Journal of Computer and Electronic Sciences

Journal of Computer and Electronic Sciences
Available online at jces.blue-ap.org
©2015 JCES Journal. Vol. 1(1), pp. 11-21, 28 February, 2015
E-ISSN: 2311-6730

Detection and Removal of cracks and in Digitized Paintings

Sateesh V[1*] and David Solomon Raju Y[2]

1-PG Student, M. Tech, Dept of ECE Holy Mary Institute of Technology & Science, Keesara, R. R. Dt.- 501 301, Telangana
2-Assoc. Prof. ECE, Holy Mary Institute of Technology & Science, Keesara, R. R. Dt.- 501 301, Telangana

Corresponding Author: Sateesh V

Received: 09 January, 2015 *Accepted: 28 January, 2015* *Published: 28 February, 2015*

ABSTRACT

Crack detection and removal of cracks in digitized paintings is an integrated methodology where the cracks are detected by thresholding the output of the morphological top hat transform. The thin dark brush strokes which have been misidentified as cracks are removed using either MRBS function or a semi automatic procedure based on region growing. Cracks are filled by using the order statistics filters or controlled anisotropic diffusion is performed.

Keywords: *Anisotropic diffusion, detection of cracks, order statistics filters, top-hat transformation, virtual restoration of paintings.*
©2014 JAAS Journal All rights reserved.

INTRODUCTION

Many paintings, especially old ones, suffer from breaks in the substrate, the paint, or the varnish. These patterns are usually called cracks or craquelure and can be caused by aging, drying, and mechanical factors. Age cracks can result from non-uniform contraction in the canvas or wood-panel support of the painting, which stresses the layers of the painting. Drying cracks are usually caused by the evaporation of volatile paint components and the consequent shrinkage of the paint. Finally, mechanical cracks result from painting deformations due to external causes, e.g., vibrations and impacts.

The appearance of cracks on paintings deteriorates the perceived image quality. However, one can use digital image processing techniques to detect and eliminate the cracks on digitized paintings. Such a "virtual" restoration can provide clues to art historians, museum curators and the general public on how the painting would look like in its initial state, i.e., without the cracks. Furthermore, it can be used as a nondestructive tool for the planning of the actual restoration. A system that is capable of tracking and interpolating cracks. The user should manually select a point on each crack to be restored. A method for the detection of cracks using multi-oriented Gabor filters. Crack detection and removal bears certain similarities with methods proposed for the detection and removal of scratches and other artifacts from motion picture films. However, such methods rely on information obtained over several adjacent frames for both artifact detection and filling and, thus, are not directly applicable in the case of painting cracks. Other research areas that are closely related to crack removal include image in painting which deals with the reconstruction of missing or damaged image areas by filling in information from the neighboring areas, and disocclusion, i.e., recovery of object parts that are hidden behind other objects within an image. Methods developed in these areas assume that the regions where information has to be filled in are known. Different approaches for interpolating information in structured and textured image areas have been developed. The former are usually based on partial differential equations (PDEs) and on the calculus of variations whereas the latter rely on texture synthesis principles. A technique that decomposes the image to textured and structured areas and uses appropriate interpolation techniques depending on the area where the missing information. The results obtained by these techniques are very good. A methodology for the restoration of cracks on digitized paintings, which adapts and integrates a number of image processing and analysis tools is proposed in this paper. The methodology is an extension of the crack removal framework. The technique consists of the following stages:

- Crack detection;
- Separation of the thin dark brush strokes, which have been misidentified as cracks;
- Crack filling (interpolation).

A certain degree of user interaction, most notably in the crack-detection stage, is required for optimal results. User interaction is rather unavoidable since the large variations observed in the typology of cracks would lead any fully automatic algorithm to failure. However, all processing steps can be executed in real time, and, thus, the user can instantly observe the effect of parameter tuning on the image under study and select in an intuitive way the values that achieve the optimal visual result. Needless to say, only subjective optimality criteria can be used in this case since no ground truth data are available. The opinion of restoration experts that inspected the virtually restored images was very positive. Two methods for the separation of the brush strokes which have been falsely identified as cracks. Methods for filling the cracks with image content from neighboring pixels.

