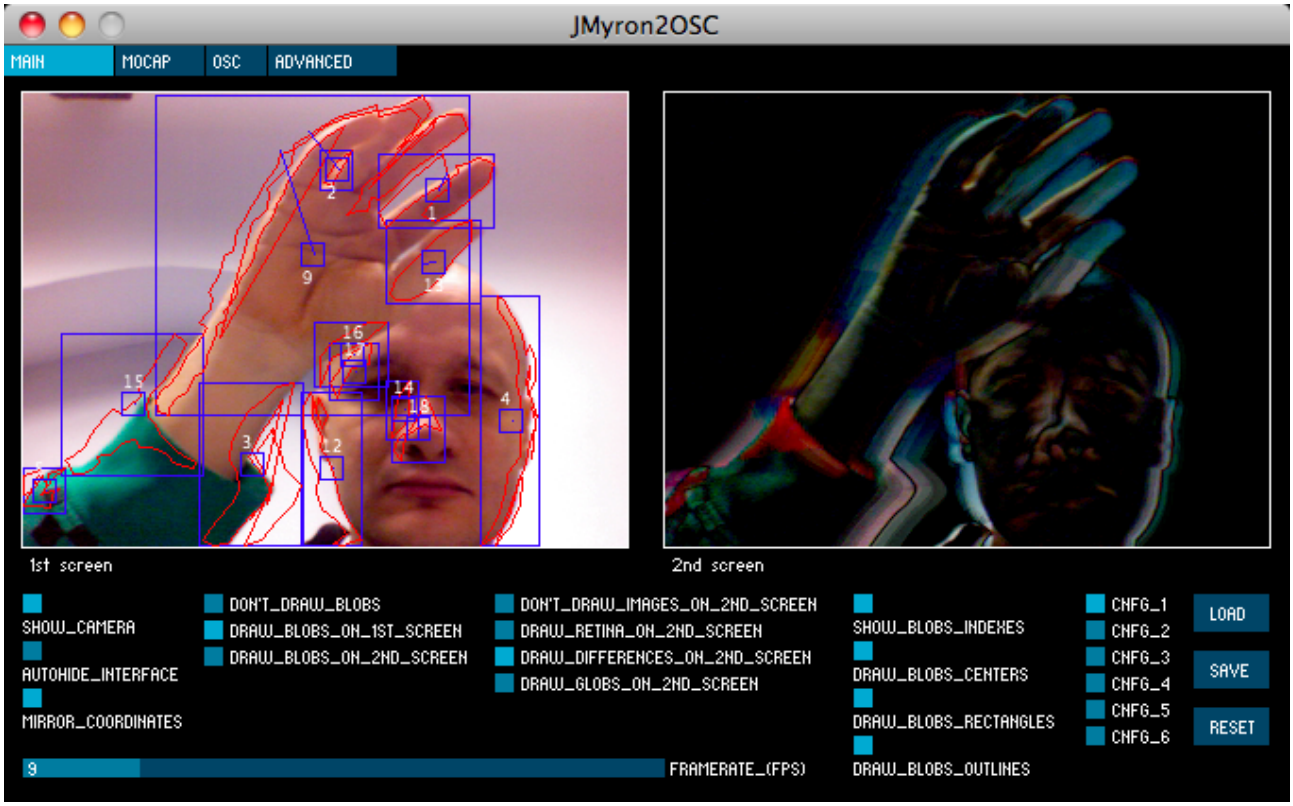
JMyron2OSC

wprowadzenie

Paweł Janicki
http://paweljanicki.jp
pawel@wrocenter.pl



JMyron2OSC to relatywnie prosta w obsłudze, ale zarazem funkcjonalna i dysponująca szeregiem zaawansowanych funkcji aplikacja, pozwalająca na wykrywanie ruchomych obiektów w polu widzenia kamery wideo i transmitująca dane za pomocą protokołu OSC (*OpenSound Control*). Historia aplikacji wiąże się silnie z Interaktywnym Placem Zabaw. Interaktywny Plac Zabaw (IPZ) to wystawa dla najmłodszej publiczności, złożona z interaktywnych instalacji medialnych, stworzona przez Patrycję Mastej, Dominikę Sobolewską i Pawła Janickiego, a wyprodukowana przez Centrum Sztuki WRO (http://www.wrocenter.pl). Podczas pracy nad instalacjami i obiektami wchodzącymi w skład wystawy szybko okazało się, że – chociaż istnieje wiele komercyjnych i niekomercyjnych narzędzi realizujących funkcje podobne do **JMyron2OSC** – to brakuje prostej i szybkiej w obsłudze aplikacji, której można łatwo nauczyć osoby nie posiadające doświadczenia w klasycznym programowaniu lub np. środowiskach programowania wizualnego (w rodzaju MaxMSP/Jitter, Pure-Data, itp.).

