



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



# A Machine Learning based Recommendation System for furniture selection

---

Degree Thesis  
submitted to the Faculty of the  
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona  
Universitat Politècnica de Catalunya  
by

María Isabel Manresa Román

In partial fulfillment  
of the requirements for the degree in  
*Technologies and Services of Telecommunication* **ENGINEERING**

Advisor (UPC): Javier Ruiz Hidalgo  
Advisor (StageInHome): Pol Albacar Fernández  
Barcelona, Date 21/06/2022



## Abstract

Artificial Intelligence (AI) is one of today's fastest growing technologies, and has been evolving for decades. It allows machines to have the ability to "learn", and self-correct. This technology is used in many fields, such as decision making, diagnostics in medicine, pattern recognition and virtual reality among others.

This project has been carried out at StageInHome, a startup company specialized in AI and Deep Learning for interior decoration. In this case, the proposal is to build a bed recommendation system.

For this project, we have created databases of both images and metadata, an image retrieval according to resemblance, a classifier to differentiate bed types and finally a user interface that allows an easy use of the whole implemented system, including also a price filter.

## Resum

La Intel·ligència Artificial (IA) és una de les tecnologies amb més perspectiva de creixement d'avui en dia, i que porta evolucionant desde fa dècades. Aquesta tecnologia permet que màquines tinguin la capacitat d'aprendre, raonar o autocorregir-se i s'utilitza en camps molt diversos com en presa de decisions, diagnòstics de medicina, reconeixement de patrons i realitat virtual entre d'altres.

Aquest projecte s'ha desenvolupat a StageInHome, una empresa especialitzada en IA i Deep Learning per a la decoració automàtica d'espais interiors. En aquest cas, es proposa un sistema recomanador de llits.

Per aquest projecte, s'han creat bases de dades, tant de imatges com de metadades, un recomanador d'imatges segons la semblança, un classificador per diferenciar els tipus de llit i finalment una interfície d'usuari que permet un fàcil ús de tot el sistema implementat també un filtre de preu.

---

## Resumen

La Inteligencia Artificial (IA) es una de las tecnologías con más perspectivas de crecimiento de hoy en día, y que lleva evolucionando desde décadas atrás. Esta tecnología permite que máquinas tengan la capacidad de "aprender", "razonar" o autocorregirse y es utilizada en muchos campos, como en toma de decisiones, diagnósticos en medicina, reconocimiento de patrones y realidad virtual entre otros.

Este proyecto se ha llevado a cabo en StageInHome, empresa especializada en IA y Deep Learning para la decoración de espacios interiores. En este caso, se propone construir un sistema recomendador de camas.

Para este proyecto, se han creado bases de datos tanto de imágenes como de metadatos, un recomendador de imágenes según el parecido, un clasificador para diferenciar los tipos de cama y finalmente una interfaz de usuario que permite un fácil uso de todo el sistema implementado incluyendo también un filtro de precio.

---

## Acknowledgements

First of all, I would like to thank Javier Ruiz Hidalgo, for his supervision during the whole project and for his help during its development.

Also, I would like to thank Pol Albacar as well as all the StageInHome team for their help and support during the last four months I have been working on this project.

To my parents, for their unconditional support, for believing in me, even when I did not, and for doing all the efforts to make possible for me to study in the UPC.

And finally, to my partner, for the patience and for being by my side, making these years great despite the stress and difficulties.

## Revision history and approval record

Revision	Date	Purpose
0	24/04/2022	Document creation
1	17/06/2022	Document revision

### DOCUMENT DISTRIBUTION LIST

Name	
María Isabel Manresa Roman	
Javier Ruiz Hidalgo	
Pol Albacar Fernández	

Written by:		Reviewed and approved by:	
Date	15/06/2022	Date	21/06/2022
Name	María Isabel Manresa Román	Name	Javier Ruiz Hidalgo
Position	Project Author	Position	Project Supervisor

# Contents

<b>List of Figures</b>	<b>8</b>
<b>List of Tables</b>	<b>8</b>
<b>1 Introduction</b>	<b>10</b>
1.1 Project Overview and Statement of purpose . . . . .	10
1.2 Objectives . . . . .	10
1.3 Work Plan . . . . .	11
<b>2 State of the Art and Fundamentals</b>	<b>12</b>
2.1 Image Retrieval . . . . .	12
2.2 Segmentation . . . . .	14
2.2.1 Object Detection . . . . .	14
<b>3 Methodology</b>	<b>16</b>
3.1 Preprocessed Database . . . . .	16
3.2 Image Retrieval . . . . .	18
3.3 Segmentation and Object Detection . . . . .	19
3.4 Classification . . . . .	19
<b>4 Results</b>	<b>20</b>
4.1 Preprocessed Database . . . . .	20
4.1.1 Web Scraping . . . . .	20
4.1.2 Database preprocessing . . . . .	20
4.2 Image Retrieval . . . . .	22
4.3 Segmentation and Object Detection . . . . .	25
4.4 Classification . . . . .	28
4.5 User Interface . . . . .	32
<b>5 Budget</b>	<b>36</b>
5.1 Material . . . . .	36
5.2 Salaries . . . . .	36
5.3 Total Costs . . . . .	36
<b>6 Conclusions</b>	<b>37</b>
<b>7 Future Work</b>	<b>37</b>
<b>References</b>	<b>38</b>
<b>Appendices</b>	<b>40</b>
<b>A Work Plan</b>	<b>40</b>

## Lists

### List of Figures

1	Project overview diagram . . . . .	10
2	Gantt diagram (1) . . . . .	11
3	Gantt diagram (2) . . . . .	11
4	Gantt diagram (3) . . . . .	11
5	Hand-crafted and Deep Learning Features . . . . .	12
6	Examples of Low level features and Deep level features . . . . .	13
7	Examples and comparison of segmentation types . . . . .	14
8	Overview of some two-stage detection frameworks for object detection . . .	15
9	General outline of the project . . . . .	16
10	Metadata fields scraped from the original webpage . . . . .	17
11	Content based Image Retrieval block diagram . . . . .	18
12	Feature extraction from Deep Learning . . . . .	19
13	Database preprocessing block diagram . . . . .	21
14	Examples of images decorated and white background . . . . .	21
15	Test of the DeepCBIR code . . . . .	22
16	Structure of the output of the Image Retrieval block . . . . .	23
17	Confusion Matrix Image Retrieval . . . . .	24
18	Example Image Retrieval (1) . . . . .	25
19	Examples Segmentation . . . . .	26
20	Example of a section in the MOS . . . . .	27
21	Examples classifier's database . . . . .	28
22	Suggested learning rate . . . . .	29
23	Classifier's training and validation loss . . . . .	30
24	Confusion matrix of the classifier . . . . .	30
25	Suggested LR for master/individual problem . . . . .	31
26	Doubtful examples of Individual and Master beds . . . . .	31
27	User Interface: password . . . . .	32
28	User Interface: Image uploader . . . . .	33
29	User Interface: price selector . . . . .	33
30	User Interface: Classifier . . . . .	33
31	User Interface: Recommendations . . . . .	34
32	User Interface: Products . . . . .	34
33	User Interface: Product metadata . . . . .	35

### Listings

#### List of Tables

1	Summary of the MOS results. . . . .	27
2	Table with hyper parameters for the classifier's training. . . . .	29



---

3	Weights used for the classifier training. . . . .	29
4	Table of the Material costs . . . . .	36
5	Table of the Salaries . . . . .	36
6	Table Total costs . . . . .	36
7	WP1. . . . .	40
8	WP2. . . . .	40
9	WP3. . . . .	41
10	WP4. . . . .	41
11	WP5. . . . .	42
12	WP6. . . . .	42
13	WP7. . . . .	42
14	WP8. . . . .	43

# 1 Introduction

## 1.1 Project Overview and Statement of purpose

The project has been carried out at StageInHome, which is a startup company based in Barcelona, that by using Artificial Intelligence can decorate interior spaces with only one picture as an input image.

The pieces of furniture that the user can see in the StageInHome's decorated output are generated using generative models, meaning that they do not exist. So the user could not buy them if they liked some of the items.

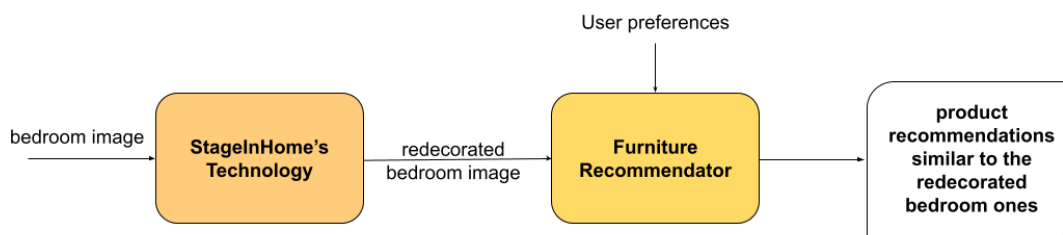


Figure 1: Diagram that presents how the technology of StageInHome and this project fit together.

The aim of this project is to build a furniture recommendor that uses the previous mentioned generated images as input. Then, using image processing techniques, AI as well as some user preferences, presents to the costumer similar pieces with some additional information and where they can buy them. A diagram can be seen in Figure 1.

The idea of this project was suggested by the company after working in the PAE subject, where StageInHome asked the group to build a furniture "Shazam". This was a system that aimed to identify different pieces of furniture in a room and tell the user where to buy or find them.

## 1.2 Objectives

In this section the main objectives of the project are shown.

- Develop a Web Scraping algorithm for a specific site.
- Analyze images with image processing techniques.
- Develop an Image Retrieval block that takes into account similarity and user inputs.
- Create a user interface to show results.
- Efficient and well-structured code.

