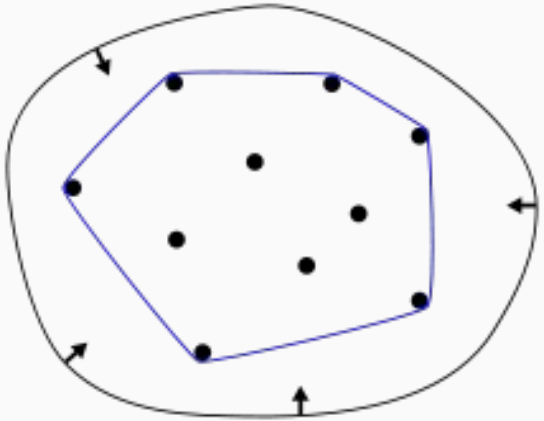
Otoczka wypukła dla zbioru punktów w przestrzeni dwuwymiarowej

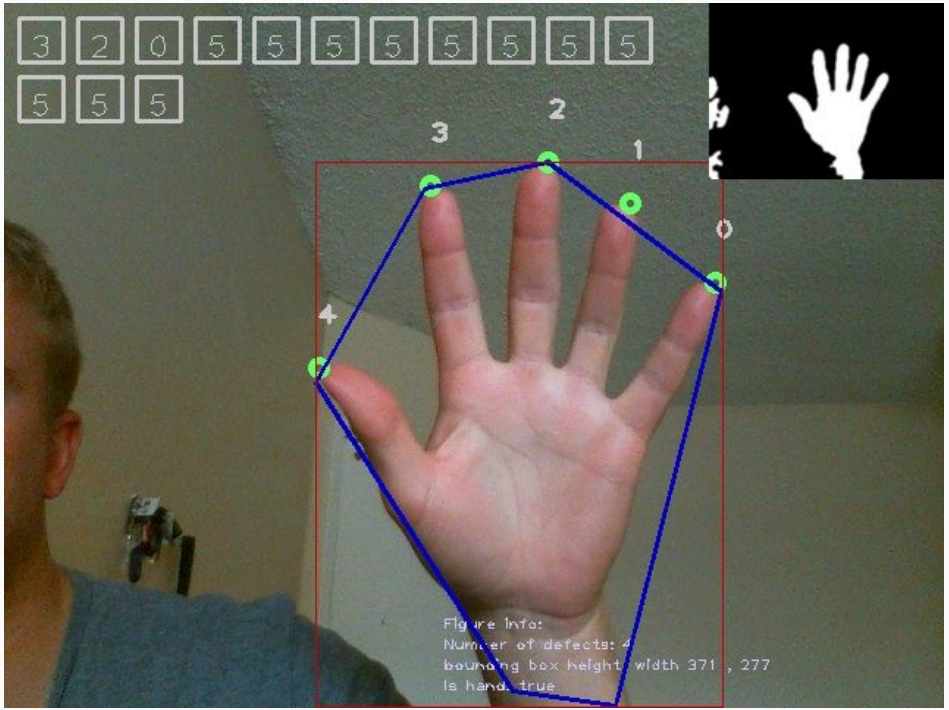
Wojciech Dymek

Agenda

- Otoczka wypukła
- Algorytmy:
  - **dziel i rządź**
  - **QuickHull**
  - **Chan**
- Bibliografia