JMyron2OSC w zamierzeniach miał być taką właśnie aplikacją, choć ewoluując stał się pełnowartościowym narzędziem przydatnym w kreowaniu prac interaktywnych, bazujących na kamerze jako interfejsie (prostszy, łatwiejszy do zastosowania w typowych projektach system śledzenia ruchu realizuje równolegle rozwijana aplikacja **ActiveZones2OSC** – interfejsy użytkownika i struktura obu aplikacji są podobne, natomiast **Jmyron2OSC** korzysta z bardziej zaawansowanego konceptu analizy ruchu: zamiast aktywnych stref definiowanych w **ActiveZones2OSC** stara się ona wydzielić i śledzić parametry jednego lub wielu ruchomych obiektów). Autorem **ActiveZones2OSC** i **JMyron2OSC** jest **Paweł Janicki** (http://paweljanicki.jp).

Licencja

Autor udziela licencji na korzystanie z aplikacji **JMyron2OSC** do niekomercyjnego użytku prywatnego. Wykorzystanie komercyjne i/lub wiążące się z publicznym prezentacjami lub publicznym funkcjonowaniem realizacji opartych o **JMyron2OSC** wymaga pisemnej zgody autora. Aplikacja **JMyron2OSC** nie może być redystrybuowana na nośnikach fizycznych lub elektronicznie bez zgody autora.

Autor dołożył wszelkich starań by program **JMyron2OSC** pracował stabilnie i wydajnie, jednak nie ponosi odpowiedzialności, za ewentualne nieprawidłowe funkcjonowanie ani za ewentualne szkody wynikające z prawidłowej lub nieprawidłowej pracy aplikacji **JMyron2OSC**.

Instalacja i wymagania systemowe

Program **JMyron2OSC** został napisany w środowisku programistycznym Processing (http://processing.org) i korzysta intensywnie z wielu możliwości i komponentów napisanych dla tego środowiska – przede wszystkim z biblioteki *JMyron (WebCamXtra)* autorstwa Josha Nimoy'a (w przeciwieństwie do bliźniaczej aplikacji **ActiveZones2OSC**, która wykorzystuje *JMyron [WebCamXtra]* jedynie do pobierania danych z kamery i detekcji tła, więcej informacji o *JMyron [WebCamXtra]* znaleźć można na stronie http://webcamxtra.sourceforge.net/).

Aplikacja działa na Mac OSX (wymaga maka z procesorem Intel) i MS Windows. Warunkiem uruchomienia programu jest jednak obecność w systemie maszyny wirtualnej Javy – użytkownicy komputerów Apple są tu w lepszej sytuacji, ponieważ maszyna wirtualna Javy jest standardowo zainstalowanym komponentem systemu operacyjnego (jakkolwiek ma się to zmienić w przyszłości) natomiast osoby korzystające z Windows muszą odpowiednią dla swojej wersji systemu operacyjnego wersję JRE (*Java Runtime Environment*) zainstalować samodzielnie (środowisko Java jest bezpłatne), najlepiej korzystając ze strony http://java.com.

JMyron2OSC do prawidłowej pracy wymaga oczywiście podłączonej do komputera kamery. Aplikacja radzi sobie z kamerami USB (w tym PS3 Eye – oczywiście w tym przypadku należy zainstalować odpowiednie dla naszego systemu operacyjnego drivery; dla OSC będzie to *Macam*: http://webcam-osx.sourceforge.net; dla systemów z rodziny Windows: *CL Eye Platform* Driver: http://codelaboratories.com) i FireWire oraz digitizerami obrazu korzystającymi z tych złączy. Kamera lub digitizer muszą być podłączone przed uruchomieniem aplikacji (w przeciwnym razie program nie wykryje urządzenia).

JMyron2OSC korzysta z pierwszego w kolejności urządzenia przechwytywania wideo (jeśli więc mamy w systemie więcej takich urządzeń, dobrze jest wyłączyć te, z których nie zamierzamy korzystać). Program wykrywa również kamery wbudowane w laptopy. Sporadycznie pojawiają się jedynie problemy w wykorzystaniu zewnętrznych kamer USB podłączonych do komputerów Apple. Jeśli uruchomiliśmy aplikację zapominając wpiąć uprzednio kamerę – wystarczy zamknąć program, wpiąć (i ewentualnie włączyć) kamerę i uruchomić program ponownie (zwykle należy odczekać też kilka-kilkanaście sekund aby system operacyjny wykrył urządzenie).

Poza wymienionymi komponentami **JMyron2OSC** nie wymaga żadnych specjalnych zabiegów – wystarczy skopiować folder z aplikacją w wybrane miejsce na dysku. Dwukrotne kliknięcie na ikonie pliku „JMyron2OSC” uruchomi program.

Kamera jako sensor ruchu

Nawet prosta kamera sprzężona z odpowiednim oprogramowaniem może stać się wydajnym sensorem ruchu – tego rodzaju technologie często bywają wykorzystywane w instalacjach alarmowych, mogą też przydać się w wielu innych dziedzinach. Warto tu nadmienić, że zastosowania okołoartystyczne – m.in. za sprawą pionierskich prac amerykańskiego twórcy Myrona W. Kruegera – pojawiły się bardzo wcześnie.