### 1.3 Work Plan

The work plan has a total of 8 main work packages: Web Scraping, Filter Database, Image Retrieval, Segmentation, Classification, Join all blocks, User Interface and Documentation. These have been distributed in 4 months. The Gantt diagram is presented in Figure 2, Figure 3 and Figure 4.

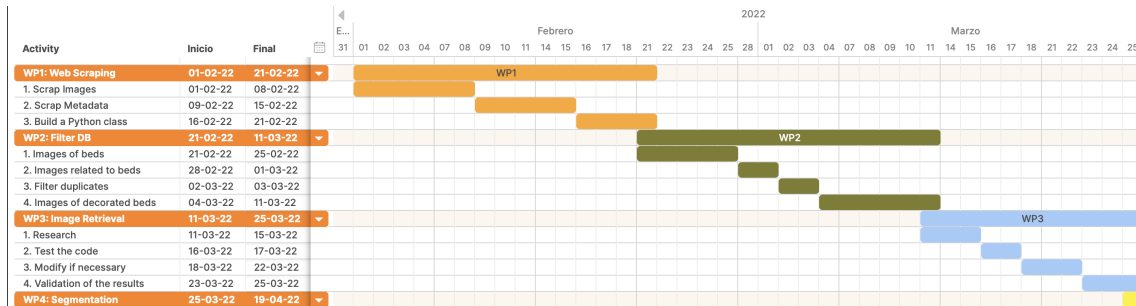


Figure 2: Gantt diagram of the project (1).

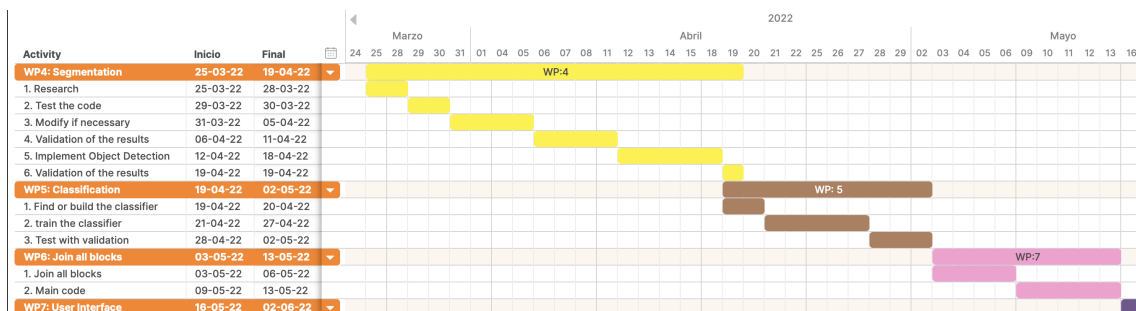


Figure 3: Gantt diagram of the project (2).

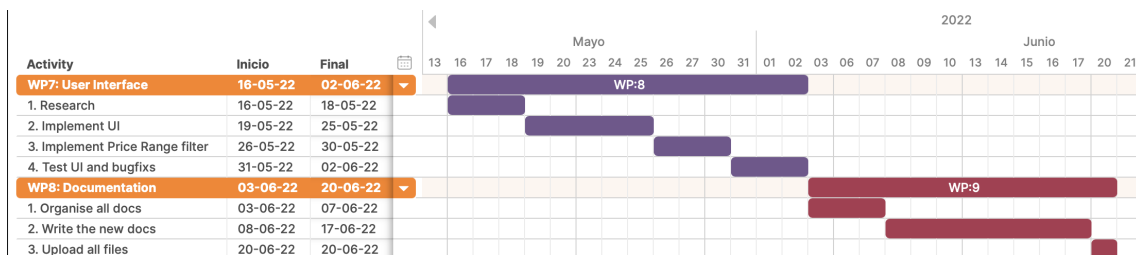


Figure 4: Gantt diagram of the project (3).

More about the work packages can be found in Appendice A.

## 2 State of the Art and Fundamentals

In this section there will be a review of the technologies that are being used in the Image Retrieval and in the segmentation fields. In addition we will explain the relevant context in order to make a better understanding of the next Methodology and Results.

### 2.1 Image Retrieval

The aim of Image Retrieval systems is to find similar images by searching and retrieving them from a large digital database [1].

With this purpose there are different approaches [2]:

- Text based Image Retrieval uses the text associated to an image to know what it contains. It allows to present the information as a textual query, and will find the relevant images based on the match between the textual query and the manual annotations of images [3].
- In Content Based Image Retrieval, the images are searched and retrieved depending on the similarity of their visual contents to a query image. For this features of the image are used with a feature extractor that is used to extract level features like color, texture and shape.
- Semantic Based Image Retrieval uses the semantic meaning of the images. This is one of the efforts that aims to close the semantic gap problem, which is the difference between the image representation and the needs of the user's information. There are two main approaches: Annotating images or image segments with keywords through automatic image annotation or adopting the semantic web initiatives.

From now on, we will focus on Content Based Image Retrieval since it seems the best option for our use case. It consists on four steps: data collection, build up a feature database, searching in the database and finally order the results of the retrieval by calculating the distance between the features of the query image and of the images in the database.

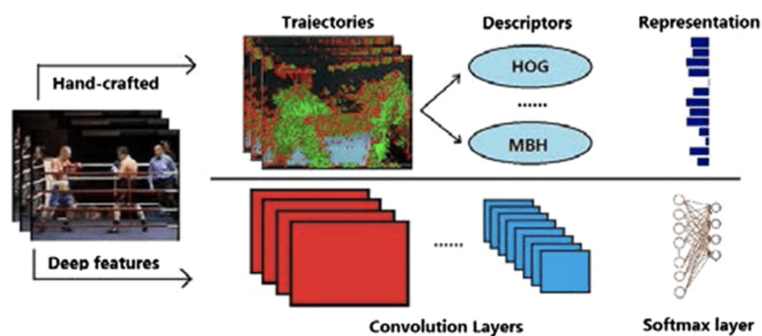


Figure 5: Hand-crafted and Deep Learning Features [4].

This features can be Hand-crafted features or Deep Level features (Figure 5). The hand-crafted features, are extracted by using information presented in the image itself with

Image Processing techniques such as HOG. The Deep Level features are obtained by deep learning using Neural Networks such as CNN.

In DeepCBIR [5], they propose to use deep learning features derived from a convolution network that has been trained for a large image classification problem. From it, a n-dimensional feature vector will be obtained.

The feature vector mentioned above comes from Convolutional Neural Networks (CNNs) which are a sequence of layers that have a hierarchy. They start with an input layer followed by one or multiple hidden layers and finally, they end in an output layer. The first layers are basically edge detectors and they extract low level features. As we go deeper in the architecture, the model represents deep level features that are extracted from the low-level ones (Figure 6). The features extracted in the layers of a CNN that is trained as an image classifier serves as good descriptor for image retrieval tasks.

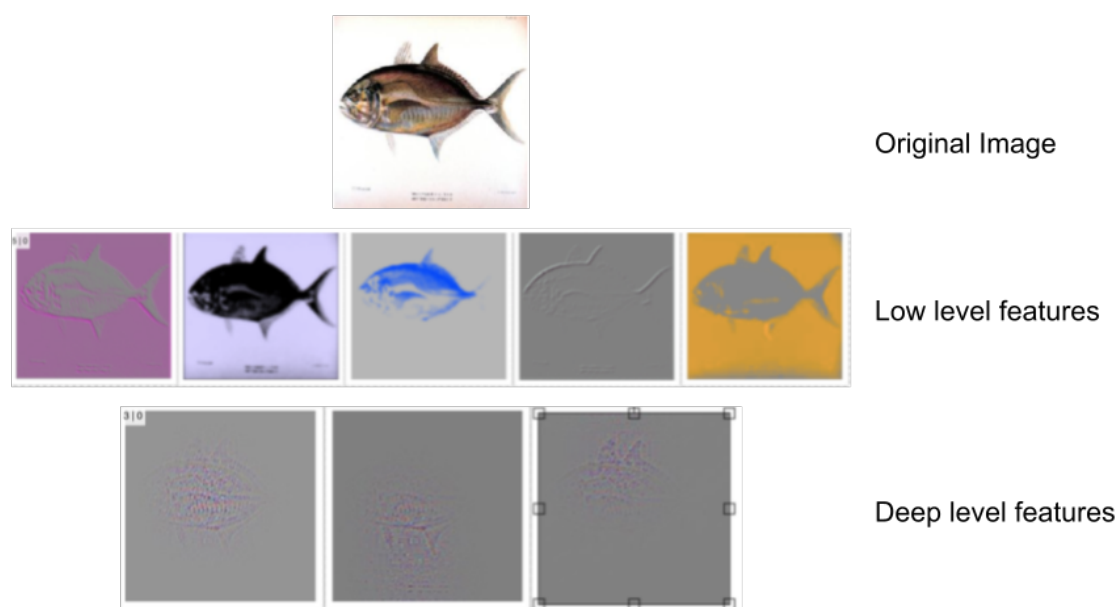


Figure 6: Examples of Low level features and Deep level features [5].

On the one hand, there are Global features which are global descriptors of an image and produce a single vector with values that describe aspects of the image such as shape, colour and texture. This are useful for classification purposes. The main advantage of this type of features is its ability for extraction and the low computational cost, but they often fail to take into account the visual content of the Image.

On the other hand, Local descriptors, are more effective for high level applications. They have a higher complexity and computational cost since the features have more dimensions in the space. Nowadays, research focuses on the use of Deep Learning techniques for the extraction of these features since they seem to outperform the traditional methods.

## 2.2 Segmentation

We can distinguish three main types of segmentation: Semantic Segmentation, Object Detection, and Instance Segmentation (Figure 7).