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

The existing methods for processing digital images are there which actually deal with enhancing the image picture quality, brightness, color etc. These factors can be degraded due to aging process. Such a image processing technique algorithm concentrates on improving those factor alone. There are not designed to analysis and improve in the cracks region. The cracks removal has to be rectified in the different manner. The principle applied to improve image color, brightness and other characteristic can not be used for crack detection and removal. This project concentrates on the digital image processing algorithm that deals only with crack detection and removal

2.2 PROPOSED SYSTEM

The proposed system deals with digital image processing techniques that detects and remove the cracks in the images. A system that is capable of tracking and interpolating cracks. The user should manually select a point on each crack to be restored. A method for the detection of cracks using multi-oriented Gabor filters. Crack detection and removal bears certain similarities with methods proposed for the detection and removal of scratches and other artifacts from motion picture films. However, such methods rely on information obtained over several adjacent frames for both artifact detection and filling and, thus, are not directly applicable in the case of painting cracks. Other research areas that are closely related to crack removal include image in painting which deals with the reconstruction of missing or damaged image areas by filling in information from the neighboring areas, and disocclusion, i.e., recovery of object parts that are hidden behind other objects within an image. Methods developed in these areas assume that the regions where information has to be filled in are known. Different approaches for interpolating information in structured and textured image areas have been developed. The former are usually based on partial differential equations (PDEs) and on the calculus of variations whereas the latter rely on texture synthesis principles. A technique that decomposes the image to textured and structured areas and uses appropriate interpolation techniques depending on the area where the missing information. The results obtained by these techniques are very good. A methodology for the restoration of cracks on digitized paintings, which adapts and integrates a number of image processing and analysis tools is proposed in this paper. The methodology is an extension of the crack removal framework. The technique consists of the following stages:

- There should be some method through which crack area in the digital image can be detection;
- Separation of the thin dark brush strokes, which have been misidentified as cracks;
- Crack filling (interpolation).

A certain degree of user interaction, most notably in the crack-detection stage, is required for optimal results.

2.2.1 DETECTION OF CRACKS

Cracks usually have low luminance and, thus, can be considered as local intensity minima with rather elongated structural characteristics. Therefore, a crack detector can be applied on the luminance component of an image and should be able to identify such minima. A crack-detection procedure based on the so-called top-hat transform is proposed in this paper. The top-hat transform is a grayscale morphological filter defined as follows:

$$Y(x) = f(x) - f_{nB}(x) \qquad (1)$$

where $f_{nB}(x)$ is the opening of the function f(x) (in our case, the luminance component of the image under study) with the structuring set , defined as

$$nB = B\Phi B\Phi \ldots \Phi B(\text{n times}) \quad (2)$$

In the previous equation, Φ denotes the dilation operation. A square or a circle can be used as structuring element B. The final structuring set nB is evaluated only once using and is used subsequently in the opening operation of (1). The opening f_{nB} of a function is a low-pass nonlinear filters that erases all peaks (local maxima) in which the structuring element nB cannot fit. Thus, the image $f - f_{nB}$ contains only those peaks and no background at all. Since cracks are local minima rather than local maxima, the top-hat transform should be applied on the negated luminance image. Alternatively, one can detect cracks by performing closing on the original image f(x) with the structuring set nB and then subtracting from the result of closing $f^{nB}(x)$

$$Y(x) = f^{nB}(x) -- f(x) \qquad \text{--------} \ (3)$$

It can be easily shown that the result of (3) is identical to that of applying (1) on the negated image. Use of (3) does not require negation of f(x) which grand it a small but not negligible computational advantage over (1).
In situations where the crack-like artifacts are of high luminance, as in the case of scratches on photographs, negation of the luminance component prior to the crack detection is not required, i.e., the crack detection procedure can be applied directly on the luminance image. The user can control the result of the crack-detection procedure by choosing appropriate values for the following parameters:

- The type of the structuring element B
- The size of the structuring element B and the number n of dilations in (2).