Zakres analizy obrazu może obejmować wykrywanie naruszenia zdefiniowanych stref (jak w przypadku **ActiveZones2OSC**), śledzenie pojedynczego lub wielu obiektów jednocześnie (właśnie tego rodzaju *tracking* uzupełniony o szereg dodatkowych opcji realizuje **JMyron2OSC**), analizę kształtu, wykrywanych znaczników (w tym systemy w rodzaju Reactivision [http://reactivision.sourceforge.net] lub ARToolKit [http://www.hitl.washington.edu/artoolkit/] oraz niektóre kontrolery gier), itp. Ponadto poza oczywistym zakresem światła widzialnego można budować systemy śledzenia ruchu wykorzystujące kamery podczerwieni lub termiczne (ma to znaczenie, jeśli np. chcemy uniezależnić działanie naszego systemu od przypadkowych zmian oświetlenia).

Zasadniczo każdy system śledzenia ruchu wykorzystujący kamerę działa porównując ze sobą kolejne klatki obrazu: przyjmując, że kamera jest nieruchoma (oraz nie manipulujemy ustawieniami obiektywu i zakładając stałe parametry oświetlenia w paśmie widzianym przez kamerę) różnice w kolejnych klatkach obrazu wynikają z obecności w kadrze ruchomych obiektów. Idąc tym tropem dość łatwo jest budować i implementować (lub przynajmniej wyobrazić sobie taką możliwość) różne algorytmy analizujące – biorąc pod uwagę wydajność współczesnych komputerów może się to odbywać w czasie rzeczywistym – zdigitalizowany obraz pod kątem wykrywania i śledzenia zmian wskazujących na obecność „ruchu” w kadrze (zaawansowane systemy śledzenia ruchu potrafią w tej chwili znaczenie więcej, łącznie z analizą geometrii obrazu i rekonstrukcją przestrzeni 3D).

Co potrafi JMyron2OSC?

Aplikacja analizuje obraz pobierany w czasie rzeczywistym przez urządzenie wideo (kamera, digitizer) starając się wyłowić z niego informacje o ewentualnych ruchomych obiektach. Na informacje składają się dane o położeniu obiektu (koordynaty centrum i opisujący obiekt prostokąt) oraz jego kształcie, wektor ruchu, kolor centrum obiektu. Dane te są wizualizowane wewnątrz interfejsu **JMyron2OSC**, ale też – a w zasadzie: przede wszystkim – mogą być przesyłane do zewnętrznych aplikacji i urządzeń za pomocą protokołu komunikacyjnego OSC (*OpenSound* Control, więcej o OSC na stronie: http://opensoundcontrol.org).

Tym samym informacje uzyskane z **JMyron2OSC** można wykorzystać w praktycznie dowolnym celu, sprzęgając program z większością środowisk programistycznych, aplikacji i urządzeń stosowanych podczas konstruowania struktur interaktywnych. Aplikacja dysponuje dwoma głównymi, przełączanymi trybami emisji komunikatów OSC w zależności od potrzeb dzieląc uzyskane dane na niewielką ilość długich komunikatów OSC o zmiennej długości lub dłuższą serię krótkich komunikatów o stałej długości – ma to praktyczne znaczenie w przypadku sprzęgania **JMyron2OSC** z wizualnymi językami programowania (MaxMSP. PureData, itp.), ponieważ wielu użytkownikom tych ostatnich trudno jest przetwarzać komunikaty OSC o zmiennej długości bez sięgania po skrypty lub inne techniki wiążące się z głębszą znajomością tych środowisk. **JMyron2OSC** dysponuje też trzecim trybem transmisji komunikatów OSC (*PBL mode*) stanowiącym pozostałość po wczesnych wersjach instalacji

wchodzących w skład IPZ (patrz wstęp) – jakkolwiek jest on pozostawiony głównie dla zgodności z poprzednimi wersjami, to w niektórych przypadkach może okazać się przydatny. W trybie tym **JMyron2OSC** transmituje wszystkie informacje o aktualnym stanie monitorowanej przestrzeni jako jeden duży komunikat OSC. Więcej informacji o zawartości i strukturze komunikatów OSC wysyłanych przez aplikację znajduje się w dalszej części tego dokumentu.

Oczywiście **JMyron2OSC** posiada szereg funkcji umożliwiających dostrojenie działania programu do wymagań użytkownika. Wśród globalnych parametrów i opcji warto wymienić możliwość ustawienia progu zadziałania sensora ruchu (technicznie rzecz biorąc jest to wartość różnicy w poziomie jasności piksela obrazu tła i piksela [o tych samych współrzędnych] bieżącej klatki obrazu), częstotliwości odświeżania obrazu, szeregu parametrów protokołu OSC (w tym możliwość detekcji i transmisji tylko wybranych parametrów systemu śledzenia ruchu oraz przeskalowywania wybranych parametrów), itp.

Motion tracking **w JMyron2OSC**

Jak wspomniałem wyżej, systemy śledzenia ruchu wykorzystujące kamerę jako sensor (pomimo, że działają w oparciu o, w ogólności, jedną i tę samą koncepcję) znaczenie różnią się od siebie sposobami, w jakie w praktyce wcielają w życie ideę leżącą u podstaw ich konstrukcji. Tym samym w zależności od zastosowanego systemu śledzenia ruchu monitorowana przestrzeń i zachodzące w niej wydarzenia są interpretowane na rozmaite sposoby.