- Semantic Segmentation has the purpose to predict a label for each pixel of an Image. It does not distinguish between different objects of the same class.
- Object Detection recognizes object categories and it predicts the position of the object by a bounding box. So it does distinguish between different items that are included in the same class.
- Instance segmentation is a combination of the previous explained techniques and is able to identify different objects while it assigns the same label to objects that are the same class, and also separates them pixel by pixel with a mask.

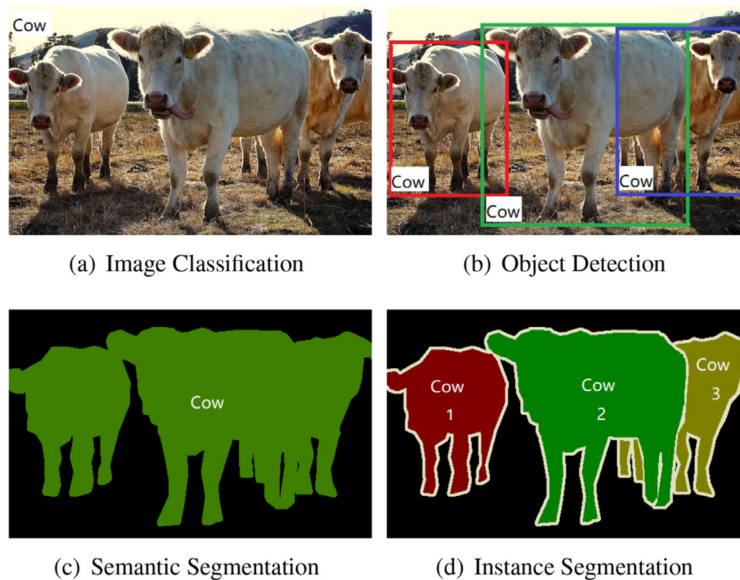


Figure 7: Examples and comparison of segmentation types [6].

So, for spatial detection there can be two different settings: On the one hand, a bounding box, which is a rectangle that localizes objects. And on the other hand, a pixel-mask that is more precise than the previous one and segments the objects by telling which pixels belong to the detected ones.

### 2.2.1 Object Detection

Object detectors can be divided in two categories:

- Two-Stage detectors, like R-CNN [7], first they get a proposal of regions from where to extract the features. Then, for each proposed region, a classifier predicts the category of each one of them.

- One stage detectors, such as YOLO [8], make predictions of objects on each location without the region scan step. Although the performance of the first ones is better, one-stage detectors are more time-efficient.

From now on we will focus on the Two-Stage detectors since are the ones that have been used in this project because in this user case the computation time is not that important as its performance. In Figure 8 are some of the architectures used .

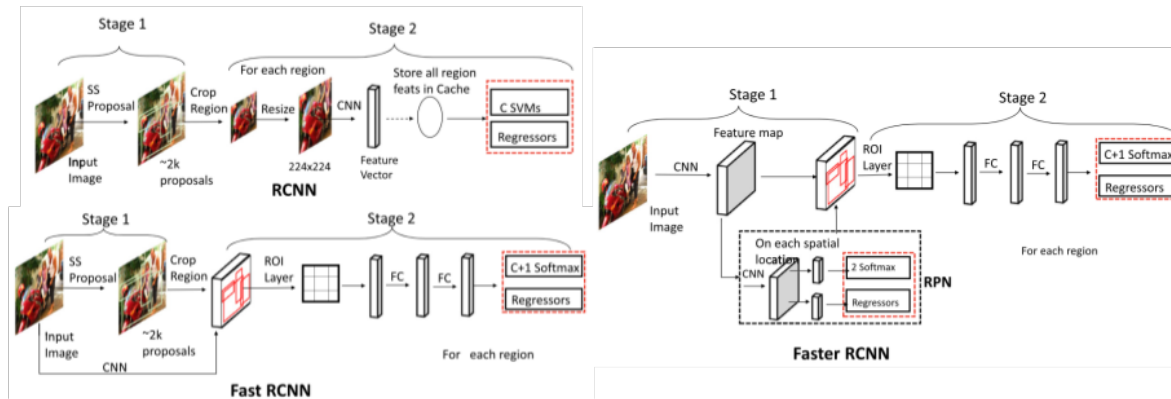


Figure 8: Overview of different two-stage detection frameworks for generic object detection [6].

R-CNN [7] is a pioneer neural network as two-stage object detector and its work flow has three main steps: Proposal generation of the regions, feature extraction and region classification. First it generates around 2.000 proposals (separating the regions that can easily be classified as background). Then each of these proposed regions is cropped and resized and is encoded into a feature vector by using a deep CNN. Next, this feature vector goes through a one-vs-one classifier. Finally, in order to make the bounding box fit the object, box regressors are used with extracted features as inputs.

The main drawbacks of this method is that the features of each proposal are extracted separately meaning that there are a lot of duplicated computations and it is very time consuming not only the training part, but also the testing. The three steps mentioned earlier are independent and can not be implemented as an end-to-end proposal. It also struggles to generate high quality proposals in complex situations.

Another convolutional neural network that has some improvements compared to the previous one is the Fast R-CNN [9]. It has fixed length regions and it computes a feature map for the whole image with fixed length region features. To extract these features a ROI Pooling layer is used and it only takes one value on  $N$  for the  $N \times N$  grid. Next, these feature vectors are fed into a sequence of fully connected layers that lead to a classification layer to generate the softmax probabilities and with it the label. Finally leading to a regression layer to have a more accurate bounding box. This approach can be optimized to an end to end and it achieves a better detection accuracy while having a better training and inference speed.

Finally, The Faster R-CNN [10] relies on a RPN, which is a fully convolutional network that takes an image of arbitrary size and generates a set of object proposals on each position. This network slides over the feature map using a  $m \times m$  sliding window and computing a feature vector for each position. This feature vector goes to an object classification layer which classifies each proposal as an object or background and to a bounding box regression layer. Although it can be optimized in an end-to-end manner, it is difficult to detect small objects since it uses a single deep layer feature map to make the final prediction.

### 3 Methodology

Figure 9 represents the structure of the project which has been divided in different blocks. At the end, they all together work to deliver to the user the final recommendation images and metadata as outputs.

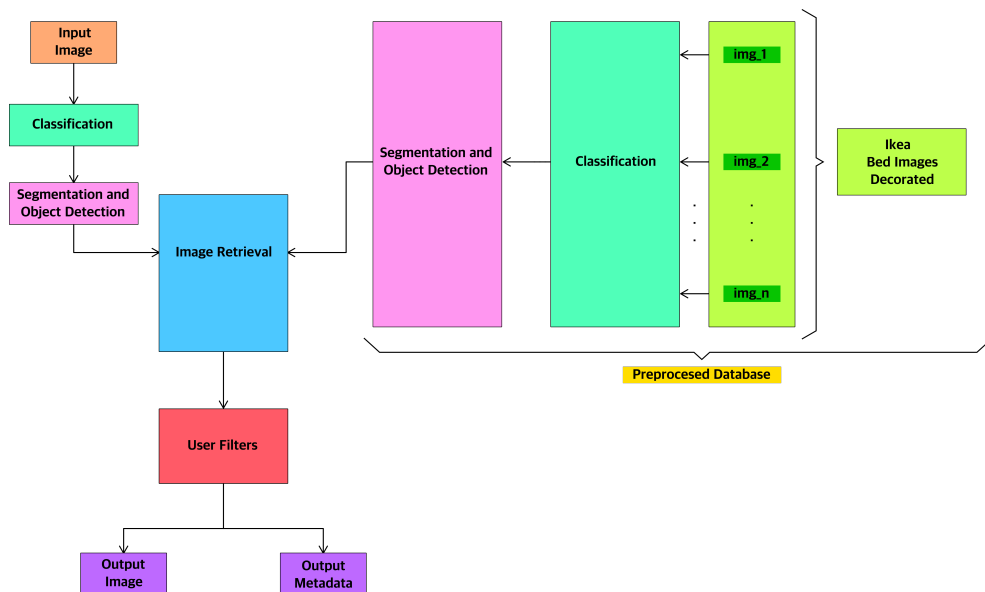


Figure 9: Outline of the project.

The work flow begins when the user uploads an image, generated by StageInHome, that will be classified according to its bed type (master, individual or cradle). Next, the image will go through the segmentation block where a bounding box will be computed and it will be used jointly with a preprocessed database to find similar images in the Image Retrieval block. After that, the user will be able to filter taking into account the price of the products. Finally the retrieved image will be displayed as well as all the related products within the price range with its information.

#### 3.1 Preprocessed Database

In order to build this recommendation system we firstly needed to build an image database



where the system searches given an image of a piece of furniture. The database does not only need to have all the possible recommended images, but also a way of linking them to products, and all its information as metadata.

Since there was a need of having a big database in order to be able to have better recommendations, Web scraping was done. Ikea's webpage is public and seemed a good option for this because it has a wide range of pieces of furniture and it is available almost worldwide.

Also, we needed information to display about the products and that was scraped too as metadata. For this we took information of 10 fields that can be seen in Figure 10:

1. name: Name of the product
2. short description: Short description of what the product is
3. price: Price in \$
4. long description: Description of some interesting characteristics of the product.
5. product id: Product id in the Ikea website
6. measurements: Dimensions of the product (if available)
7. rating: User rating out of 5.
8. product url: link of the Ikea's official website where the user can buy the product.
9. category: Category where Ikea has classified the product.
10. subcategory: Subcategory where Ikea has classified the product.

