Unha aproximación Deep Learning para render de nubes de puntos

Martín Sánchez Fontao

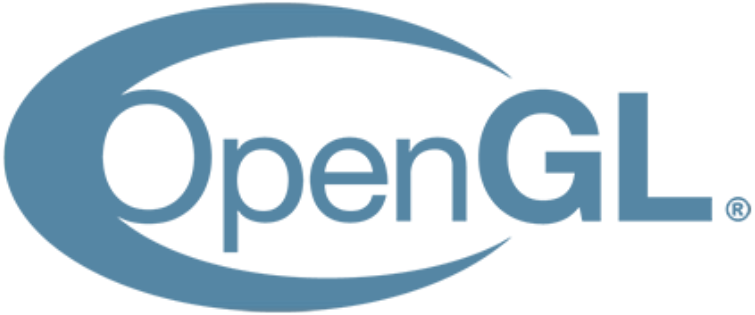
Visualización de gráficos 3D

Machine Learning

APIS Renderizado 3D

API aberto

Especificación estándar

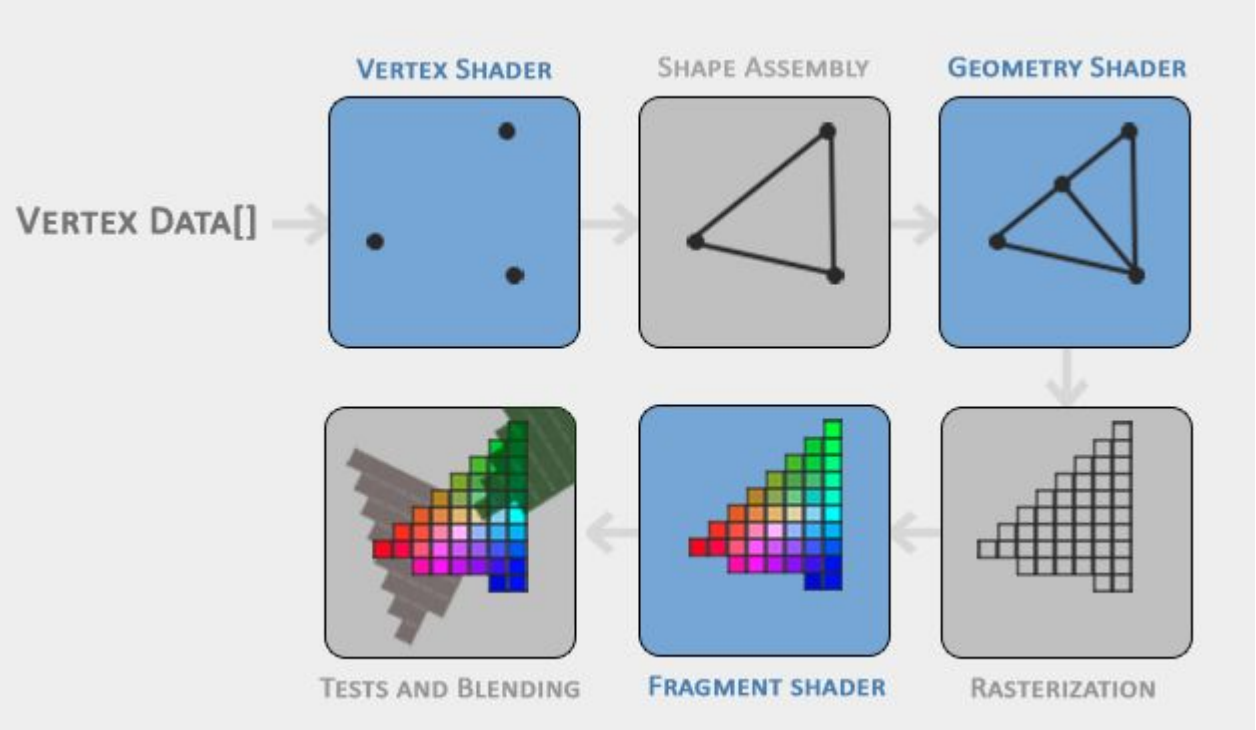


Multiplataforma

Microsoft® DirectX®

Métodos clásicos de renderizado

Pipeline gráfico



Representación dos vértices do obxecto

Proxección de 3D a 2D en pantalla.