JMyron2OSC analizuje napływające klatki obrazu, starając się wyłowić z nich informacje o poruszających się w kadrze obiektach. Aplikacja potrafi identyfikować wiele ruchomych obiektów jednocześnie, przypisując każdemu z nich unikalny index oraz wyłuskać ze strumienia wideo następujące parametry każdego obiektu:

- pozycję centrum obiektu
- pozycję i wielkość prostokąta, w którym zawiera się obiekt
- ilość i koordynaty punktów obrysu obiektu
- index (unikalny numer reprezentujący ruchomy obiekt – index pozwala śledzić wiele poruszających się jednocześnie obiektów identyfikując każdy)
- kolor centrum
- koordynaty miejsca w którym pojawił się ruchomy obiekt o danym indeksie
- wektor ruchu (wyliczany w stosunku do poprzedniej klatki obrazu)

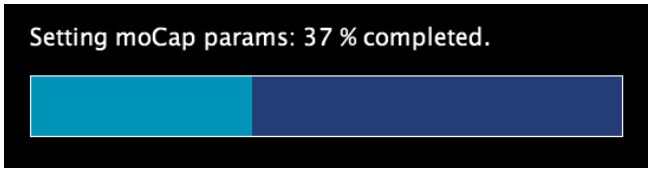
Poza dla każdej klatki obrazu generowany jest zestaw informacji globalnych i statystycznych obejmujących m.in.:

- ilość ruchomych obiektów na scenie (w aktualnej klatce analizowanego obrazu)
- maksymalna ilość punktów obrysu obiektu
- parametry strumienia wideo (rozdzielczość, *fps*)

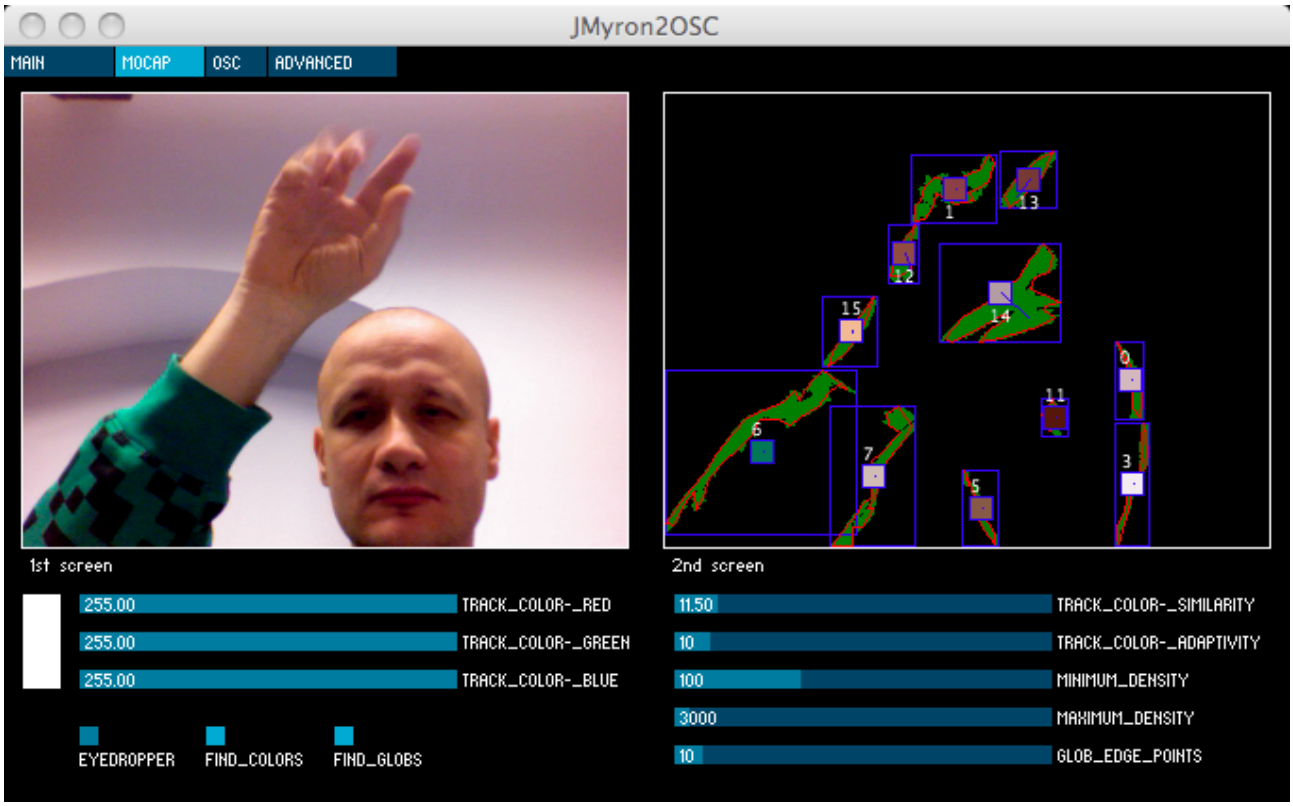
- prędkość procesowania (*fps*)
- ilość i indeksy nowych ruchomych obiektów
- ilość i indeksy obiektów, które zniknęły z kadru
- ilość i indeksy obiektów, które kontynuują swój ruch

Parametry dotyczące pozycji obiektów są normalizowane do zakresu pomiędzy „0” (zero) i „1”. Szczegółowy opis wszystkich parametrów znajduje się w części poświęconej komunikacji OSC.