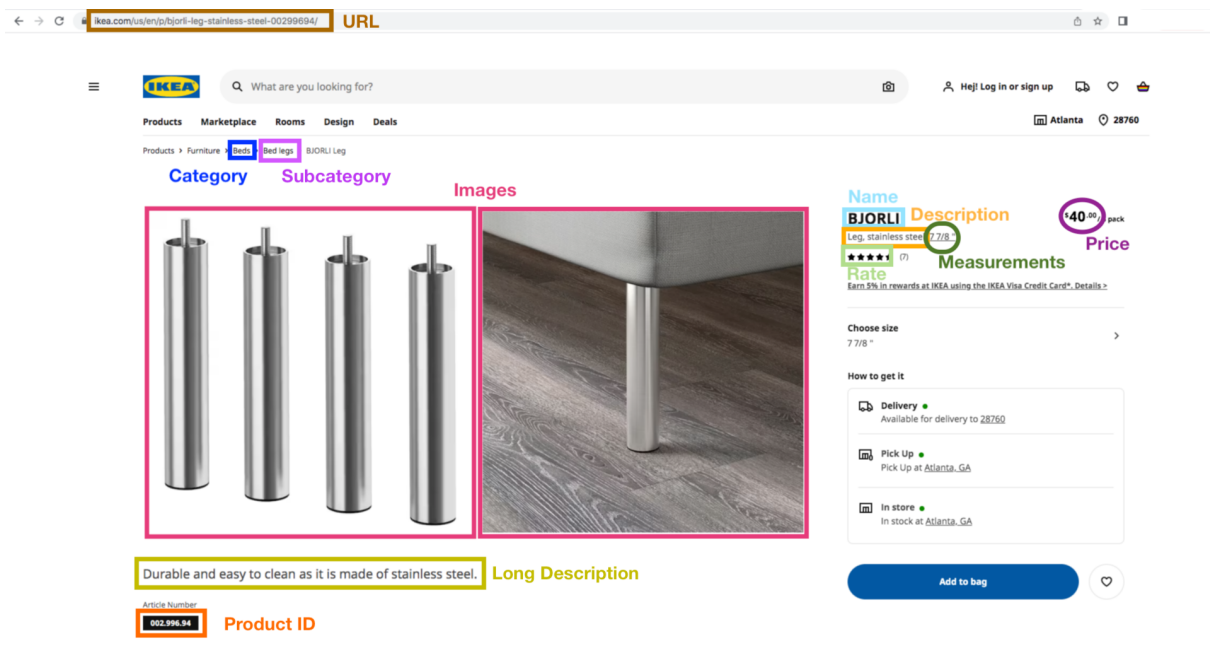


Figure 10: Metadata fields scraped from the original webpage.

These fields were chosen because of their relevant information for the user and for database processing purposes that will be addressed later on this document.

After all the web scraping there was a need to prepare the database of images that later would be used in the Image Retrieval. Nowadays StageInHome only works with bedrooms so we decided to just work with beds. Since the input image would be of an already decorated bedroom, the comparison needed to be made between both images being decorated bedrooms. Also there was a removal of duplicated images. Finally using other blocks from the code, a classification of types of bed and crops using bounding boxes were done.

### 3.2 Image Retrieval

This block consists of a Content Based Image Retrieval that needs an input image, the scope, which is the number of retrieved images that the system will give as an output, and a database of images. From here, there will be a feature extraction. In Figure 11 there is a diagram with the outline.

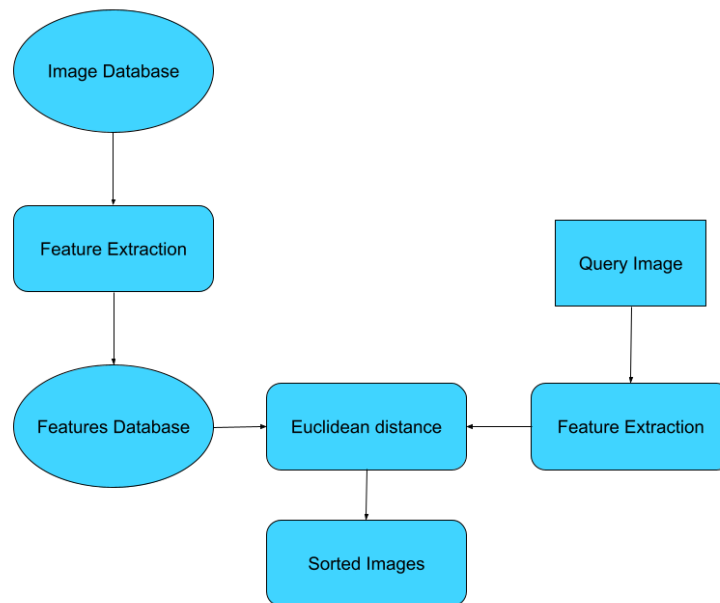


Figure 11: Content based Image Retrieval block diagram.

The approach used is the one presented in DeepCBIR [5]. For the feature extraction, they propose to use a pretrained classification model that removes the last SoftMax activation, which is the layer that calculates the final probabilities for each class, and keeps the previous fully connected layer. With this, it obtains all the deep level features information in a n-dimensional feature vector (Figure 12).

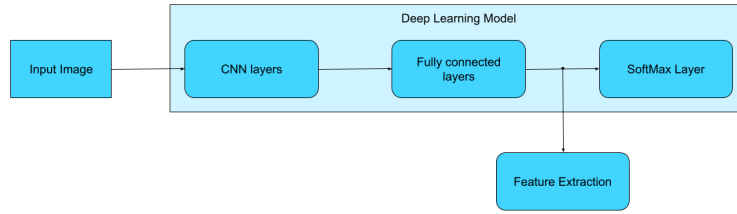


Figure 12: Feature Extraction from a Deep Learning model.

The classification model implemented is an Inception Resnet V2 model that is pretrained with ImageNet [11]. Finally in order to sort the images from the database by similarity, the Euclidean distance (1) between the features of the database and the ones of the input image is computed. The first retrieved image will be the one that minimizes this value.

$$D_{ij}^2 = \sum_{v=0}^n (x_{vi} - x_{vj})^2 \quad (1)$$

### 3.3 Segmentation and Object Detection

Taking into account that the plan is to compute the image retrieval of the bed, the background of both the input image and the database images is not important and it will interfere with the final result, a segmentation block was needed.

After doing a comparison between Instance segmentation and bounding boxes that will be further explained in the 4.3 of the Results, the decision was to use a Faster R-CNN (X101-FPN) [12] from Detectron2 [13], which was trained with COCO dataset [14]. This was the chosen one since the documentation said its box Average Precision is 43.0, the highest available in the object detection category in the Detectron2 Model Zoo. The model, detects all types of beds, including cradles with the same label.

### 3.4 Classification

The main reason for implementing this block was to improve the performance of the Image Retrieval block by first classifying the bed type. Depending on the output label, the Image retrieval will use a different database to compute recommendations. There are four possible labels: master bed, individual bed, cradle and others. If the label is 'others' it will ask for another image since it does not contain a bed. Otherwise, depending on the label, the used database will only contain, individual beds, master beds or cradles.

Another reason why this block was required is because, as explained in 3.3, the Object Detection model is pretrained and it has all the types of bed under the same label.

The final version of the classifier was obtained by fine tuning an already pre-trained architecture ResNet34 with ImageNet [11]. By doing this, there is a redirection of the training to the specific problem that there is a need to solve.

## 4 Results

The general work flow followed through the development of the whole project has been very similar. First the code is developed in a Jupyter Notebook to see if every part works properly. After that, the code goes to a sequential Python script where later will be arranged in functions and classes if needed. This methodology allows scalability of the code and makes it more efficient, not forgetting that the first objective is for it to work and a comprehensible code.

This section has been divided into the 5 main blocks worked during the project: Web scraping, Image Retrieval, Segmentation and Object Detection, Classification and User Interface. In each block, is explained the different implementations and the validations done as well as the results, conclusions and decisions made from them.

### 4.1 Preprocessed Database

#### 4.1.1 Web Scraping

This part has been developed from scratch and it has three main parts: The web scraping of the Ikea images, the product's metadata and the product thumbnail images. In all of them an API has been used to extract the data as well as Beautiful Soup [15], which is a Python library for pulling data out of HTML and XML files.

It should be noted that the web scraping code depends a lot on the web page that is being scraped, meaning that if it was needed to scrap more online stores most of the code would not be reused and that is why we needed a big one where we could extract almost everything. With this approach it was only needed the scraping of one web page.

First, the scraping of all the Ikea images with Beautiful Soup, which resulted in obtaining 99.316 images that we saved for processing later.

For the product's metadata the same library was used, and everything was saved in a table. This table has 24.996 rows meaning that there is metadata from 24.996 different products.

Finally, in order to show the product's thumbnail in the User Interface, a third web scraping needed to be done, in this case using the previous table of metadata with the column "product url", and identifying something distinctive in the HTML for the thumbnail image. We saved this images in a separate folder with the product id as a name. As a result, the thumbnail's folder contains 24.996 images and they correspond to the 24.996 products mentioned earlier.

Although everything was used and developed as separate blocks, in the final version of the code everything is inside a Python class where now all the steps can be done at once. Also some functions that could be useful for other scrapers or that were more general were separated to an utilities and helpers files and are imported in the class.

#### 4.1.2 Database preprocessing

First of all, there is a global diagram of the steps followed to process the database of

possible recommendations in Figure 13.