Paso de xeometría a píxeles en pantalla.

Etapas programables

Nubes de puntos

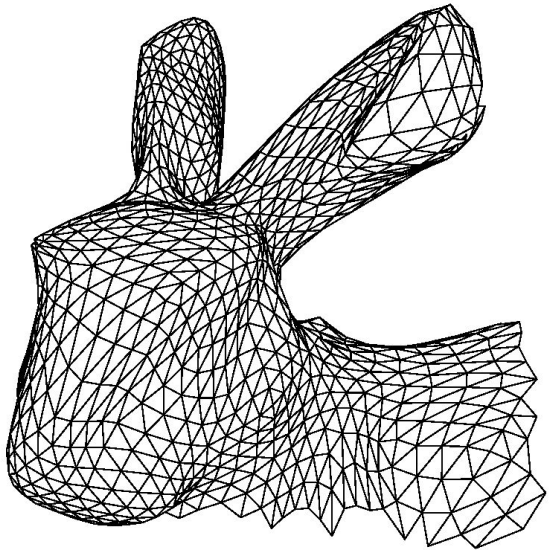
Que son

Como se obteñen

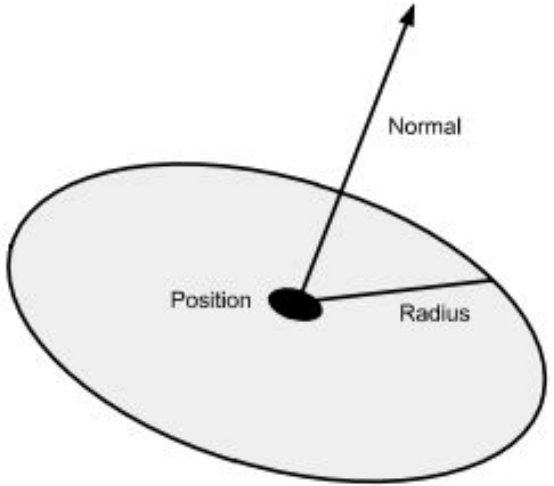
Pros e contras

Nubes de puntos e normais

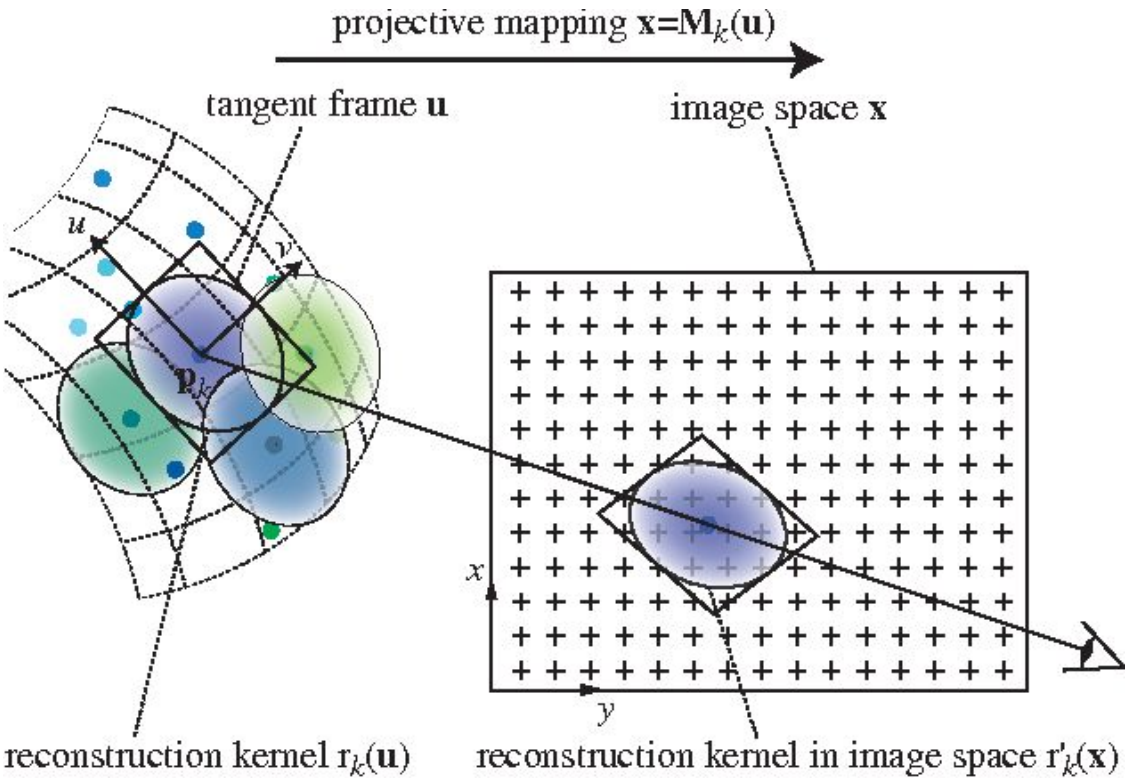
Malla de polígonos



Splatting



Splatting


projective mapping $\mathbf{x}=\mathbf{M}_k(\mathbf{u})$
tangent frame $\mathbf{u}$
image space $\mathbf{x}$
u
v
$\mathbf{p}_k$
x
y
reconstruction kernel $r_k(\mathbf{u})$
reconstruction kernel in image space $r'_k(\mathbf{x})$