Interfejs i użytkowanie



Po kliknięciu w ikonę aplikacji **JMyron2OSC** automatycznie wykryje urządzenie przechwytywania wideo (kamerę lub digitizer) i wyświetli (przez około 10 sekund) informację o zbieraniu danych dla systemu śledzenia ruchu – w tym czasie aplikacja analizuje dane z kamery starając się wygenerować obraz tła, czyli monitorowanej przez kamerę przestrzeni w momencie, kiedy nie ma tam ruchomych obiektów (dobrze jest w tym czasie nie przemieszczać kamery, nie zmieniać warunków oświetleniowych i pozostawić monitorowaną przestrzeń wolną od ruchomych obiektów).

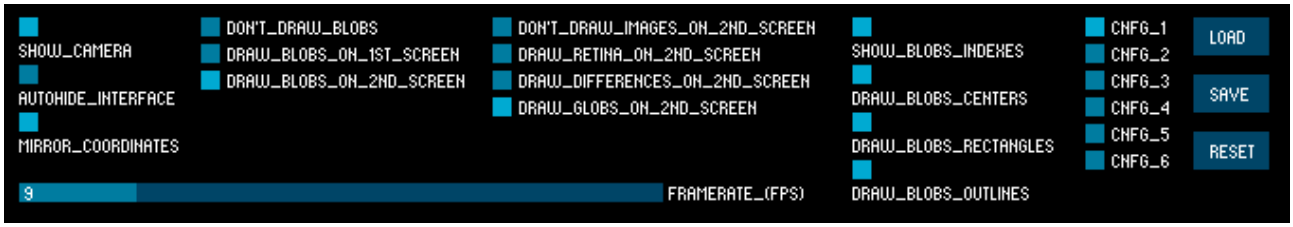


Po chwili aplikacja wyświetli główny zestaw kontrolek interfejsu użytkownika, w tym dwa panele z podglądem wideo (panel po lewej stronie pokazuje obraz z kamery) – klikając dowolną zakładkę zauważymy, że panele podglądu wideo są zawsze widoczne.

Interfejs głównego (i jedynego) okna aplikacji podzielony jest na zakładki grupujące tematycznie poszczególne parametry. Pomiędzy zakładkami przemieszczamy się klikając przyciski (opisane: **MAIN**, **MOCAP**, **OSC**, **ADVANCED**) w lewej górnej części okna.

MAIN

Zakładka **MAIN** grupuje podstawowe parametry dotyczące wyświetlania oraz opcji zapisu i ładowania pliku konfiguracyjnego.



Po lewej stronie zakładki **MAIN** znajduje się grupa 3 pól wyboru: *show camera*, *autohide interface* i *mirror coordinates*. Pole *show camera* pozwala włączyć lub wyłączyć wyświetlanie obrazu z kamery (lub innego urządzenia przechwytywania wideo, z którym w danym momencie współpracuje aplikacja) w lewym panelu wideo (lewy panel wideo opisany jest jako *1st screen*).

Aktywizacja opcji *autohide interface* pozwala na ukrywanie wszystkich elementów interfejsu graficznego (łączenie z panelami wideo), jeśli użytkownik nie wykona przez kilka sekund żadnej akcji (nie wciśnie żadnego klawisza i nie użyje myszy) – pozwala to zaoszczędzić trochę mocy obliczeniowej w przypadku pracy ze słabszymi komputerami. Wciśnięcie klawisza lub ruch myszą przywraca interfejs. Gdy interfejs jest ukryty analiza obrazu i transmisja danych *via* OSC przebiegają normalnie, a program wyświetla wówczas pulsujący napis *Move mouse or press any key to restore interface.* – pulsowanie sygnalizuje prawidłową pracę programu.

Pole (standardowo jest ono aktywne) *mirror coordinates* pozwala natomiast włączyć lustrzane odbicie obrazu.

Suwak *framerate* (wyskalowany w klatkach na sekundę, czyli *fps*) służy do ustalenia częstotliwości, z jaką będą analizowane kolejne klatki wideo. Wyższe częstotliwości oznaczają lepszą reaktywność programu, ale zarazem bardziej obciążają procesor. Ponadto parametr *framerate* wpływa też na odświeżanie samego okna programu.

Grupa trzech przycisków opisanych jako *don't draw blobs*, *draw blobs on 1st screen* i *draw blobs on 2nd screen* służy do wyboru panelu, w którym wyświetlane będą wizualizowane parametry wykrytych w kadrze ruchomych obiektów. Poza wyborem lewego lub prawego panelu można całkowicie wyłączyć wyświetlanie obiektów.