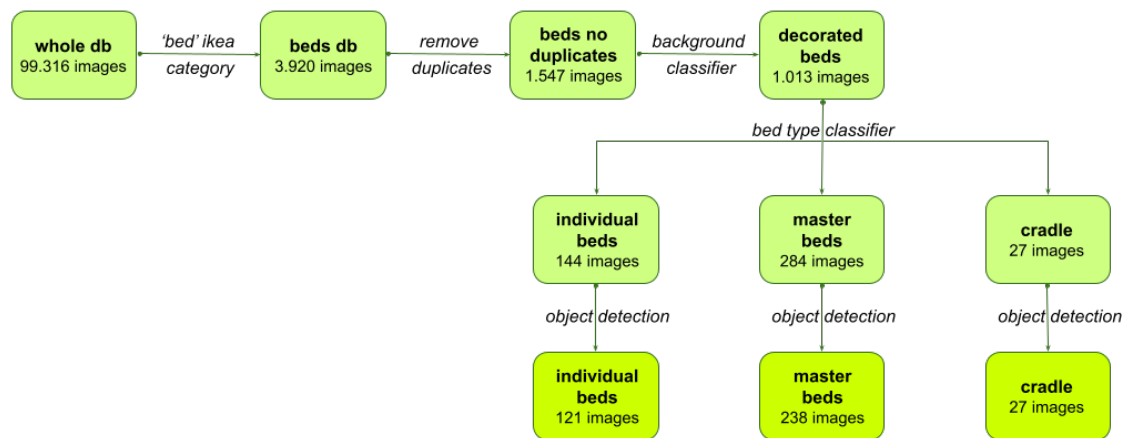


Figure 13: Database preprocessing block diagram.

To address the decision of only using beds, a classification of images was needed. For this a classifier with the following classes was used: "empty room, bedroom, kitchen, living room, bathroom and others". The plan was only to use the images labeled as bedrooms but after a manual data exploration, the conclusion was that it did not work properly with the images used. That is when the decision of adding the category and subcategory fields in the metadata came to mind. It was a certain way that there would not be any bed-related products missing. So by building a Jupyter Notebook to copy the images under the desired category into another folder the result ended up being 3.920 images under the Ikea's category "bed".

Another thing noted in the previous manual data exploration is that there were a lot of duplicated images and they could not be deleted without losing information since its name related them to the product. Here is where another table came to play. Using an already built script of detecting duplicates with some modifications and a new Jupyter Notebook to read the output of the previous one, a table with images information was created. This table only has two columns: "original image" and "product ids" being the first one the name of the file that will not be deleted and the last one, a list of product ids of the products that the image is linked to. After removing all the duplicates there were 1.547 images.

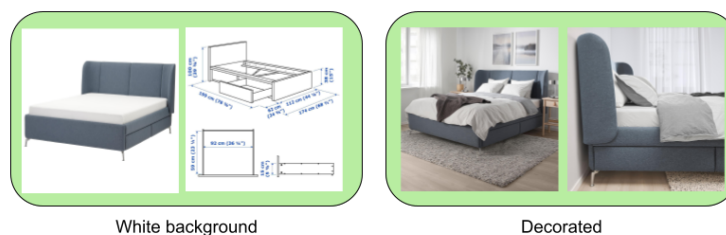


Figure 14: Examples decorated and white background images

Finally, there were two situations, decorated or white background and a classification of images taking into account its background was needed. For this purpose a small manual classification was done in order to set a threshold of white pixels. Above this threshold the image would be considered with white background and if it was under it the image was considered a decorated bedroom. This was done in a Jupyter Notebook and after setting the threshold in 3.000.000 pixels the images were separate in two different folders: decorated and white background. Finally a manual data exploration was done to correct the possible miss-classifications which ended having 1.013 images in the decorated folder and 534 images in the white background one. There are a couple of examples of these images in Figure 14

To sum up in this block the database ended up having 1.013 decorated bedroom images without duplicates. This images could be easily related to its products by the two-column table of images information, and also to this product's metadata using the table of metadata. The database will need more processing in the segmentation and the classification blocks.

## 4.2 Image Retrieval

As said earlier, this is a big important block of the project and it will determine the next steps. It was build from DeepCBIR [5] code adapting the output to what was needed. First some tests were done to check if everything worked as the paper said. It was very straight forward to test the proposed system and there was only the need to extract the features of our database. At this point our database was not fully preprocessed (1.013 images) but it was still good for testing.

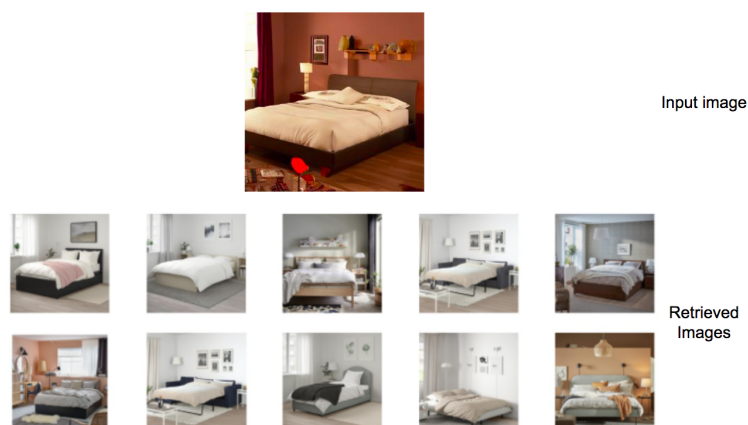


Figure 15: Test of the DeepCBIR code with a generated Image as an input.

The tests were like the one seen in Figure 15, and it seemed like it worked pretty well and that we could continue with this approach. These tests confirmed that the algorithm did what was needed for the project so the modifications began, we modified the algorithm while still using the CBIR class and built another class on top where the outputs were custom

made. We managed to not only retrieve the image path, but also all the information related to the products on that image.

The output is a list of retrieved images and its length will be the same as the scope defined as an input. Each item of this list will be a dictionary with two keys: "retrieved\_img\_path", which will be the path where we can find the retrieved image, and "products". This key will correspond to a list of products that are related to that retrieved image. Every item of this list is a dictionary that has all the metadata information (Figure 16).

```
[
  {
    "retrieved_img_path":
    "products": [
      {
        "name":
        "short_description":
        "price":
        "long_description":
        "product_id":
        "measurements":
        "rating":
        "product_url":
        "category":
        "subcategory":
      }
    ]
  }
]
```

Figure 16: Structure of the output of the Image Retrieval block.

In order to do more tests to be sure of how the algorithm worked, the functions of the feature extraction from the database were reviewed and some new were created. The extraction of features of the database only needs to be done once and it generates a .npy file that can be saved and then loaded when using the system. For this new versions of the functions, one consists on extracting, saving and loading the features of a database. Another, to extract, save and load the features of multiple databases at once. And finally one to just load the features of the database with the .npy file.

Despite doing all the tests some kind of validation was needed. The best option seemed to be a manual classification of the database and see if the retrieved images belonged to the same class as the input. For this a confusion matrix was used. A confusion matrix is a tool to compute validation results and allows visualization of the performance of an algorithm mostly in classification problems. Each column represents the actual value of the input and each row corresponds to the tested system's predicted class. This is a very visual way to see if a class is often being confused with another.



Figure 17: Confusion matrix of the Image retrieval block with whole images as the database and as input images with a scope of 5.

The Figure 17 is the confusion matrix of the Image Retrieval. It was not done correctly because the classes were not chosen right since some of the images could be classified as more than one label (individual, cradle, sofa-bed, wires, headboard, nothing). It had an accuracy of 42.5% which is not a good result and not conclusive because of the error in the classes. So this validation did not reflect what was intended to, but still we could come to some conclusions:

There was a need to implement a classifier for different types of beds: individual, master or cradle because since the CBIR was a pretrained model it was not picking this kind of details that were important for our purpose.

The other point was that there were a lot of images in the database that needed to be deleted such as the ones that had people in them and the ones classified as others, meaning that the image did not contain a complete decorated bed. After this database preprocessing, there were 455 images left.

There were still some questions remaining, such as which features were more important for the algorithm. With this purpose new tests were done as can be seen in Figure 18.





Figure 18: Example Image Retrieval using a whole image as an input and the whole images of the database

The first conclusion was that it worked properly because if the input image was in the database, the first retrieved image was itself. Second, the perspective of the image was more important than expected. Also that was more important the colour than the structure of the bed itself. All depending on what is in the database. There are no visible weights in the code so there is no control over this kind of preferences. Makes obvious the need of a classifier.

At this point the decision was to move on to the next block to make sure the work plan was followed and knowing that the following blocks would make the performance of this one to improve.

It should be pointed that the final version of this code is a Python class that needs the query image and the database to search retrieved images as inputs, as well as a scope that is the number of images that the algorithm will retrieve.

### 4.3 Segmentation and Object Detection

The first idea that was addressed was using a semantic segmentation model that already existed in the StageInHome code. So it just needed to be modified in order to make inference in just one class (bed). Some tests were made using whole images, bounding boxes and segmentation, but the last one was discarded because of the malfunctioning of the model (Figure 19). The bounding boxes were obtained by taking the left-most, right-most, top-most and down-most pixels.



Figure 19: Examples of the bad performance of the Instance Segmentation model.

Since the goal of this block is to make better the Image retrieval one, they needed to be tested together so the same was done with the database, and the Image retrieval was seeing:

- whole input image + whole database
- cropped bounding box of the input image + whole database
- cropped bounding box of the input image and cropped bounding box of the database

The first thing noted was that it made no sense to make comparisons between images that had not been treated the same way, meaning that if the database was cropped using a bounding box, the input should too. The main reason was that when Image retrieval uses as an input image one that already is in the database, the first retrieved image should be itself, and this did not happen if the same method was not used.

The thing was that this model of Instance Segmentation was not working properly since it was pretrained. The main problem was that the detection did not include the whole piece of furniture that is why bounding boxes were considered and the Image retrieval seemed to perform better. But still the segmentation was not good enough so there was a need to changing models.

