans une image s'appelle *jit.findbounds*. Comme nous suivons la couleur dans une vidéo, nous analyserons - comme vous pouvez vous y attendre - une matrice bidimensionnelle à 4 plans de données ***char***, mais vous pouvez utiliser *jit.findbounds* pour des matrices de n'importe quel type de données et de n'importe quel nombre de plans.

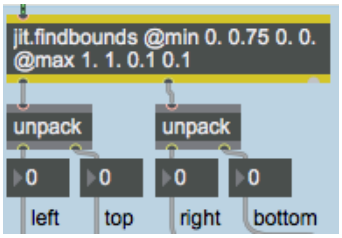
Voici comment fonctionne *jit.findbounds*. Vous spécifiez une valeur minimale et une valeur maximale que vous souhaitez rechercher dans chaque plan, en utilisant les attributs **min** et **max** de *jit.findbounds*. Lorsque *jit.findbounds* reçoit une matrice, il recherche dans l'ensemble de la matrice les valeurs qui entrent dans la plage que vous avez spécifiée pour chaque plan. Il envoie les index de cellule qui décrivent la région où il a trouvé les valeurs désignées. En effet, il envoie les index de la région ***limitrophe*** dans laquelle les valeurs apparaissent. Dans le cas d'une matrice 2D, la région limitrophe sera un rectangle, donc *jit.findbounds* enverra les index pour les cellules de gauche-haut, et bas-droit de la région dans laquelle il a trouvé les valeurs spécifiées.

Dans cet exemple, nous utilisons l'objet *jit.movi*e pour lire un film (en fait une animation) d'une boule rouge qui se déplace. Il s'agit évidemment d'une situation plus simple que celle que vous trouverez dans la plupart des vidéos, mais elle nous donne un cadre clair dans lequel nous pouvons voir comment *jit.findbounds* fonctionne. Remarquez que nous avons utilisé des arguments saisis pour initialiser les attributs **min** et **max** de *jit.findbounds*.

Valeurs minimales et maximales spécifiées pour chacun des quatre plans

Il y a quatre arguments pour ces attributs: une valeur pour chacun des quatre plans de la matrice que *jit.findbounds* recevra. L'attribut **min** définit la valeur minimale acceptable pour chaque plan et l'attribut **max** définit la valeur maximale acceptable. Ces arguments amènent *jit.findbounds* à rechercher toute valeur comprise entre 0 et 1 dans le plan alpha, toute valeur comprise entre 0,75 et 1 dans le plan rouge et toute valeur comprise entre 0 et 0,1 dans les plans vert et bleu. Comme les données dans la matrice seront de type ***char***, nous devons spécifier les valeurs que nous voulons rechercher en termes de nombre décimal compris entre 0 et 1. (Voir les ***tutoriels 5 et 6*** pour savoir comment les valeurs ***char*** sont utilisées pour représenter couleurs.) Nous voulons suivre l'emplacement d'une balle rouge, nous demandons donc à *jit.findbounds* de rechercher les cellules qui contiennent des valeurs très élevées dans le plan rouge et des valeurs très faibles dans les plans vert et bleu. (Nous accepterons n'importe quelle valeur du plan alpha.)

• Cliquez sur le *toggle* pour lancer le *metro*. Au fur et à mesure que la balle rouge se déplace, *jit.findbounds* indique les index de cellule du rectangle de délimitation de la boule. Arrêtez le *metro* et examinez les chiffres sortis de *jit.findbounds*. Vous verrez quelque chose comme ceci:

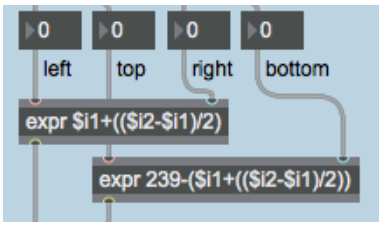


jit.findbounds rapporte la région où la couleur spécifiée apparaît

L'objet *jit.findbounds* signalera la région où il trouve les valeurs souhaitées dans tous les plans de la même cellule. Dans cette image, l'objet *jit.findbounds* a trouvé les valeurs que nous avons demandées quelque part dans les colonnes 120 à 159 et quelque part dans les lignes 50 à 89 incluses. Cela est logique, puisque la boule rouge a un diamètre d'exactement 40 pixels. Ces index de cellules décrivent la région carrée 40x40 cellules où se trouve la balle dans cette image particulière de la vidéo.