Cztery, umieszczone mniej więcej w środkowej części interfejsu, przyciski *don't draw images on 2nd screen*, *draw retina on 2nd screen*, *draw differences on 2nd screen*, *draw globs on 2nd screen* umożliwiają wybór obrazu wyświetlanego na prawym panelu. Poza oczywistą możliwością wyłączenia wyświetlania możemy wybrać wyświetlanie obrazu tła (*retina*), obrazu różnic pomiędzy tłem i aktualną klatką wideo (*differences*) lub obrazu wykrytych obiektów (*globs*) – ten ostatni obraz powstaje poprzez porównanie obrazu tła i bieżącej klatki obrazu a następnie zastosowania dla

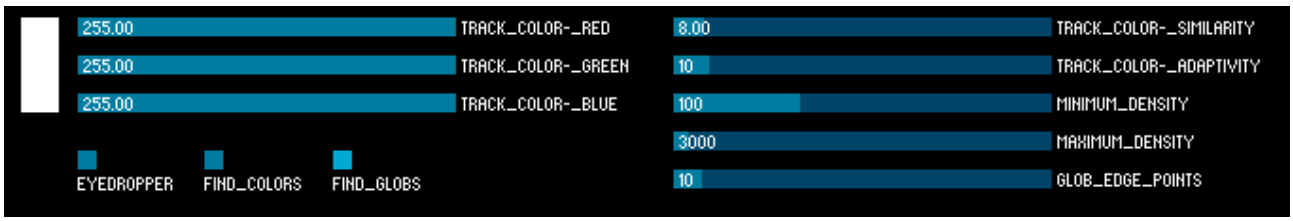
każdego piksela operacji porównania z zadanym progiem (kontrolki parametrów pozwalających dostroić algorytm generowania tła i porównywania go z kolejnymi klatkami obrazu znajdują się w zakładce **MOCAP**).

Grupa czterech pól wyboru umieszczonych na prawo od opisanego przed chwilą zestawu kontrolek umożliwia określenie, które parametry ruchomych obiektów mają być wizualizowane – warto w tym miejscu zaznaczyć, że ustawienia określane tymi przyciskami zadziałają tylko wtedy, jeśli nie wyłączymy wyświetlania ruchomych obiektów (wybierając opcję *don't draw blobs* z grupy *don't draw blobs*, *draw blobs on 1st screen* i *draw blobs on 2nd screen*). Pole *show blobs indexes* pozwala pokazać lub ukryć identyfikatory obiektów. Pola *draw blobs centers*, *draw blobs rectangles* i *draw blobs outlines* umożliwiają natomiast włączenie lub wyłączenie wyświetlania centrów obiektów, prostokątów, w których się zawierają i ich obrysów.

Prawa część interfejsu (zbór kontrolek opisanych *cnfg 1 – cnfg 6* oraz przyciski *load*, *save* i *reset*) odpowiadają za zarządzanie konfiguracją programu. Ustawienia (dotyczące różnych aspektów pracy aplikacji, transmisji komunikatów OSC, czułości systemu śledzenia ruchu, itp.) dokonane za pomocą kontrolek interfejsu użytkownika mogą zostać zapisane (przycisk *save*) do wyselekcjonowanej aktualnie konfiguracji (*cnfg 1 – cnfg 6*). Wyselekcjonowana konfiguracja może zostać w dowolnym momencie przywołana przyciskiem *load*. Konfiguracja zapisana pod numerem „1” jest przywoływana automatycznie podczas startu aplikacji. Ponadto przyciskiem *reset* możemy w dowolnym momencie zresetować ustawienia większości kontrolek.

Grupa trzech przycisków (opisanych: *no image on 2nd screen*, *draw stage on 2nd screen*, *draw differences on 2nd screen*) odpowiada za zawartość panelu podglądu wideo po prawej stronie okna. Wybranie pierwszego przycisku spowoduje wyłącznie panelu, wybranie drugiego (*draw stage on 2nd screen*) wyświetli podgląd obrazu utworzonego w rezultacie porównania tła i bieżącej klatki obrazu, a trzeciego (*draw differences on 2nd screen*) wyświetli podgląd obrazu utworzonego w rezultacie porównania tła i bieżącej klatki obrazu a następnie zastosowania dla każdego piksela operacji porównania z zadanym progiem (suwak *threshold* z zakładki **ZONES**) w celu sprawdzenia, czy różnica jasności odpowiadających sobie współrzędnymi pikseli tła i bieżącej klatki wideo wskazuje na wystąpienie ruchu (niewielkie wartości wskazują raczej na szum tła, przypadkowe zmiany oświetlenia, itp.). Obraz tła jest generowany na bieżąco, na podstawie kilkudziesięciu ostatnich klatek obrazu z kamery (jest to prosta, ale skuteczna technika, dzięki której system m.in. sam przystosowuje się – w pewnych granicach – do warunków oświetleniowych).

MOCAP



Ta zakładka grupuje kontrolki związane z parametrami systemu śledzenia ruchu.