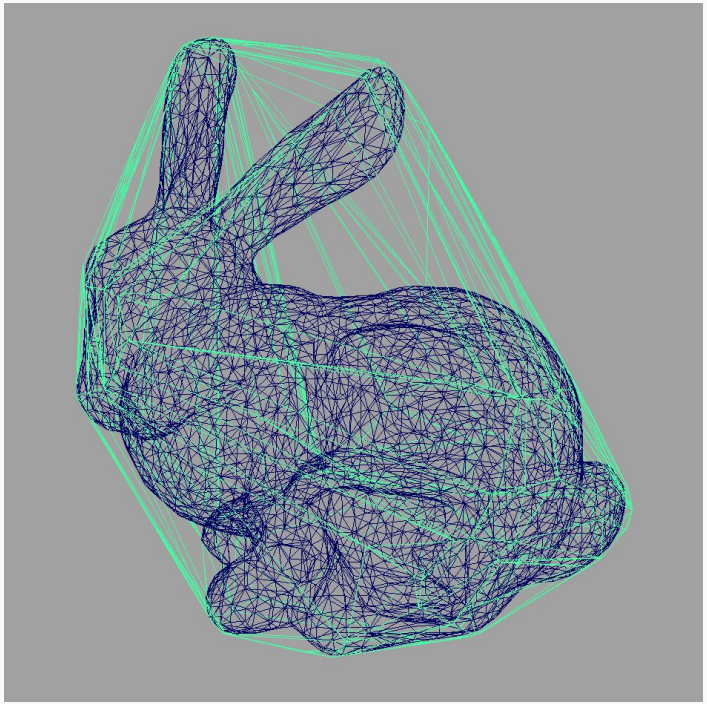
Otoczka wypukła

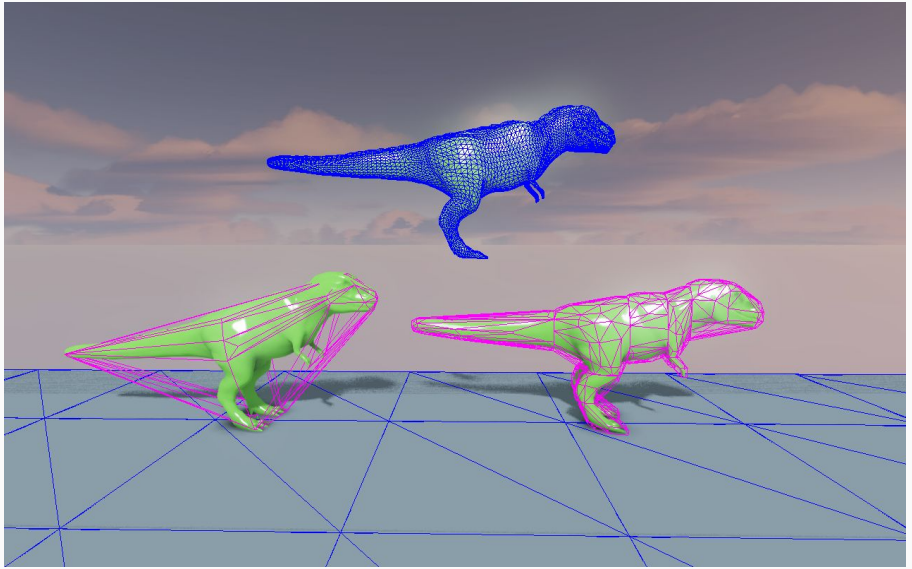




http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/

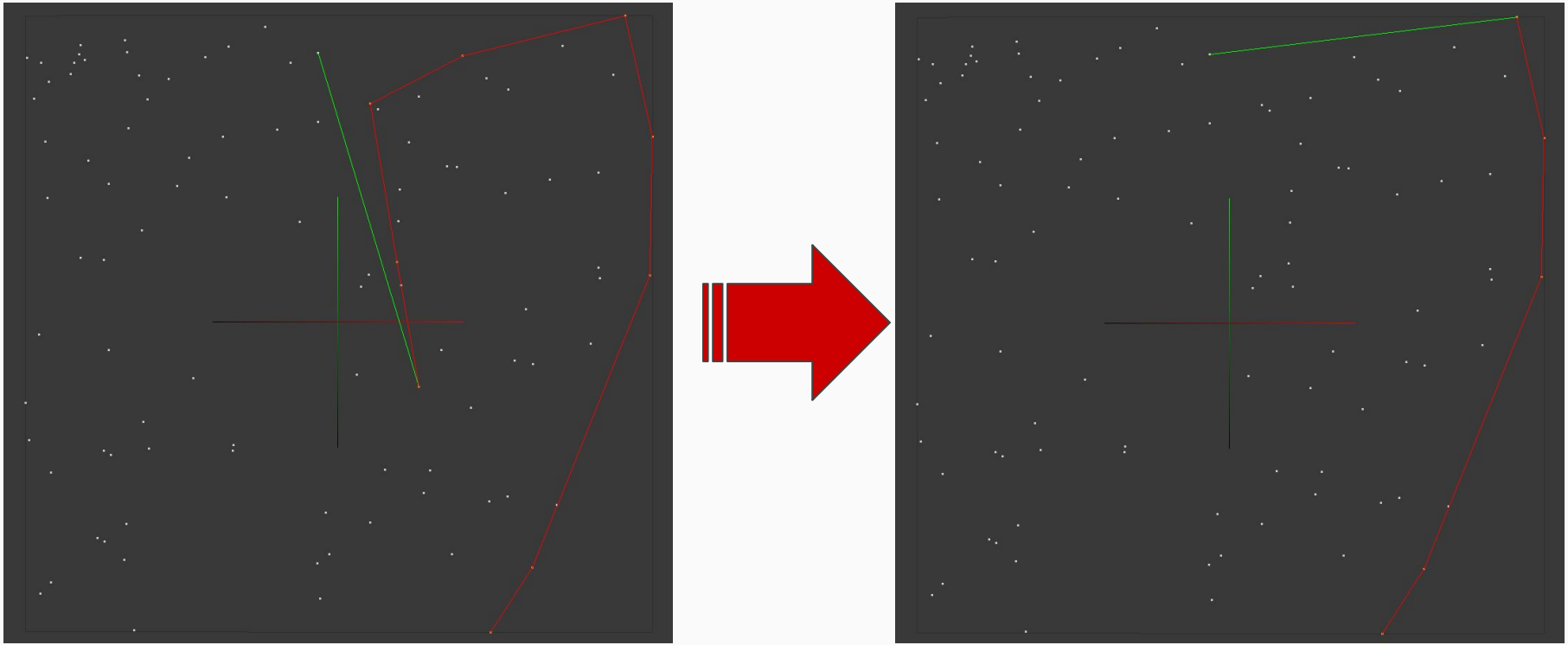
Otoczka wypukła



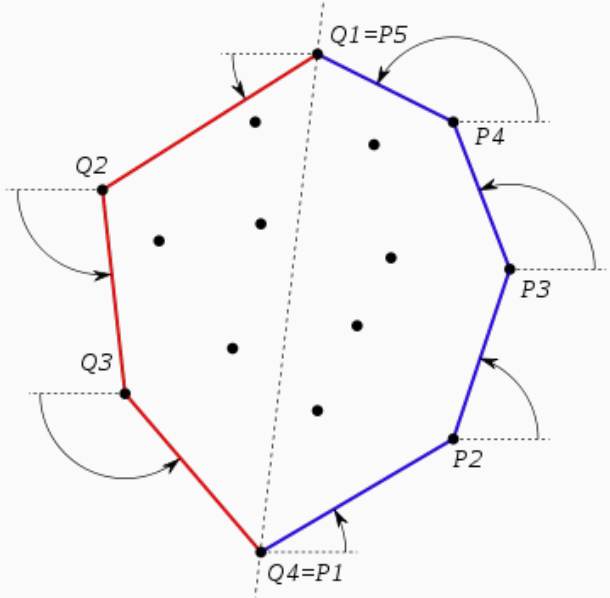


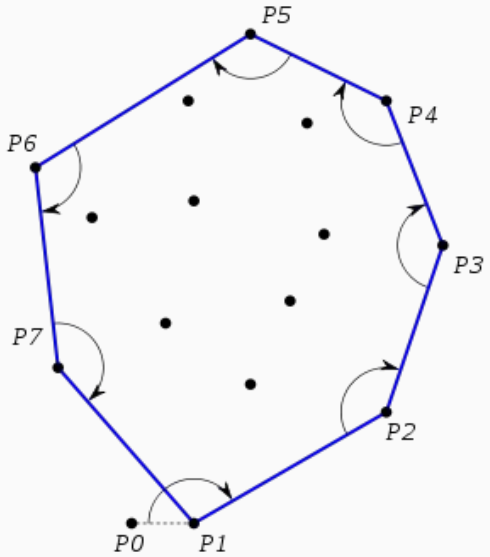
https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects

Algorytm Grahama



Algorytm Jarvisa





źródło: https://pl.wikipedia.org/wiki/Algorytm_Jarvisa

Algorytm Divide And Conquer

S - wejściowy zbiór punktów

k - zadana stała

F={S}

1. Dopóki którykolwiek zbiór z F jest większy niż stała k, podziel zbiór ze względu na medianę x-owych współrzędnych
2. Wyznacz opoczki zbiorów z F
3. Sklej otoczki