The team was aware that the semantic segmentation models usually do not work perfectly, but as said earlier, the results with the bounding boxes were surprisingly good and to make sure that it was the best approach a MOS was done in order to know which type of segmentation worked better.

A MOS is a numerical way to determine human perspective of the quality of the system by doing a survey where the participants have to rate from one to five, in this case, the similarity (1 Bad, 2 Poor, 3 Fair, 4 Good, or 5 Excellent).

Since the instance segmentation model was not working properly, the segmentation Ground Truths of ADE20k [16] were used as input images and also as database images to make sure that the results of this survey were as objective as possible. The approaches compared in this test were input image using a cropped bounding box + database with a cropped bounding box and input image just the segmentation of the bed + database just the segmentation of the bed.

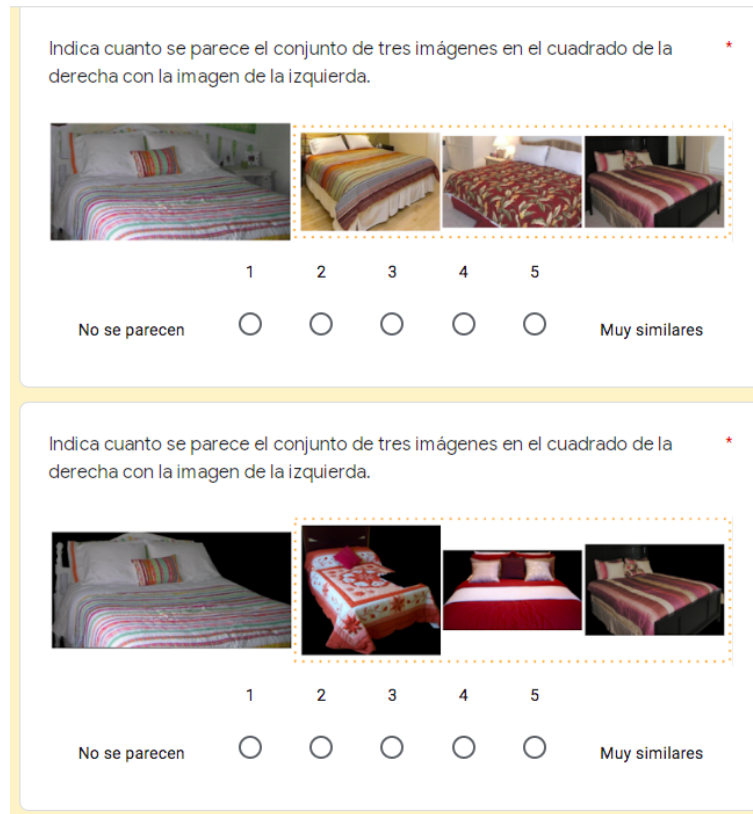


Figure 20: Example of a Section in the MOS.

For each section (Figure 20) there were two questions (using bounding box or semantic segmentation), so in total there were 8 questions. The only difference between them were the examples since the question was always the same: "Select how much do the three images inside the rectangle on the right look like the one in the left." In there is an actual example of a section.

	1st section	2nd section	3rd section	4th section	Total Mean
Bounding Box	4,24	4,44	3,96	3,72	<b>4,09</b>
Semantic Segmentation	3,24	3,56	3,08	2,72	<b>3,15</b>

Table 1: Summary of the MOS results.

The results, shown in Table 1, were clear and did not only confirm that the bounding boxes method worked better, but with a mean above 4, there was an assurance that the Image Retrieval block worked very well too, since the people were asked about similarity.

Taking into account the diagnostic, the decision was to use an Object Detection approach and to do a crop of the predicted bounding box with the most probability of being a bed. For this, Detectron2 [13] was used since it was tested in the PAE project and there was

a good certainty that it performed pretty well for furniture pieces with the pretrained blocks.

For the database part was very important to just keep one crop of each image so the decision was to use the bounding box with a higher score (probability). This methodology will be the same for the input image.

After implementing this block, the total number of images remaining in the database was 386 using a minimum threshold of detection of 0.5. There were some tests with a threshold of 0.9 and 0.7, but the number of images was reduced significantly (312 and 355), and the quality with the lower threshold was still very good since the best crops were kept.

## 4.4 Classification

In order to train this classifier, we first needed a training and validation database which was obtained by doing web scraping [17]. This was an already built scraper so we only needed to set the keywords needed. After this process we ended up with a total of 7.896 images (individual: 2.840 images, master: 2.799 images and cradle: 2.257 images). Then the Ikea images were added, and a duplicates removal and some manual classification were done ending up with a total of 3.683 images (individual: 1.135 images, master: 1.330 images and cradle: 1.218 images). The training and validation database were obtained by doing a 85% and 15% split of the previous one. Also a class "others" was added with 1200 random images from ImageNet [11]. An example of these classes can be seen in Figure 21.

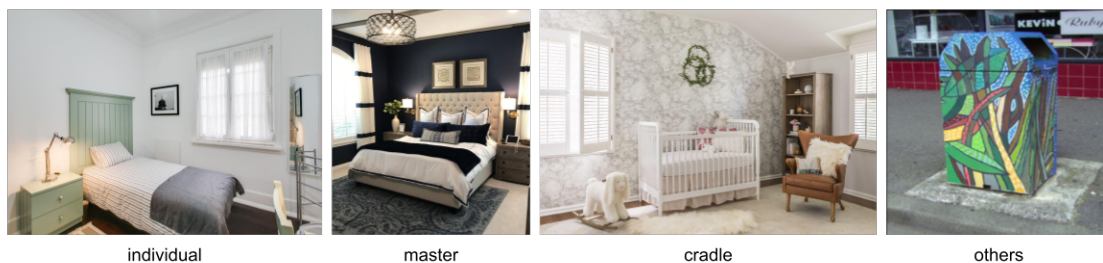


Figure 21: One example of each classifier's class.

This manual classification is the last step that needed the Ikea database before extracting its features, ending with 27 images of cradles, 144 images of individual beds and 282 images of master beds. With these images the 3 databases for the Image Retrieval algorithm were built.

After some tests, the best option seemed to be one with the "others" class, using weights (Table 3) and with the following hyper-parameters (Table 2):

Hyper parameters			
Training		Model	
Learning Rate	0.001202	Architecture	resnet34
Batch size	32	Threshold	0.7
Epochs	20		
Multi-classification	True		
Loss option	cross entropy		
Training type	Normal fit		

Table 2: Table with hyper parameters for the classifier’s training.

Weights	
Bed type	Weight
individual	0,7629
cradle	0,7671
master	0,6963
others	0,7737

Table 3: Weights used for the classifier training.

The learning rate was set using a functionality of fast.ai [18] and it computes it by searching the point with maximum slope and deducting a octave in the logarithmic space. In Figure 22 the dot is the suggested learning rate.

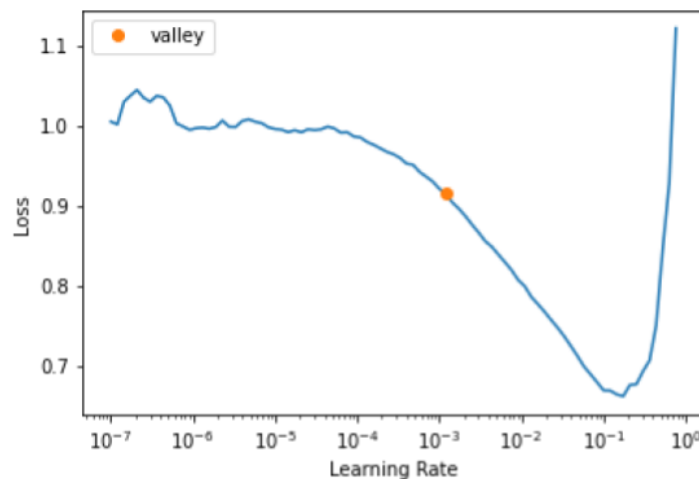


Figure 22: Suggested learning rate.

With all that, the result was a training with a validation and training loss that can be seen in Figure 23.

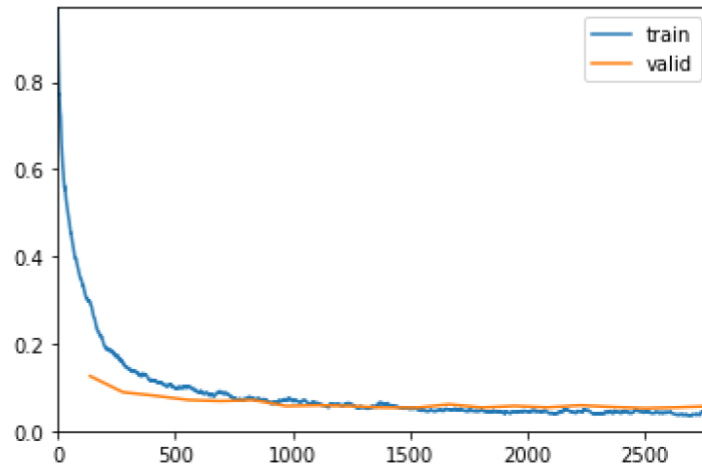


Figure 23: Classifier's training and validation loss.

Finally to analyse the performance of the classifier, a confusion matrix was used as well as its accuracy.

