CS2310 Milestone 1

Sketch-based Image Retrieval

Yu Du

**Motivation**

With the popularity of compact digital cameras, taking photos is no longer a luxury hobby. With digital cameras, it is common for people to have thousands of pictures taken from daily life. This raises a problem?

How can we efficiently manage thousands of photos or more?

An ideal photo management system should be intelligent. So it can classify and organize photos automatically with little user assistance. And it also should have a good image retrieval user interface. With a good image retrieval interface, users can quickly find the photo that they want, instead of going through all the photos in the photo library.

A traditional way to solve the image retrieval problem is using keywords. For each photo, we tag keywords to each photo manually or automatically. When user wants to search photos, they type the keywords to search matched photos. A problem for keyword based searching is its inaccuracy. For photos has similar keywords, the keyword based algorithm lacks more information to further rank photos and put the best candidates into the head of the candidate list.

One of the better ways to search photo is sketch-based image retrieval. Instead of simply typing keywords, a sketch-based image retrieval system lets user quickly draw a sketch of the photo that he/she wants to search with a user-friendly GUI. With the sketch, user can easily express much more complex constraints on the photos than simple keywords. They can specify the graph elements that the photo contains, the spatial relations among the graph elements and the attributes of the graph elements like color and object type. After the user draws the sketch, the sketch-based image retrieval system can capture these complex constraints and compare these constraints with all the photos in the database to find matched photos. Each photo will get a fitness score which indicates how many percent of the complex constraints have been matched. Finally, the matched photo will be displayed based on the fitness score of each photo.

**GUI Design**

Figure 1 gives the conceptual GUI of a sketch-based image retrieval system. On the top, there is a tool bar which contains the functions like load/save sketches, express spatial constraints and a quick search bar for traditional keyword-based search. On the left, there is an icon bar which

contains the graph elements can be drawn on the canvas. The left icon bar is also interactive with the top keyword search bar. When the user types some keywords, matched graph elements is also dynamically added to the left icon bar. In the center of the GUI is the canvas, user can drag graph elements onto the canvas to proper location. On the right side, there is a property editor window, in which user can specify the attribute values of the selected graph elements.