Notez que la sortie de *jit.findbounds* de ses deux premières sorties se fait sous la forme de deux listes. La première sortie indique la cellule de départ où les valeurs ont été trouvées dans chaque dimension et la deuxième sortie indique la cellule de fin de la région dans chaque dimension. (Comme il s'agit d'une matrice 2D, il n'y a que deux valeurs dans chaque liste et nous utilisons les objets *unpack* pour les visualiser individuellement.)

Si nous voulions connaître ***un point unique*** qui décrit l'emplacement de la balle dans l'image vidéo, nous pourrions prendre le point central de cette région rectangulaire rapportée par *jit.findbounds* et l'appeler l'emplacement de la balle. C'est ce que nous faisons avec les objets *expr*. Pour chaque dimension, nous prenons la différence entre la cellule de départ et la cellule d'arrivée, nous la divisons en deux pour trouver le centre entre les deux, puis nous l'ajoutons à l'index de la cellule de départ pour obtenir notre point de localisation unique.

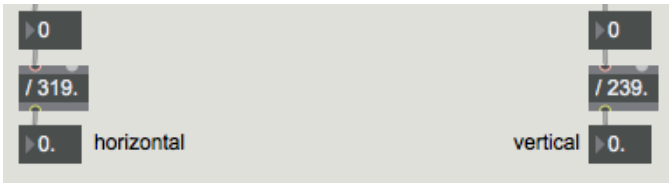


Calcul du point central du rectangle

Notez que pour la dimension verticale, nous soustrayons en fait la coordonnée d'emplacement verticale de 239. C'est s'explique par le fait que les index de cellule vont de haut en bas, mais nous préférons penser que la hauteur de l'objet va de bas en haut. (c'est également le comportement de l'objet *slider,* donc puisque nous allons afficher la coordonnée verticale avec le *slider*, nous devons exprimer la coordonnée comme augmentant de bas en haut.)

Nous envoyons les résultats de notre calcul de localisation à une paire d'objets *sliders* pour démontrer que nous suivons avec succès le centre de la balle, et nous affichons les coordonnées dans les boîtes de *nombres*. Nous mettons également les coordonnées à l'échelle dans la plage de 0 à 1, pour montrer avec quelle facilité l'emplacement horizontal et vertical de la balle pourrait être utilisé pour modifier une activité ou un attribut ailleurs dans un patch Max. Par exemple, nous pourrions utiliser l'emplacement vertical pour contrôler le volume d'une vidéo ou d'un son MSP, ou

nous pourrions utiliser la coordonnée horizontale pour affecter la rotation d'une image.



Mettez à l'échelle les coordonnées d'emplacement dans la plage 0-1, pour les utiliser ailleurs dans le patch.

Suivi d'une couleur dans une image complexe

Eh bien, tout cela a bien fonctionné pour l'exemple simple d'une balle rouge sur fond blanc. Mais le suivi d'un objet unique dans une vidéo réelle est beaucoup plus difficile. Nous allons vous montrer certains des problèmes que vous pouvez rencontrer et quelques astuces pour les résoudre.

• Assurez-vous que le film ***Redball*** est arrêté. Double-cliquez maintenant sur l'objet *patcher* **bballtracking** pour afficher un ***exemple plus détaillé***. Cliquez sur le *toggle* intitulé ***Start***/***Stop*** dans le coin supérieur gauche du sub-patch [***bballtracking***] pour lancer la vidéo.

Ce film présente des objets aux couleurs distinctes: une chemise rouge, un pantalon vert et une balle jaune et bleue. Potentiellement, cela pourrait être utile pour le suivi des couleurs. Cependant, il y a quelques facteurs qui rendent le suivi de cette balle un peu plus compliqué que dans l'exemple précédent.

Tout d'abord, les quelques lignes de balayage supérieures de la vidéo (les quelques lignes supérieures de la matrice) contiennent des déchets que nous ne voulons vraiment pas analyser. Ces déchets sont un artefact malheureux de la numérisation imparfaite de cette vidéo particulière. De telles imperfections sont courantes et peuvent compliquer le processus d'analyse. Deuxièmement, l'image n'est pas très saturée en couleurs, de sorte que les différentes couleurs ne sont pas aussi distinctes que nous le souhaiterions. Enfin, la balle sort entièrement du cadre à la fin du clip de quatre secondes. (Lorsque *jit.findbounds* ne trouve aucune instance des valeurs recherchées, il signale des index de cellule de début et de fin de **-1**.) Quatrièmement, si nous voulons suivre la couleur jaune pour trouver l'emplacement de la balle dans l'image, nous devons reconnaître que la balle n'est pas toute d'une seule nuance de jaune. En raison de la texture de la balle et de l'éclairage, elle apparaît en fait comme une gamme de jaunes, et nous devrons donc identifier soigneusement cette gamme dans *jit.findbounds*.