Trzy suwaki *track color red*, *track color green* i *track color blue* oraz skojarzone z nimi pole podglądu ustawionego koloru odpowiadają za wybór barwy, która będzie wyłuskiwana z obrazu powstałego w wyniku porównania obrazu tła z aktualną klatką obrazu pobranego z urządzenia wideo. Innymi słowy: poruszające się w kadrze

obiekty muszą odpowiadać wzorcowi kolorystycznemu bazującemu na barwie ustawionej za pomocą opisywanych tutaj suwaków i tolerancji ustawianej znajdującym się po prawej stronie suwakiem *track color similarity*. W praktyce ustawienie bazujące na kolorze białym i odpowiednio dużej tolerancji (czyli niskim ustawieniu parametru *similarity*!) spowoduje, że każdy ruchomy obiekt będzie wykrywany przez system śledzenia ruchu.

Pod suwakami ustawień składowych koloru znajduje się zestaw 3 pól wyboru. Pole *eyedropper* służy do włączania /wyłączania alternatywnej w stosunku do opisanej powyżej metody ustawienia śledzonego koloru – po zaznaczeniu pola *eyedropper* wystarczy kliknąć interesujące nas miejsce w kadrze (w lewym lub prawym panelu wideo) by pobrać jego kolor.

Pole *find color* umożliwia włączanie / wyłączanie procedury wykrywającej kolory centrów śledzonych obiektów.

Pole *find globs* aktywuje / dezaktywuje algorytm wykrywania obiektów – oczywiście dezaktywacja systemu śledzenia ruchu powoduje, że nie będą uzyskiwane żadne dane o poruszających się w kadrze obiektach.

Rząd suwaków po prawej stronie pozwala dostroić system śledzenia ruchu do naszych potrzeb. Wspomniany wyżej suwak *track color similarity* służy ustawieniu tolerancji w stosunku do bazowego koloru – w zależności od ustawień suwaka możemy sprecyzować, jak bardzo barwa wykrywanych obiektów może się różnić od koloru bazowego. Suwak ten umożliwia – w praktyce – ustawienie progu zadziałania sensora ruchu (jest to wartość różnicy w poziomie jasności piksela obrazu tła i piksela [o tych samych współrzędnych] bieżącej klatki obrazu [pod warunkiem, że odpowiada ustawionemu wzorcowi kolorystycznemu] wystarczająca do uznania, że został wykryty „ruch").

Umieszczony poniżej suwak *track color adaptivity* pozwala określić szybkość reakcji algorytmu detekcji tła – aby przekonać się, w jaki sposób ustawienia tego parametru wpływają na algorytm detekcji tła najlepiej poeksperymentować włączając prawy panel wideo w tryb *retina* (zakładka **MAIN**, kontrolka *draw retina on 2nd screen*) i manewrując ustawieniami suwaka.

Suwaki *minimum density* i *maximum density* umożliwiają ustawienie minimalnej i maksymalnej wielkości wykrywanych obiektów – standardowe ustawienia obu suwaków są w miarę uniwersalne, ale oczywiście warto eksperymentować samodzielnie poszukując najlepszych dla konkretnych warunków ustawień obu parametrów (w zależności od użytej kamery, warunków oświetleniowych, itd.).

Ostatni suwak – *glob edge points* – pozwala na ustawienie precyzji wykrywania konturów śledzonych obiektów. Parametr kontrolowany tym suwakiem pozwala na ustalenie maksymalnej ilości punktów definiujących kontur każdego z obiektów.

OSC

Zakładka **OSC** zawiera pola pozwalające na edycję większości parametrów transportu danych protokołem OSC (część parametrów związanych z OSC znajduje się również w zakładce **ADVANCED**).

W lewej części sekcji znajdują się kontrolki umożliwiające wybór sposobu, w jaki transmitowane będą komunikaty OSC. Wybranie pbl spowoduje włączeniu trybu, w którym transmitowane komunikaty są zgodne z aplikacją-klientem instalacji interaktywnej Malownie Światłem (Painting by Light: PBL) – komponentu wystawy Interaktywny Plac Zabaw (pierwsze wersje **JMyron2OSC** powstały na potrzeby IPZu). Tryb oceanus służy komunikacji z pozostałymi komponentami instalacji interaktywnej o tym samym tytule (więcej informacji: http://www.moving-stories.eu/artists/pawel-janicki). Tryby *pbl* i *oceanus* nie będą więc przydatne dla szerszego grona odbiorców, którzy mogą jednak skorzystać z dwóch innych (i w zasadzie głównych) trybów komunikacji.