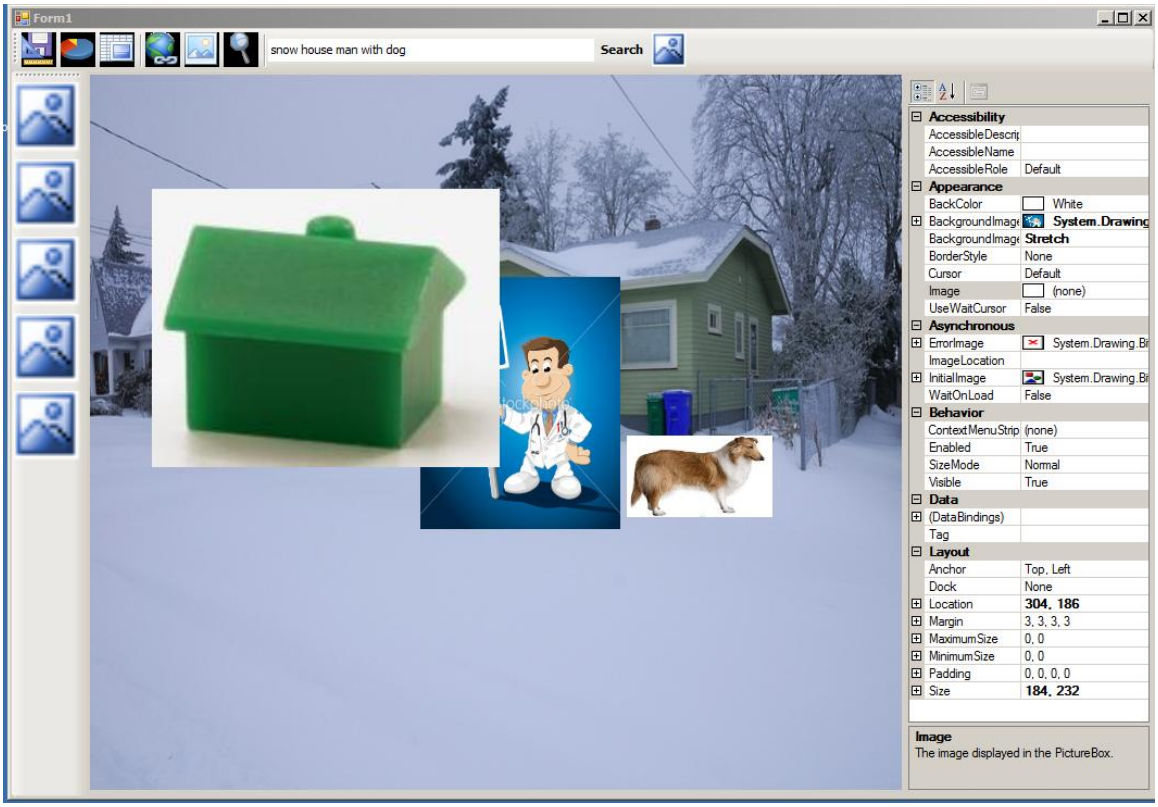


Figure 1: The conceptual GUI of a sketch-based image retrieval system

**Image Index File**

For a sketch, we can capture the graph elements and related constraints from the sketch. For a photo, we assume similar information can also be extracted by an intelligent image analysis module. For our experiment we will extract the information manually from the photos.

When the information is extracted from the sketch or the photos, we need to design an index file format to record the information in a formatted way. Later the image match algorithm can use the information to search best photo candidates. Because the image index information is semi-structure information. We use XML file to store the information. Another merit of XML format is we can extend the XML file format with new fields without break the file compatibility.

Figure 2 shows an example XML file for the image index information

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <elements>
    <element>
      <id>1</id>
      <boundbox>
        <left>100</left>
        <right>400</right>
        <top>30</top>
        <bottom>300</bottom>
      </boundbox>
      <type>person</type>
      <name>Mike</name>
      <has_glass>yes</has_glass>
      <has_hat>yes</has_hat>
    </element>
    <element>
      <id>2</id>
      <boundbox>
        <left>100</left>
        <right>400</right>
        <top>30</top>
        <bottom>300</bottom>
      </boundbox>
      <type>house</type>
      <!--attributes for house-->
      <roof_color>black</roof_color>
      <wall_color>red</wall_color>
    </element>
    <element>
      <id>3</id>
      <boundbox>
        <left>100</left>
        <right>400</right>
        <top>30</top>
        <bottom>300</bottom>
      </boundbox>
      <type>dog</type>
      <color>black</color>
    </element>
  </elements>
  <background>
    <keywords>
      <keyword>snow</keyword>
      <keyword>moutain</keyword>
    </keywords>
  </background>
  <query>
    <rank1>
      <contain>
        <element_id>1</element_id>
      </contain>
      <location>
        <element_id>1</element_id>
        <position>center</position>
```

```
      </location>
    </rank1>
    <rank2>
      <contain>
        <element_id>3</element_id>
      </contain>
    </rank2>
    <rank3>
      <relative_location>
        <src_element_id>1</src_element_id>
        <dest_element_id>1</dest_element_id>
        <direction>right</direction>
      </relative_location>
    </rank3>
    <rankn>
      <background>
        <keywords>
          <keyword>snow</keyword>
        </keywords>
      </background>
    </rankn>
  </query>
</root>
```

Figure 2: An example of image index XML file

The file contains three parts:

The elements section records all the graph elements on the photo/image.

The background section records the keywords for the background

The query section is only for stretch, which records the constraints and their priorities of the query.

For each graph element, it has several fields to describe the attribute values of the element. Different element type will have different attributes.

For the query, currently we support 4 types of query constraints:

Contain: indicate the element that the photo should contain

Location: indicate the location where the element should be located in the photo

Relative_location: indicate the relative location between two graph elements.

Background: indicate the keyword that the background should be contained.

Because constraints will have different priorities, we use rank tag to group constraints to different groups.

**Image Matching Algorithm**

For the initial version, we will scan the index file of all photos to get the candidate list.

For each photo, we will compare its image index file with the sketch's image index file based on the query information of the sketch. A score formula will be used to calculate the matchness between the sketch and the photo. The rank group of each query constraint will affect the weight the query constraint. After comparing with the index of all the photos, the photos will be ordered by their fitness score and Top-K photos will be displayed on the result page.