Essayons de résoudre certains de ces problèmes. Comme nous l'avons montré dans le ***didacticiel 14***, certains objets Jitter nous permettent de désigner un rectangle source d'une image que nous voulons visualiser, et qui est différent de la matrice complète. Dans le ***didacticiel 14***, nous avons montré les attributs **srcdimstart**, **srcdimend** et **usesrcrect** de *jit.matrix*, et nous avons mentionné que *jit.movie* possède des attributs comparables, appelés **srcrect** et **usesrcrect**. Utilisons ces attributs de *jit.movie* pour recadrer l'image vidéo, en se débarrassant de certaines parties que nous ne voulons pas voir.

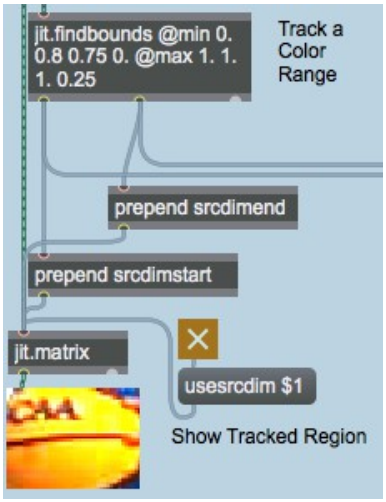
• Cliquez sur la boîte de *message* intitulée ***Crop Source Image***. Cela envoie à *jit.movie* les index de cellule d'un nouveau rectangle source que nous voulons visualiser et indique à *jit.movie* d'utiliser ce rectangle source au lieu de la matrice complète. Reamrquez qu'en commençant à la ligne **4** (c'est-à-dire en commençant par la cinquième ligne de la matrice), nous élimons les déchets en haut de l'image. Nous supprimons également 20 pixels du côté gauche de l'image source, afin que le premier rebond de la balle se produise exactement dans le coin inférieur gauche. Maintenant nous sommes concentrés sur la partie de la vidéo que nous souhaitons analyser.

• Nous allons maintenant nous occuper de nos autres problèmes. Cliquez sur le petit objet *preset* intitulé ***Setup*** dans le coin inférieur droit de la fenêtre. Cela règle tous les objets de l'interface utilisateur sur les paramètres souhaités.

Il définit l'attribut **loop** de *jit.movie* sur 2 pour la lecture en va-et-vient, et il définit un point final de boucle au temps **2160** (juste au moment où la 54ème image se produirait), de sorte que le film est maintenant lu en va-et-vientde l' image 0 à l'image 53 et inversement. Le film est maintenant joué jusqu'au moment du premier rebond de la balle sur le trottoir, puis change de direction.

Nous avons également envoyé certaines valeurs à l'objet *jit.brcos*a (abordé en détail dans le ***didacticiel 7***) pour définir ses attributs **brightness**, **contrast** et **saturation** comme nous le souhaitons. Cela ne donne pas exactement la meilleure image possible, mais cela rend les différentes couleurs plus distinctives et les compresse dans une petite plage de valeurs, ce qui les rend plus faciles à suivre pour *jit.findbounds*.

Et nous avons activé l'attribut **usesrcdim** de l'objet *jit.matrix* (au centre du patch) afin qu'il utilise maintenant la sortie de *jit.findbounds* pour déterminer son rectangle source. Vous pouvez voir la région suivie affichée dans la fenêtre *jit.p* étiquetée ***Show Tracked Region.***



Utilisation de la sortie de *jit.findbounds* pour définir les attributs **srcdimstart** et **srcdimend** de *jit.matrix*.

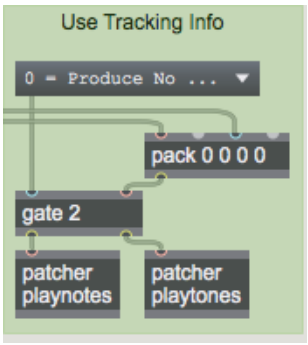
Le jaune de base de la balle contient des quantités presque égales de rouge et de vert. Nous avons donc défini les attributs **min** et **max** de *jit.findbounds* pour rechercher les cellules contenant des valeurs élevées dans les plans rouge et vert et une valeur faible dans le plan bleu. Vous pouvez voir qu'avec des paramètres minutieux de *jit.brcosa* et des attributs **min** et **max** de *jit.findbounds*, nous avons réussi à obtenir un suivi très fiable de la partie jaune de la balle.