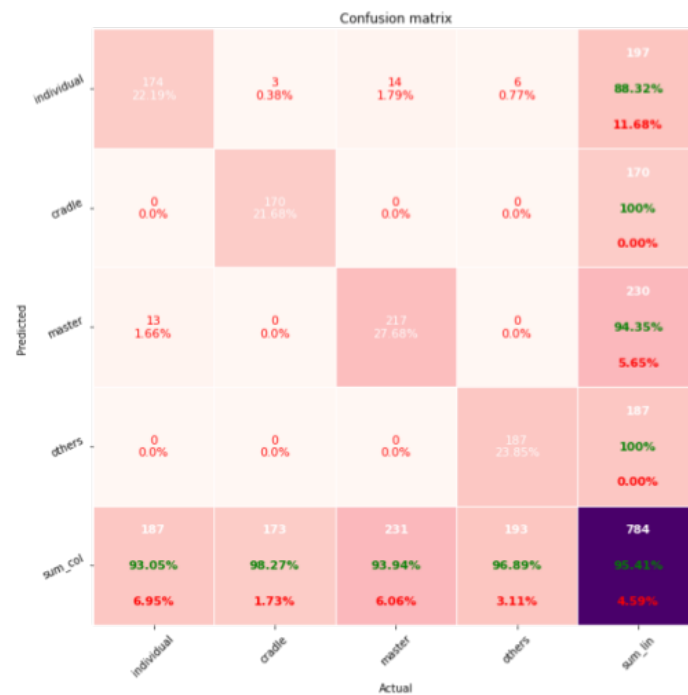


Figure 24: Confusion matrix of the classifier.

Although the accuracy is 95.41%, in the confusion matrix[19], shown in Figure 24, is clear that there is a problem with the individual and master classification since they are

confused very often. After more tests, it was not possible to make a better classifier with the model used.

Changing the database for a bigger one with only the labels master/individual, the LR graph shown in Figure 25 was obtained:

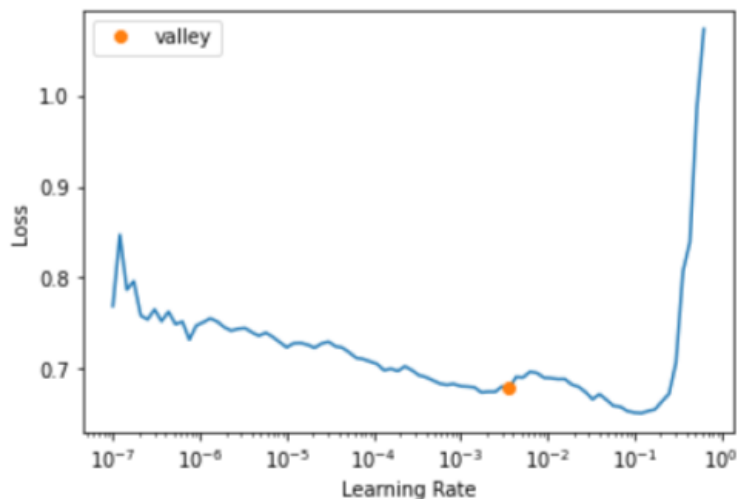


Figure 25: Suggested LR for the master individual problem.

In the case in Figure 25, as the minimum is at  $10e-3$  deducting an octave produces the suggested value to be in a positive gradient which does not make sense. This occurs as the algorithm does not check how the gradient is at the suggested value. Also, since it is very flat it means that the system will learn very slow and makes clear that is a very difficult problem to solve. Sometimes it was a challenge for the team to even classify the images that were used as database (Figure 26).



Figure 26: Doubtful examples of Individual and Master beds.

We came to the conclusion that it needs a big change of approach and model which was not possible due to the work plan. The solution for this was to add a new selector in the user interface that shows the label of the classifier and also lets the user to change it.

## 4.5 User Interface

The main goal of this block was to build a User Interface that was user friendly, and pleasant to the eyes. Also in this part there was implementation of the price filter, using the metadata extracted during the web scraping.

The price filter is the user filter that has been mentioned earlier and it consists on the user giving a price range, which will be taken into account when showing the products to the user. This will not condition the recommendations of the image retrieval, but still will make an impact in the products shown.

The user interface was developed using Streamlit [20] because it is a tool designed to build and share data apps coded as scripts in Python, the same programming language that has been used during the whole project. Although it has limited possibilities, it is very straight forward to develop a functional user interface. In addition, all the demos in StageInHome are done using it and in their code there are some useful tools that have been used in order to fit all the blocks explained above.



### Furniture Recommendor

*This DEMO offers actual recommendations based on your images*

**Password required.**

To access this demo you need a password. Please enter the password provided to you .

 0/20

Figure 27: User Interface: Password.

This demo is password protected, in Figure 27 there is an image that shows how it looks like. When the password is entered correctly, the demo starts by asking the user to upload an image (Figure 28).





## Furniture Recommendor

This Furniture recommendor only recommends beds.

Please upload a bedroom image:

Drag and drop file here  
Limit 200MB per file • PNG, JPG, JPEG

Browse files

Figure 28: User Interface: Image uploader.

Next it asks for the price range that can be selected with a range slider (Figure 29). After clicking the "Select price Range" button, it will appear the label detected by the classifier and the option to change it with a selector box (Figure 30).

Please select a price range (\$):

0 3000  
0 3000

Select price Range

Figure 29: User Interface: Price selector.

The type of bed detected is: **master**

Which type of bed would you like recommendations from?

Automatic classifier

Select type

Figure 30: User Interface: classifier.

After clicking the button "Select type" a new one with the message "Compute Recommendations" will appear and after pushing it, if the image was not a bedroom image or did not contain a master bed, individual bed or a cradle, an error message will appear

explaining the problem and to try with another image. Otherwise the system will show 6 image recommendations displayed in three columns under the title "Recommendations" like in Figure 31.

### Recommendations:

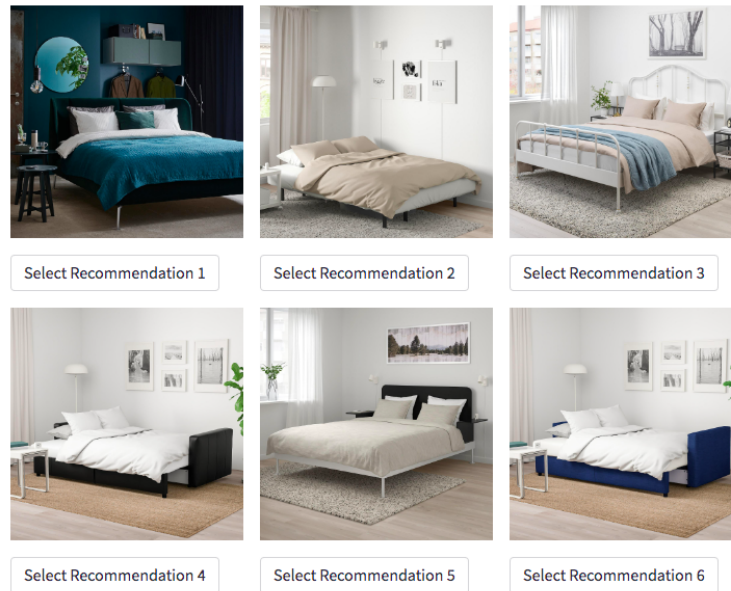


Figure 31: User Interface: Recommendations.

Under these images will be a select button with the number of the recommendation so the user can choose which one wants to know more about. After pressing one of these buttons all the products linked to the images will appear under a title that includes the number of the chosen recommendation and the number of products related to it (Figure 32).

### Recommendation 1 - 1 Products

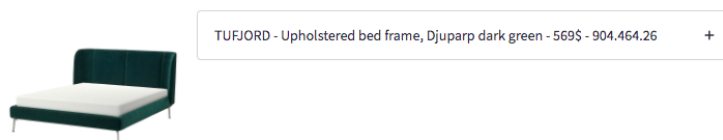



Figure 32: User Interface: Products.

This products will show its thumbnail and a small preview of its metadata only showing name, short description, price and product id. Then the user can choose which ones wants to know more by clicking the "+" and the metadata will expand showing all its fields, including the product URL where they can buy the product from (Figure 33).

## Recommendation 1 - 1 Products



TUFJORD - Upholstered bed frame, Djuparp dark green - 569\$ - 904.464.26 --

**Name:** TUFJORD

**Short Description:** Upholstered bed frame, Djuparp dark green

**Price:** 569

**Long Description:** TUFJORD will make you long for bedtime. The headboard's embracing curves help you to unwind, and make lazy mornings spent in bed even cozier. The soft velvet with a beautiful luster adds a luxurious feel.

**Product ID:** 904.464.26

**Measurements:** Queen

**User Rating:** Review: 4.2 out of 5 stars.

**Product URL:** <https://www.ikea.com/us/en/p/tufjord-upholstered-bed-frame-djuparp-dark-green-90446426/>

**Category:** Beds

**Subcategory:** Full, Queen and King size platform beds

Figure 33: User Interface: Product metadata.

In this case the user can view the metadata of more than one product at once, but this is not possible between different recommended images. It has to be noted that the system will only show products within the price range, if there is none, an error message will appear explaining that due to the price range there are no products and asking to change the price range or to choose another recommendation.

Some products have the same thumbnail image since it is the same bed with different measurements or with a subtle change of colour, but the decision was to keep it that way since in the Ikea website it is displayed like that. As a result, sometimes in the related products they all appear to be the same until the metadata is fully expanded. Also the retrieved recommendation images are 6 (scope) not only because the grid looks well, but also due to that it seemed the appropriate number for the user to have options to choose from and it was not too much information at the same time.

Finally, at the end of this project, StageInHome decided to upload this demo to an Amazon Web Services instance to make it public under their subdomain, meaning that everyone that has a password can try it in <http://demo.stageinhome.com:8510/recomendator>.

## 5 Budget

The approximate costs of this project take into account the materials and the salaries.