Tryby *short messages* i *long messages* pozwalają na wybór jednego z dwóch głównych trybów transmisji komunikatów OSC. Jeśli wybierzemy tryb *long messages*, wówczas aplikacja podzieli uzyskane dane na niewielką ilość długich komunikatów OSC o zmiennej długości. W trybie *short messages* **JMyron2OSC** generuje serię komunikatów sformatowaną w taki sposób, by każdy z nich miał stałą długość (tj. zawsze tę samą ilość parametrów – głównym powodem możliwości wystąpienia zmiennej ilości parametrów w niektórych komunikatach jest fakt, że obrys wykrytego obiektu może składać się z różnej ilości punktów). Ma to (pisałem już o tym wyżej, ale warto to przypomnieć) praktyczne znaczenie w przypadku sprzęgania **JMyron2OSC** z wizualnymi językami programowania (MaxMSP. PureData, itp.), ponieważ wielu użytkownikom tych ostatnich trudno jest przetwarzać komunikaty OSC o zmiennej długości bez sięgania po skrypty lub inne techniki wiążące się z głębszą znajomością tych środowisk. Niezależnie, czy pracujemy w trybie *short messages*, czy *long messages* komunikaty dotyczące kolejnych analizowanych klatek obrazu są wysyłane jako tzw. *bundle* – rodzaj „paczki” komunikatów grupującej dane związane z jedną klatką obrazu.

Pole wyboru *autotransmit mocap as osc messages* służy włączeniu lub zablokowaniu transmisji komunikatów.

Przycisk *manually send osc message* pozwala na asynchroniczne (w stosunku do zwykłego cyklu) wysłanie danych o wykrytych obiektach.

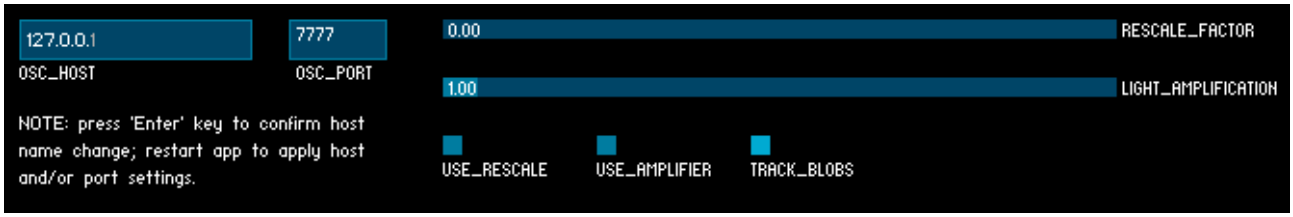
Suwak *max osc datarate* (wyskalowany w sekundach) służy ograniczeniu częstotliwości wysyłania przez program komunikatów OSC. Opcja ta została wprowadzona podczas testów **JMyron2OSC** z flOSC (code.google.com/p/flosc/) i projektami stworzonymi w Adobe Flash/Flex (http://adobe.com). Najprawdopodobniej w niektórych wypadkach przepustowość flOSC i ActionScript okazywała się zbyt niska i wówczas ograniczenie częstotliwości wysyłania komunikatów było sensownym rozwiązaniem. Problem pojawiał się sporadycznie i nie został zaobserwowany w przypadku projektów korzystających z ostatnich edycji Adobe FlashPlayera i ActionScript 3.0. Jeśli nie będzie to wywoływało negatywnych konsekwencji warto więc ustawić suwak *max osc datarate* na wartość „0” (zero) – zapewni minimalne opóźnienia w transmisji danych *via* OSC.

Pozostałe kontrolki umieszczone w zakładce: a więc grupa pól wyboru oznaczonych: *send indexes*, *send blobs centers*, *send blobs rectangles*, *send blobs outlines*, *send blobs centers colors* i *send mocap statistics* służą wskazaniu, jakie dane o stanie systemu śledzenia ruchu i wykrytych obiektach mają być transmitowane.

Szczegółowe dane dotyczące składni generowanych przez **JMyron2OSC** komunikatów OSC znajdują się w dokumencie „OSC_syntax”.

Więcej o OpenSound Control na stronie: http://opensoundcontrol.org.

ADVANCED



Zakładka **ADVANCED** zawiera zestaw kontrolek umożliwiających manipulację rzadziej wykorzystywanymi (lub wymagającymi pewnego doświadczenia w pracy z aplikacją) parametrami.

Dwa pola tekstowe – *osc host* i *osc port* – pozwalają na edycję parametrów transportu danych protokołem OSC. Edycji podlegają: parametry określające host i port (czyli lokalizacja adresata komunikatów). Zmiana hosta lub portu wymaga restartu aplikacji, zmiana hosta potwierdzenia klawiszem „Enter”.

Suwak *rescale factor* umożliwia przeskalowanie danych wykrytych obiektów (parametr kontrolowany tym suwakiem dotyczy wielkości obiektów, więc wpływa koordynaty prostokąta, w który wpisany jest obiekt i koordynaty punktów obrysu). Aby ustawienia suwaka zadziałały (czyli aby obiekty były przeskalowywane) pole wyboru *use rescale* musi być włączone.

Suwak light amplification pozwala pozwala wpłynąć na algorytm wykrywania kolorów centrów wykrywanych obiektów. Aby ustawienia suwaka wpływały na algorytm wykrywania barw musimy włączyć pole wyboru use *amplifier*.

Ostatnie pole wyboru w zakładce **ADVANCED**, czyli *track blobs* pozwala na aktywację lub dezaktywację algorytmu nadającego wykrywanym obiektom unikalne indeksy pozwalające śledzić trasę i inne parametry każdego poruszającego się obiektu niezależnie od innych wykrytych w tym samym czasie obiektów.