> Remarque: un détail assez important que nous n'avons pas vraiment abordé ici est la façon de définir le plus efficacement possible les attributs **min** et **max** de *jit.findbounds* pour suivre une couleur particulière dans une vidéo. Un certain nombre d'ajustements par essais et erreurs sont nécessaires, mais vous pouvez obtenir des informations très spécifiques sur la couleur d'un pixel particulier en utilisant l'objet *suckah* présenté dans le ***Didacticiel 10***. Vous pouvez placer l'objet *suckah* sur la *jit.pwindow* de la vidéo que vous voulez suivre, et utilisez la sortie de *suckah* pour obtenir les informations RVB de cette cellule. (Les valeurs de *suckah* sont dans la plage 0-255, mais vous pouvez les diviser par 255,0 pour les amener dans la plage 0-1.)

Utiliser l'emplacement d'un objet

Ainsi, au moins dans cette situation particulière, nous avons réussi à surmonter les difficultés liées au suivi d'un seul objet dans une vidéo. Mais maintenant que nous avons réussi, qu'allons-nous faire des informations que nous avons obtenues? Nous allons montrer deux façons d'utiliser la position d'un objet pour contrôler le son: en jouant des notes MIDI ou en jouant des sons MSP. Les deux exemples ne sont pas très sophistiqués du point de vue musical, mais ils devraient servir à illustrer le problème fondamental de la mise en correspondance des informations de localisation avec les informations sonores.

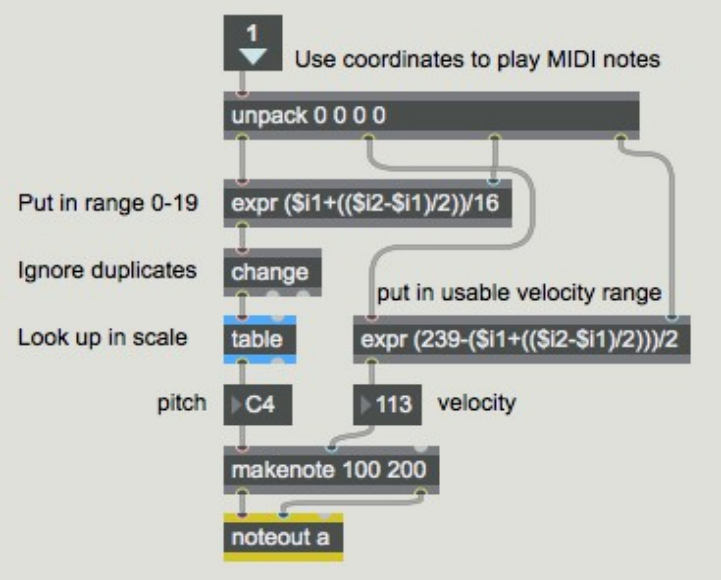
Nous enverrons les données de localisation à deux sub-patchs situés dans la partie du patcher intitulée ***Use Tracking Info***. Nous utilisons un objet *pack* pour regrouper tous les résultats de *jit.findbounds* dans une liste unique de 4 éléments, puis unous utilisons un objet *gate* pour acheminer ces informations vers le sub-patch **playnotes** (pour jouer des notes MIDI) ou le subpatch **playtones** du patcher (pour jouer des sons MSP) ou aucun (pour ne produire aucun son).



Envoyer les informations de localisation à l'un des deux sub-patchs

Jouer des notes

• Dans le *umenu* intitulé **Use Tracking Info**, choisissez l'option de menu **1 = Play MIDI Notes**. Double-cliquez sur l'objet **playnotes** du patcheur pour voir le contenu du sub-patch [**playnotes**]. Si vous n'entendez aucune note jouée (et que vous avez vérifié que le film est toujours en cours de lecture), essayez de double-cliquer sur l'objet *noteout* et de choisir un synthétiseur MIDI différent dans la boîte de dialogue du périphérique.



Le contenu du sub-patch [**playnotes**]

Dans le sub-patch [**playnotes**], nous utilisons le même type de formules de mappage que dans le premier exemple pour calculer les coordonnées de la balle et placer les informations dans une plage utilisable. Nous calculons l'emplacement horizontal et le divisons par 16 pour obtenir des nombres qui seront potentiellement compris entre 0 et 19. Nous utilisons l'objet *change* pour ignorer les nombres en double (c.-à-d. les notes répétées), puis nous recherchons la note que nous voulons jouer dans la table.