### 5.1 Material

There is a need of a powerful computer, its cost is around  $\frac{800 \cdot 0.9}{5} = 144$  per year assuming that the useful life is of 5 years. The computer has only been used for 4 months, which is a third of the year meaning  $\frac{144}{3} = 48$ . A server has also been used,  $\frac{3500 \cdot 0.9}{5} = 630$  per year,  $\frac{630}{3} = 210$  for 4 months (Table 4).

Material			
Description	Cost (€/unit)	useful life (years)	Total Cost (€)
Computer	800	5	48
Server	3.500	5	210

Table 4: Table of the Material costs

### 5.2 Salaries

Considering the salary of a junior engineer in Spain as 10 €/hour that has been working 640 hours in the project and taking into account the social security that the company has to pay as 33% (Table 5).

Salaries				
Description	Salary (€/hour)	Time dedicated (hours)	Social Security (€)	Total Cost (€)
Junior Engineer	10	640	2.112	8.512

Table 5: Table of the Salaries

### 5.3 Total Costs

The total cost of the project is 8.770 €. (6)

Total Costs	
Description	Cost (€)
Material cost	258
Salaries cost	8.512
<b>TOTAL COST :</b>	<b>8.770</b>

Table 6: Table Total costs

## 6 Conclusions

As a conclusion at the end of this project a bed recommendator has been built with its own database as well as with a detector, a classifier and an Image retrieval block. The system retrieves similar and relevant images to the user with all the information needed and also relies on a User interface that makes it easier for a user to interact with it.

The project ended having a database with 386 images separated in 3 categories by a classifier that has an accuracy of 95.41%. Also an Image retrieval block that together with an Object Detection block has a MOS of 4.09 up to 5. And with an easy to use User Interface as well as an structured code that allows scalability.

Although the database is very limited, not only in terms of quantity, but also in terms of variety and styles, the whole system is working properly and taking into account the validations that have been done, we can say that the different objectives of this project have been fulfilled.

To sum up the objectives accomplished, a Web Scraping algorithm has been developed for a specific site, in this case, Ikea. A large number of images have been analyzed using different techniques. An Image retrieval block that takes into account similarity has been correctly implemented and the user inputs are taken into account in the display section. A user friendly interface is provided for an easy interaction. It has an efficient and well-structured code, with its proper documentation in order to make it easy to understand for other developers.

## 7 Future Work

About the future work, in order to improve the performance of the whole system a new classifier with more capacity and able to choose between master and individual beds needs to be implemented. Another option could be to train or fine tuning the Object Detection block so in the same step the detection and the classification is done.

Improving the current database by adding new online stores would make possible to recommend a wider range of styles and could be followed by implementing another user filter, so they can choose the store.

Right now the recommendator only works for beds, but it would be useful to have bedding and to extend the possible recommendations to all bedroom furniture and even for other spaces. This is not going to be as difficult as it could seam due to the scalability of the project.

Finally this could be taken one step further, and substitute the piece of furniture recommended into the original image, in order to make possible for the user to conceive how it will look on their home and to increase the user experience.

## References

- [1] Artem Babenko and Victor Lempitsky. Deep convolutional features for image retrieval. *Expert Systems with Applications*, 177:114940, 2021.
- [2] K Shubhankar Reddy and K. Sreedhar. Image retrieval techniques: A survey. *International Journal of Electronics and Communication Engineering*, 9:19–27, 2016.
- [3] Meenakshi Pal and Sushil Garg. Image retrieval: A literature review. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, 2:2278–1323, 06 2012.
- [4] Zhongke Liao, Haifeng Hu, and Yichu Liu. Action recognition with multiple relative descriptors of trajectories. *Neural Processing Letters*, 51, 02 2020.
- [5] Subhadip Maji and Smarajit Bose. Cbir using features derived by deep learning. *ACM/IMS Trans. Data Sci.*, 2(3), aug 2021.
- [6] Xiongwei Wu, Doyen Sahoo, and Steven C.H. Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.
- [7] Trevor Darrell Ross Girshick, Jeff Donahue and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2013.
- [8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [12] Model zoo. [https://github.com/facebookresearch/detectron2/blob/main/MODEL\\_ZOO.md](https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md).
- [13] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [14] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.
- [15] beautiful soup. <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.

- 
- [16] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019.
- [17] Hardik Vasa. Google images download. <https://github.com/hardikvasa/google-images-download>, 2019.
- [18] Jeremy Howard et al. fastai. <https://github.com/fastai/fastai>, 2018.
- [19] Wagner Cipriano. Pretty print confusion matrix. <https://github.com/wcipriano/pretty-print-confusion-matrix>.
- [20] Streamlit. <https://streamlit.io/>.

# Appendices

## A Work Plan

This section contains all the tables of the work plan (Table 7 to Table 14).

Project: Web scraping	WP ref: WP1	
Major constituent: Constructiong a recommendation database	Sheet 1 of 8	
Short description:  Construct an Images and metadata database	Planned start date: 1/02 Planned end date: 21/02	
	Start event: 01/02 End event: 21/02	
Internal task T1: Scrap images: Create an algorithm to scrap all ikea product images.	Deliverables:	Dates:
Internal task T2: Scrap metadata: Create an algorithm to scrap all ikea product metadata.		
Internal task T3: Build a Python class: Build a Ikea_scraper Python class.		

Table 7: WP1.

Project: Filstering the database	WP ref: WP2	
Major constituent: Prepare the database	Sheet 2 of 8	
Short description:  Filter all the database into different folders	Planned start date: 21/02 Planned end date: 11/03	
	Start event: 21/02 End event: 11/03	
Internal task T1: Images ob beds: Separate all the images related only to beds	Deliverables:	Dates:
Internal task T2: Images related to beds: Separate all the images of products related to beds		
Internal task T3: Filter duplicates:Filter duplicates, building another .csv in order to not loose product information		
Internal task T4: Images of decorated beds: Get the images with decorated bedrooms		

Table 8: WP2.



Project: Image Retrieval	WP ref: WP3	
Major constituent: Algorithm to compute the recommendations	Sheet 3 of 8	
Short description: Have an algorithm that helps with the Image Retrieval of the database previously constructed.	Planned start date: 11/03 Planned end date: 25/03	
	Start event: 11/03 End event: 25/03	
Internal task T1: Research: Find a paper with code.	Deliverables:	Dates:
Internal task T2: Test the code: Test the code to see if it works with our database.		
Internal task T3: Modify if necessary: Modify the code if necessary.		
Internal task T4: Validation of the results: Validation of the results.		

Table 9: WP3.

Project: Segmentation Block	WP ref: WP4	
Major constituent: paper and code	Sheet 4 of 8	
Short description: Have an algorithm to segmentate the beds in all pictures (input image and all the possible outputs).	Planned start date: 25/03 Planned end date: 19/04	
	Start event: 25/03 End event: 19/03	
Internal task T1: Research: Find a paper with code, use the algorithm developed in the PAE project, or use the segmentator used in the company	Deliverables:  Critical review	Dates:  11/04
Internal task T2: Test the code: Test the code to see if it works with our database and how it performs with generated images.		
Internal task T3: Modify if necessary: Modify the code if necessary.		
Internal task T4: Validation of the results: Validation of the results.		
Internal task T5: Implement Object Detection: Implement Object detection block		
Internal task T6: Validation of the results: Validation of the results.		

Table 10: WP4.

Project: Classification block	WP ref: WP5	
Major constituent: code	Sheet 5 of 8	
Short description:	Planned start date: 19/04 Planned end date: 02/05	
Implement a bed classifier that distinguishes some types of beds.	Start event: 19/04 End event: 02/05	
Internal task T1: Find or build the classifier: Find a paper with code or build the classifier.	Deliverables:	Dates:
Internal task T2: Train the classifier: Train the classifier.		
Internal task T3: Test with validation: Validation of the results		

Table 11: WP5.

Project: Join all blocks	WP ref: WP6	
Major constituent: code	Sheet 6 of 8	
Short description:	Planned start date: 18/05 Planned end date: 26/05	
Build a main that uses all the previous blocks in the correct order	Start event: 03/05 End event: 13/05	
Internal task T1: Join all blocks: Put all the blocks together.	Deliverables:	Dates:
Internal task T2: Main code: Build a main code.		

Table 12: WP6.

Project: User Interface	WP ref: WP7	
Major constituent: Simulation	Sheet 7 of 8	
Short description:	Planned start date: 26/05 Planned end date: 06/06	
Develop a User Interface	Start event: 16/05 End event: 02/06	
Internal task T1: Research: Decide what will be used to develop the UI.	Deliverables:	Dates:
Internal task T2: Implement UI: Program the UI		
Internal task T3: Implement Price Range filter: Implement a product filter of price range		
Internal task T4: Test UI and bugfixs: Test the UI		

Table 13: WP7.

Project: Documentation	WP ref: WP8	
Major constituent: Documentation	Sheet 8 of 8	
Short description: Put together all the documentation of the project and write the final report.	Planned start date: 07/06 Planned end date: 21/06	
	Start event: 03/06 End event: 20/06	
Internal task T1: Organise all docs: Organise all the documentation already written	Deliverables:	Dates:
Internal task T2: Write the new docs: Write the documentation.		
Internal task T3: Upload all files: Upload all the necessary files.		Final thesis 20/06

Table 14: WP8.

---

## Abbreviations

**AI** Artificial Intelligence

**API** Application Programming Interface

**CBIR** Content Based Image Retrieval

**CNN** Convolutional Neural Network

**HOG** Histogram of Oriented Gradients

**HTML** Hyper Text Markup Language

**LR** Learning Rate

**MOS** Mean Opinion Score

**PAE** Projecte Avançat d'Enginyeria

**R-CNN** Region-based Convolutional Neural Network

**ROI** Region Of Interest

**RPN** Region Proposal Network

**URL** Uniform Resource Locator

**XML** eXtensible Markup Language

**YOLO** You Only Look Once