Visualizador Ipoint

Cargar normais do método de Deep Learning

Cargar normais mediante PCL

Comparativa entre os métodos de estimación

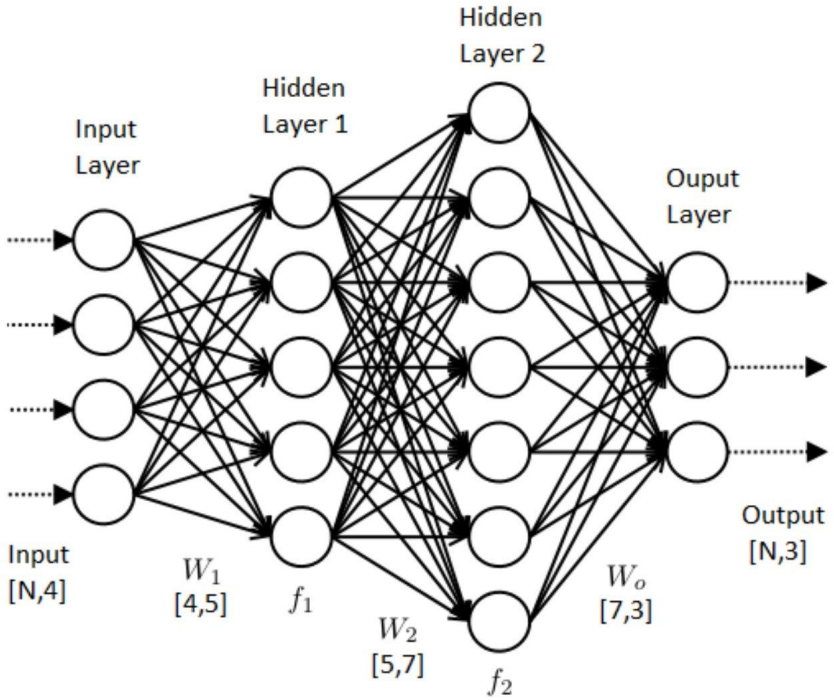
Cargar nubes de puntos

Visualizar normais cargadas