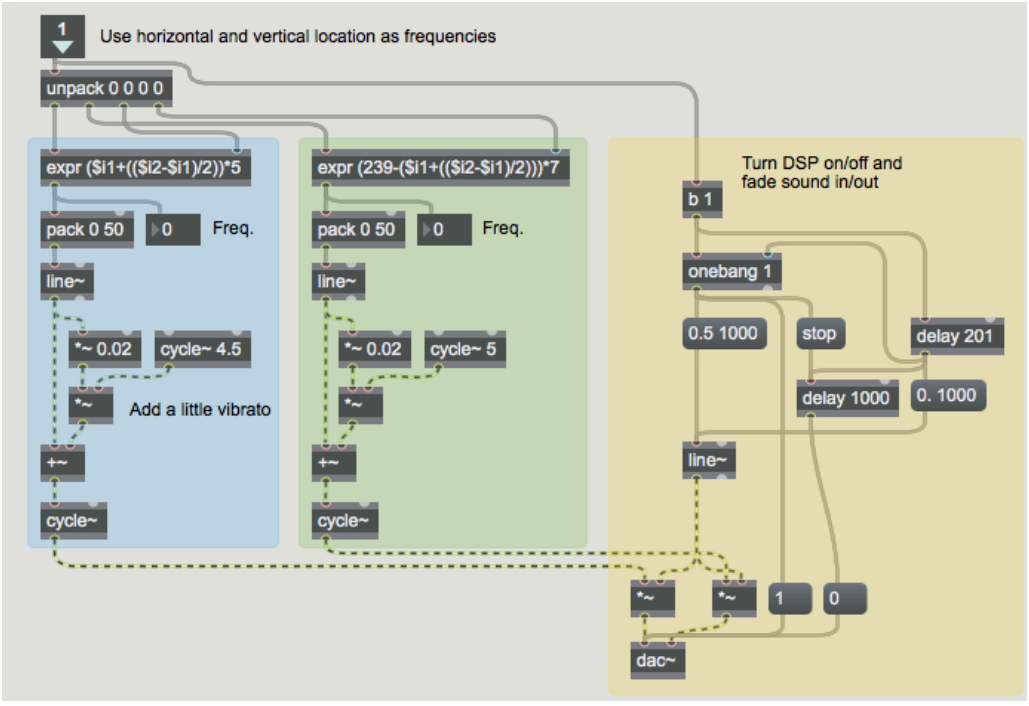
> Remarque: le mouvement du basketteur n'a aucune relation avec une échelle musicale particulière. Par conséquent, le fait de saisir les données brutes de localisation sous forme de nombres MIDI résulterait en une improvisation atonale. (Si nous voulons donner une implication tonale aux choix de hauteur, nous pouvons utiliser les nombres générés par le mouvement horizontal du ballon comme des numéros d'index pour rechercher les notes de la gamme dans une table de consultation. Si vous voulez voir (ou même modifier) le contenu d e *table*, double-cliquez simplement sur l'objet *table* pour ouvrir sa fenêtre d'édition graphique.

Nous utilisons l'emplacement vertical de la balle (que nous avons mappé dans la plage 0-119) pour déterminer les valeurs de vélocité. L'objet *makenote* assigne la durée (200 ms) aux notes et se charge de fournir les messages MIDI *note-off*. La pulsation sous-jacente de la musique (20 pulsations par seconde) est déterminée par la vitesse du *metro* qui lit le film, mais comme l'objet *change* supprime les notes répétées, chaque pulsation n'est pas itérée comme une note MIDI.

• Fermez la fenêtre de sub-patch [**playnotes**]. Cliquez sur la boîte de message intitulée **Crop and Flip source image**. Cela envoie un nouveau rectangle source à *jit.movie* pour retourner l'image horizontalement, ce qui inverse l'effet musical haut-bas du sub-patch [**playnotes**].