Algorytm QuickHull

S - wejściowy zbiór punktów

1. wybierz ***a*** - skrajnie lewy punkt oraz ***b*** - skrajnie prawy punkt
2. podziel zbiór ***S*** na dwa podzbiory:

   A - zbiór punktów znajdujących się nad prostą ***ab***

   B - zbiór punktów znajdujących się pod prostą ***ab***
3. wykonaj:

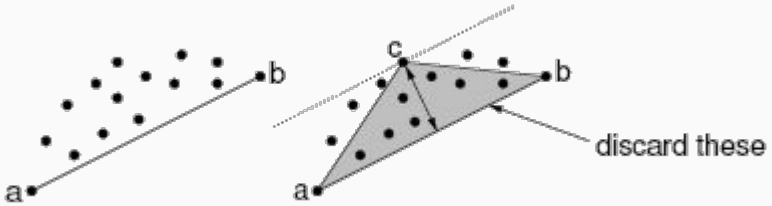
   CH_A <- *QuickHull(**A**, **ab**)*

   CH_B <- *QuickHull(**B**, **ab**)*
4. wynikiem algorytmu jest $CH_A + a + CH_B + b$

Algorytm QuickHull

QuickHull(***P***, ***ab***):

1. ***c*** <- punkt z P najbardziej odległy od prostej ***ab***
2. $\boldsymbol{P}_{\boldsymbol{ac}}$ <- zbiór punktów leżących na lewo od prostej ***ac***
3. $\boldsymbol{P}_{\boldsymbol{cb}}$ <- zbiór punktów leżących na lewo od prostej ***cb***
4. $\boldsymbol{CH}_{\boldsymbol{ac}}$ <- *QuickHull*($\boldsymbol{P}_{\boldsymbol{ac}}$, ***ac***)
5. $\boldsymbol{CH}_{\boldsymbol{cb}}$ <- *QuickHull*($\boldsymbol{P}_{\boldsymbol{cb}}$, ***cb***)
6. zwróć $\boldsymbol{CH}_{\boldsymbol{cb}}$ + ***c*** + $\boldsymbol{CH}_{\boldsymbol{ac}}$