Librerías para Deep Learning

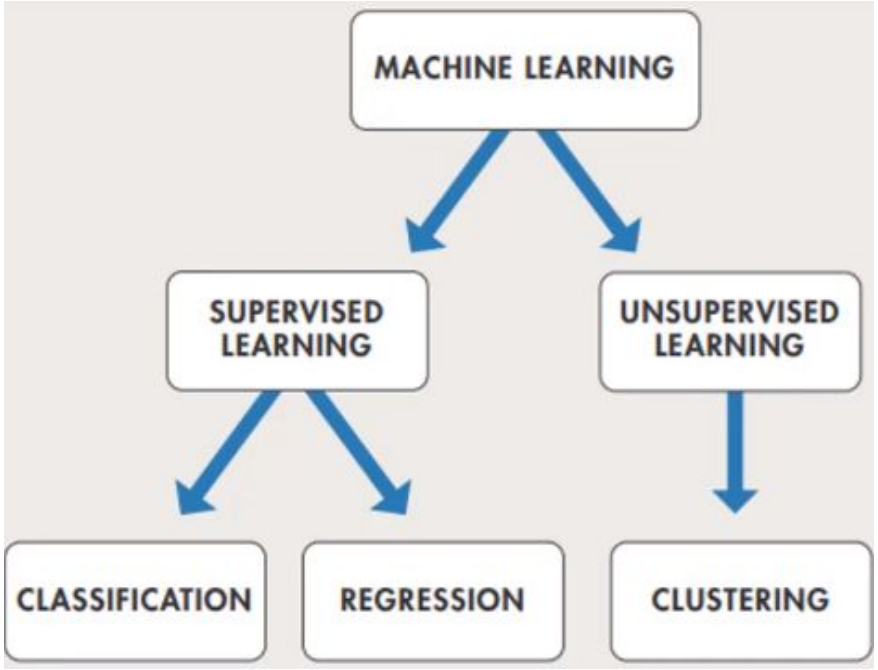
Machine learning, Deep Learning e ANN

Dataset



Predicción

Modelos posibles

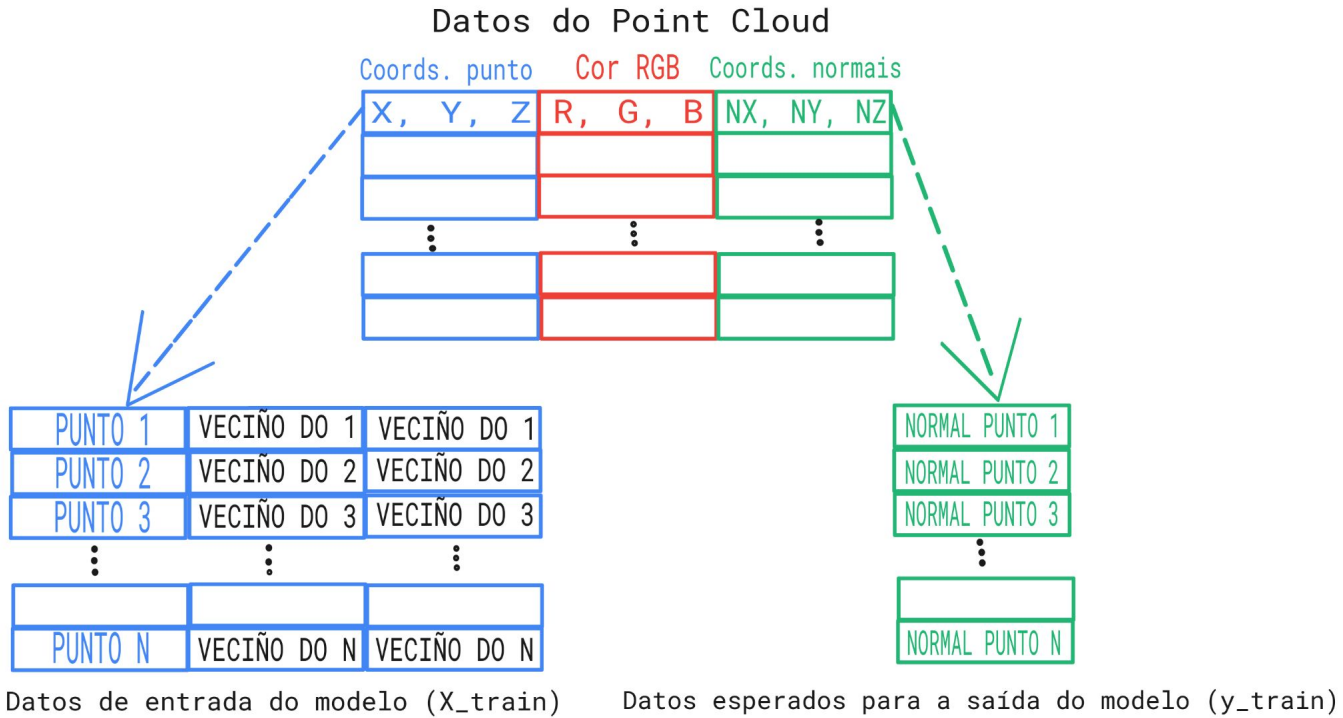


Aprendizaxe supervisado

Modelo de regresión

¿Cómo funciona?