Jouer des notes

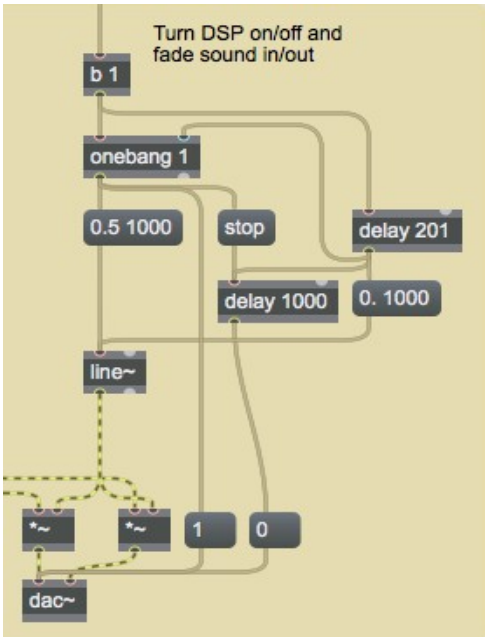
• Dans l'*umenu* intitulé **Use Tracking Info**, choisissez l'élément de menu **2 = Play MSP Tones**. Double-cliquez sur l'objet **playtones** du *patcheur* pour afficher le contenu du sub-patch [**playtones**].



Utiliser l'emplacement d'une couleur comme information de contrôle de fréquence pour les oscillateurs MSP

Nous utilisons ici les coordonnées de l'emplacement horizontal et vertical du ballon de basket comme valeurs de fréquence pour les oscillateurs MSP. Les équations que nous utilisons pour calculer ces valeurs sont quelque peu arbitraires, mais elles ont été conçues de manière à faire correspondre les deux coordonnées à des plages de fréquences similaires. La coordonnée horizontale est utilisée pour contrôler l'oscillateur du canal audio gauche, et la coordonnée verticale contrôle la fréquence de l'oscillateur dans le canal droit.

Nous utilisons la présence de messages entrants pour activer l'audio MSP (et augmenter le son), et si les messages sont absents pendant plus de 200 ms, nous atténuons le son et désactivons l'audio.



Le premier message commence et passe en fondu audio; l'absence de message pendant 201ms estompe et éteint l'audio.

• Fermez la fenêtre de sub-patch [**Playtones**]. Faites pivoter l'image vidéo horizontalement en cliquant sur les boîtes de *message* intitulées **Crop source image** et **Crop and Flip source image** pour entendre la différence d'effet sur les oscillateurs MSP.

Faire dériver plus d'informations

Dans ce didacticiel, nous avons montré une implémentation assez simple dans laquelle nous utilisons directement les coordonnées d'une région de couleur pour contrôler les paramètres de synthèse sonore ou de la performances MIDI. Avec un peu de programmation Max supplémentaire, nous pourrions potentiellement dériver d'autres informations supplémentaires sur le mouvement d'un objet.

Par exemple, en comparant l'emplacement d'un objet dans une image vidéo avec son emplacement dans l'image précédente, nous pourrions utiliser le théorème de Pythagore pour calculer la distance parcourue par l'objet d'une image à l'autre, et calculer ainsi sa vitesse. Nous pouvons également calculer la pente de son mouvement (), et ainsi (avec la fonction trigonométrique arctangente) déterminer son angle de mouvement. En comparant une valeur de vitesse à la précédente, on peut calculer l'accélération, et ainsi de suite. En comparant la taille apparente d'un objet d'une image à l'autre, nous pouvons même faire quelques suppositions grossières sur son déplacement vers ou depuis la caméra dans l'axe "z" (profondeur).

Sommaire

L'objet *jit.findbounds* détecte les valeurs comprises dans une certaine fourchette dans chaque plan d'une matrice, et il rapporte la région de la matrice où il trouve des valeurs situées dans la fourchette spécifiée de chaque plan. Ceci est utile pour trouver l'emplacement de toute plage de données numériques dans tout type de matrice. En particulier, il peut être utilisé pour trouver l'emplacement d'une couleur particulière dans une matrice à 4 plans, et peut donc être utilisé pour suivre le mouvement d'un objet dans une vidéo.

Le recadrage de l'image vidéo avec l'attribut **srcrect** de *jit.movie* permet de se concentrer sur la partie souhaitée de l'image source. L'objet *jit.brcosa* est utile pour ajuster les valeurs de couleur dans la vidéo source, ce qui permet d'isoler et de détecter plus facilement une couleur ou une gamme de couleurs spécifiques.

Nous pouvons utiliser la sortie de *jit.findbounds* pour suivre l'emplacement d'un objet, et à partir de là, nous pouvons calculer d'autres informations sur le mouvement de l'objet, comme sa vitesse, sa direction, etc. Nous pouvons utiliser les informations dérivées pour contrôler les paramètres de performance MIDI, de sons de synthèse MSP ou d'autres objets de Jitter.