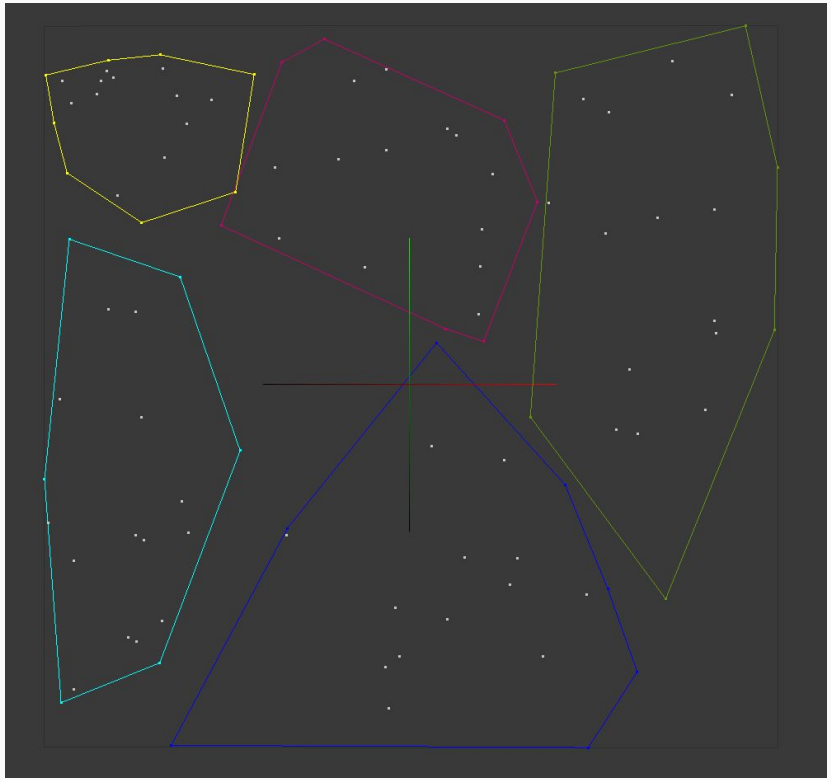


Algorytm Chan’a

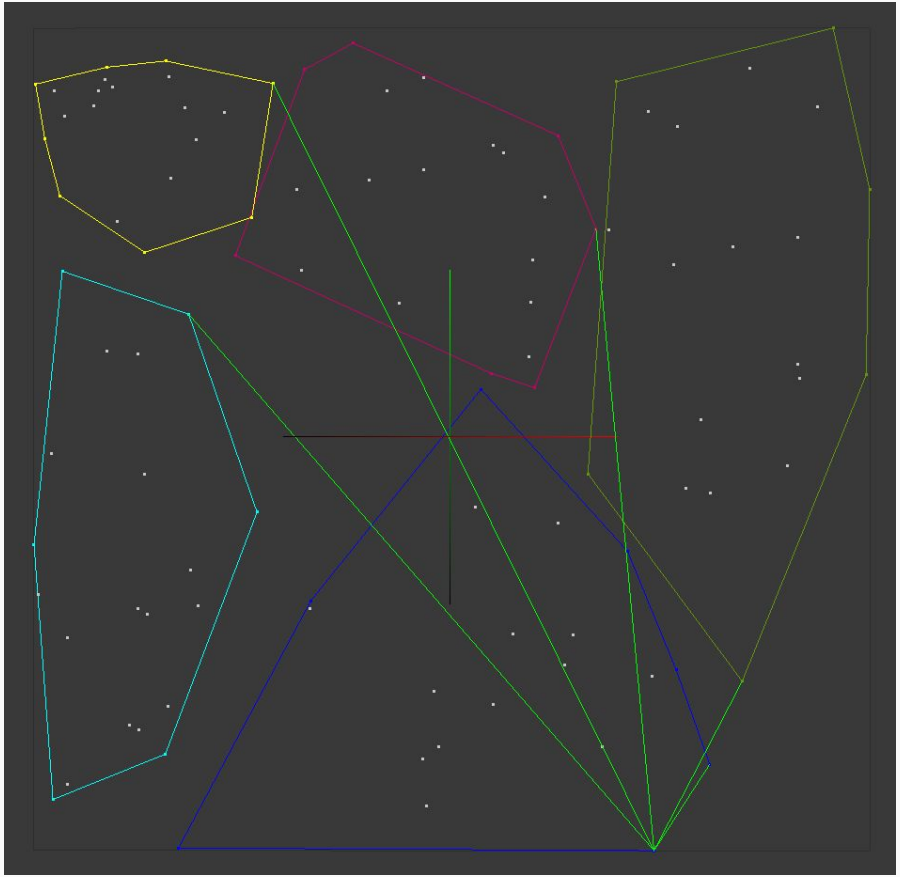
S - wejściowy zbiór punktów

m - zadana stała (najlepiej rozmiar otoczki)

1. Podziel zbór ***S*** na ***m***-elementowe podzbiory
2. Wyznacz otoczki dla podzbiorów używając algorytmu Grahama
3. Uruchomić algorytm Jarvisa dla grup



Algorytm Chan’a



http://geomalgorithms.com/a15-_tangents.html

http://tomswitzer.net/2010/12/2d-convex-hulls-chans-algorithm/

Algorytm Chan'a

```
def _rtangent(hull, p):
    # Return the index of the point in hull that the right tangent line from pto hull touches.
    l, r = 0, len(hull)
    l_prev = turn(p, hull[0], hull[-1])
    l_next = turn(p, hull[0], hull[(l + 1) % r])
    while l < r:
        c = (l + r) / 2
        c_prev = turn(p, hull[c], hull[(c - 1) % len(hull)])
        c_next = turn(p, hull[c], hull[(c + 1) % len(hull)])
        c_side = turn(p, hull[l], hull[c])
        if c_prev != TURN_RIGHT and c_next != TURN_RIGHT:
            return c
        elif c_side == TURN_LEFT and (l_next == TURN_RIGHT or l_prev == l_next) or c_side == TURN_RIGHT and c_prev == TURN_RIGHT:
            r = c               # Tangent touches left chain
        else:
            l = c + 1           # Tangent touches right chain
            l_prev = -c_next    # Switch sides
            l_next = turn(p, hull[l], hull[(l + 1) % len(hull)])
    return l
```

Bibliografia

- https://pl.wikipedia.org/wiki/Quickhull
- http://geomalgorithms.com/a15-_tangents.html
- http://tomswitzer.net/2010/12/2d-convex-hulls-chans-algorithm/
- http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/