Formato dos datos de entrada

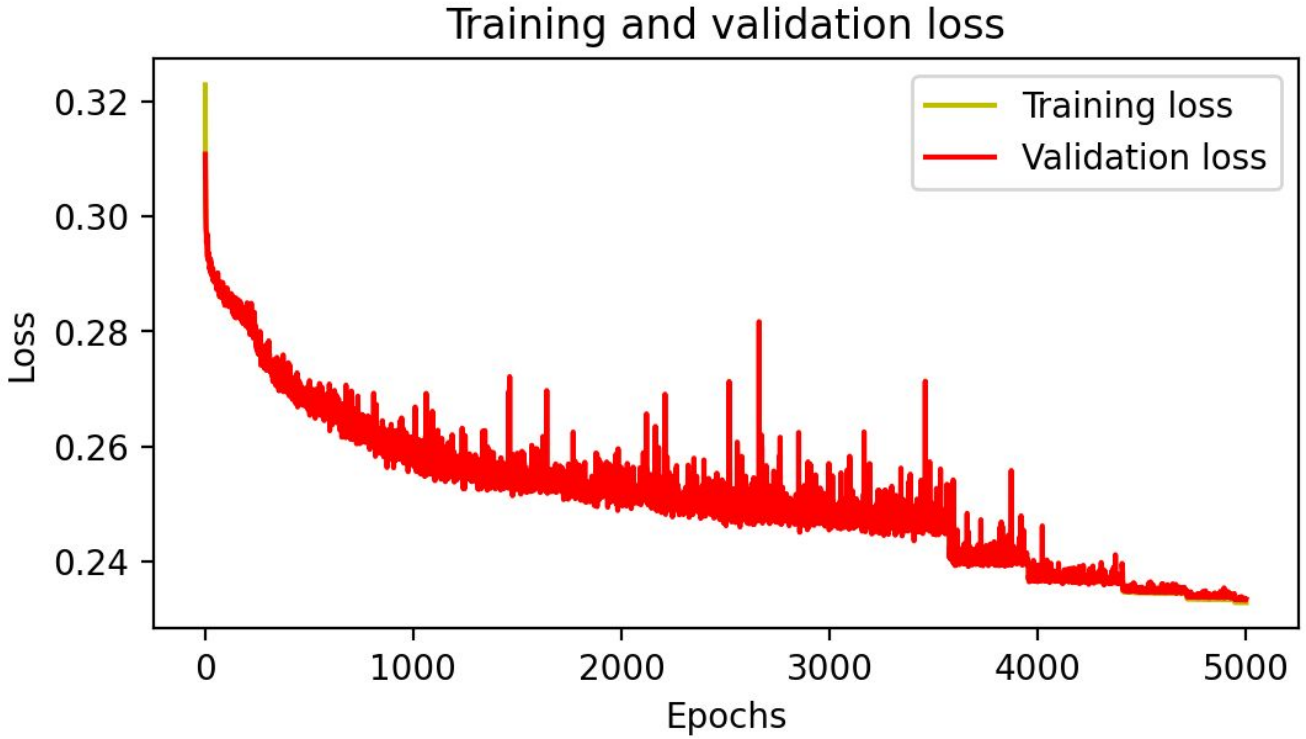


Modelo de Deep Learning

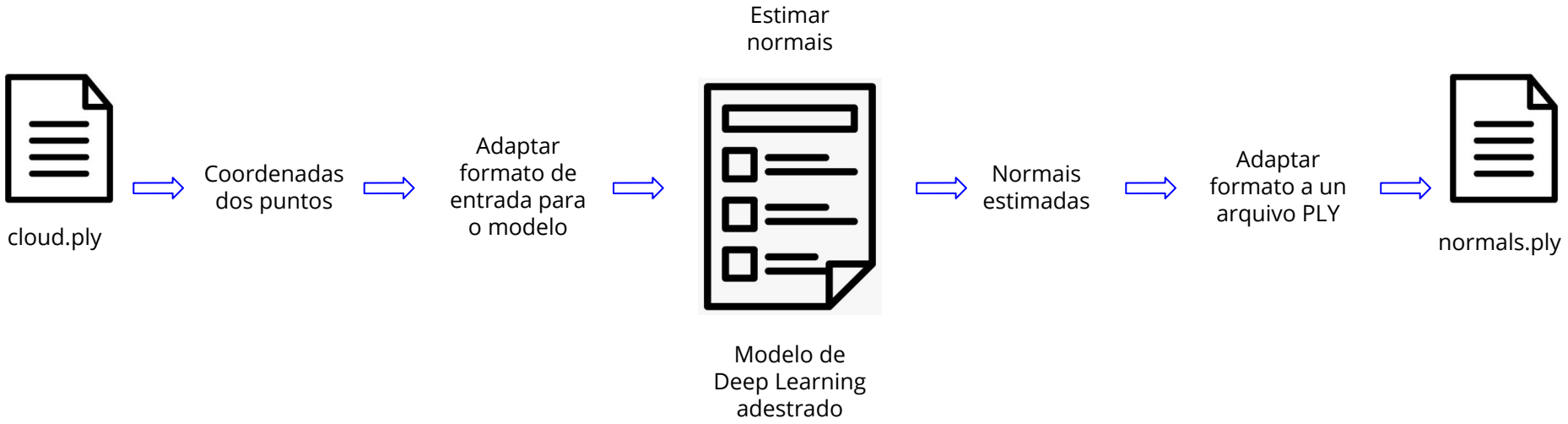
```
## Modelo de regresión.

# Cargamos os datos de entrada.
def load_inputs():
    # Datos dos que predeciremos as normais.
    X=load(arquivo)
    # Resultados esperados das normais que estimaremos.
    Y=load(arquivo)

    # Separamos os datos de entrada que usaremos para adestrar, dos
    que usaremos para comprobar se o adestramento está dando bos
    resultados.
    X_train, X_test, y_train, y_test = train_test_split(X,Y)

    return X_train, X_test, y_train, y_test
```

```
def create_model()
    model=Modelo de regresión
    model.add(input)
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(3))
    model.compile(loss=mae, optimizer=adam(learning_rate))

    return model

def fit_model(X_train, X_test, y_train, y_test)
    history=model.fit(X_train, y_train, batch_size, n_epochs,
    validation_data=(X_test, y_test))

    return history

def plot(history, epochs)
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    plot(loss, val_loss, epochs)
```

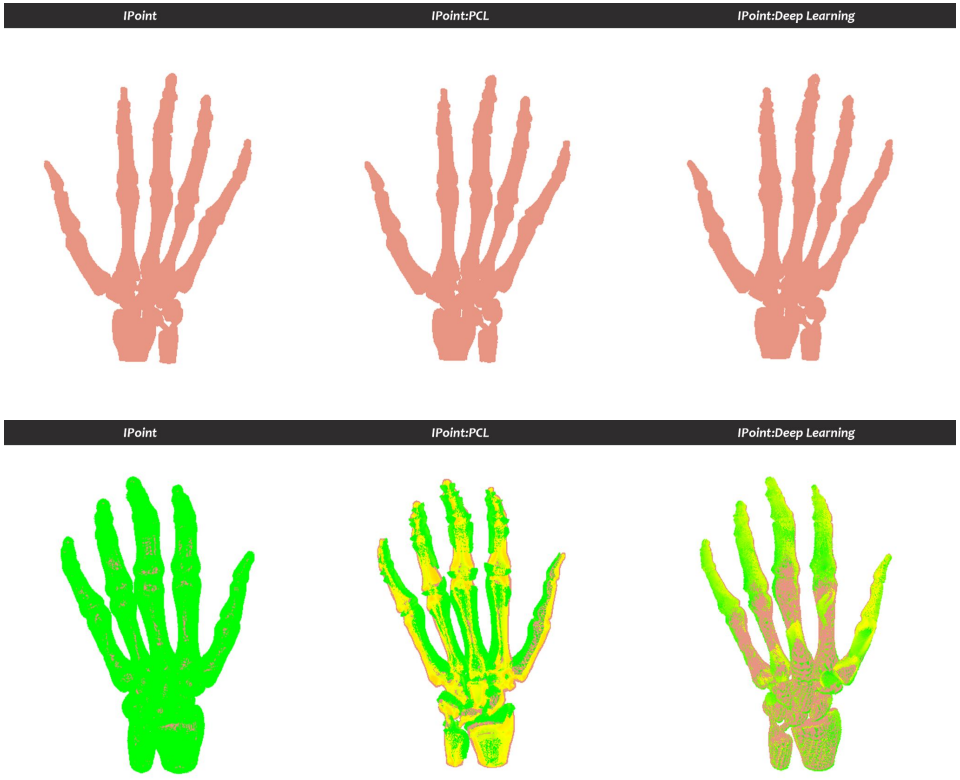
Resultados adestramento



Funcionamento NormalGenerator



Comparativa



Estimación Deep Learning	Arquivos de nubes de puntos (.ply)							
	suzanne30k	suzanne100k	hand	lucy	suzanne500k	blade	suzanne1M	head_ten24
Error medio	0,1855	0,1819	0,2697	0,2031	0,1516	0,2290	0,1577	0,1265
Desviación estandar	0,0438	0,0438	0,1008	0,0733	0,0445	0,0304	0,0369	0,0439
Nº errores > 10%	24.241	80.740	132.048	266.769	427.985	675.879	926.524	804.071
Nº errores > 25%	2.318	6.804	181.383	97.949	19.266	207.052	28.935	26.848
Nº errores > 50%	0	0	2.879	0	0	22	0	0
Nº errores > 75%	0	0	0	0	0	0	0	0

Estimación PCL	Arquivos de nubes de puntos (.ply)							
	suzanne30k	suzanne100k	hand	lucy	suzanne500k	blade	suzanne1M	head_ten24
Error medio	0,4588	0,4620	0,3101	0,3600	0,4636	0,2848	0,4626	0,4183
Desviación estandar	0,1293	0,1268	0,2267	0,1874	0,1251	0,2053	0,1261	0,1268
Nº errores > 10%	71	277	10.494	26.987	921	44.621	2.114	5.889
Nº errores > 25%	13.275	42.748	100.879	187.538	231.285	441.147	471.935	781.118
Nº errores > 50%	11.622	39.526	94.568	103.886	213.322	84.653	435.878	364.197
Nº errores > 75%	0	0	0	0	0	0	0	0

Conclusións e posible traballo futuro

Gran reto

· Tecnoloxías complexas e descoñecidas

Resultados

· Software cumple expectativas
· Buena estimación de normales
· Se cumplen los objetivos

Melloras

· Integración NormalGenerator con IPoint
· Axustar o modelo coa fin de mellorar a cota de erro

Demo

¡Moitas gracias polo voso tempo e atención!

¿Preguntas?