These parameters affect the size of the "final" structuring element nB and must be chosen according to the thickness of the cracks to be detected. It should be noted, however, that these parameters are not very critical for the algorithm performance due to the thresholding operation that will be described in the next paragraph and also due to the existence of the brush-stroke/crack separation procedure, which is able to remove crack-like brush strokes that have been erroneously identified as cracks. The fact that all the results presented in this paper have been obtained with the same top-hat transform parameters comes as a clear indication that the above statement is indeed true. These parameters were the following:

- Structuring element type: square;
- Structuring element size: 3 X 3;
- Number n of dilations in (2) : 2.

The top-hat transform generates a grayscale output image t (k, l) where pixels with a large grey value are potential crack or crack-like elements. Therefore, a thresholding operation on t (k, l) is required to separate cracks from the rest of the image. The threshold can be chosen by a trial and error procedure, i.e., by inspecting its effect on the resulting crack map. The low computational complexity of the thresholding operation enables the user to view the crack-detection results in real time while changing the threshold value, e.g., by moving a slider. This fact makes interactive threshold selection very effective and intuitive. Alternatively, threshold selection can be done by inspecting the histogram of for a lobe close to the maximum intensity value (which will most probably correspond to Crack or crack-like pixels), and assigning it a value that separates this lobe from the rest of the intensities. The result of the thresholding is a binary image marking the possible crack locations. Instead of this global thresholding technique, more complex thresholding schemes, which use a spatially varying threshold, can be used. Obviously, as the threshold value increases the number of image pixels that are identified as cracks decreases. Thus, certain cracks, especially in dark image areas where the local minimum condition may not be satisfied, can remain undetected. In principle, it is more preferable to select the threshold so that some cracks remain undetected than to choose a threshold that would result in the detection of all cracks but will also falsely identify as cracks, and subsequently modify, other image structures. The thresholded (binary) output of the top-hat transform on the luminance component of an image containing cracks (Fig. 1) can be seen in Fig. 2.

2.2.2 SEPARATION OF THE BRUSH STROKES FROM THE CRACKS

In some paintings, certain areas exist where brush strokes have almost the same thickness and luminance features as cracks. The hair of a person in a portrait could be such an area. Therefore, the top-hat transform might misclassify these dark brush strokes as cracks. Thus, in order to avoid any undesirable alterations to the original image, it is very important to separate these brush strokes from the actual cracks, before the implementation of the crack filling procedure. Two methods to achieve this goal are described in the following subsections.

A. Semi-Automatic Crack Separation

A simple interactive approach for the separation of cracks from brush strokes is to apply a region growing algorithm on the thresholded output of the top-hat transform, starting from pixels (seeds) on the actual cracks. The pixels are chosen by the user in an interactive mode. At least one seed per connected crack element should be chosen. Alternatively, the user can choose to apply the technique on the brush strokes, if this is more convenient. The growth mechanism that was used implements the Well-known grassfire algorithm that checks recursively for unclassified pixels with value 1 in the 8-neighborhood of each crack pixel. At the end of this procedure, the pixels in the binary image, which correspond to brush, strokes that are not 8-connected to cracks

will be removed. The above procedure can be used either in a stand-alone mode or applied on the output of the MRBF separation procedure described in the next section to eliminate any remaining brush strokes.



Figure 1. Original painting

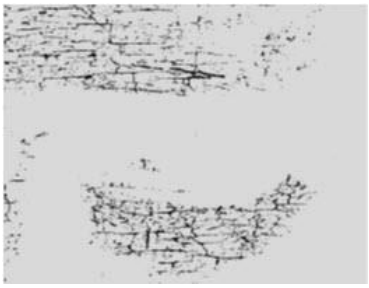


Figure 2. Thresholded output of the top-hat transform (threshold value: 23)

In our implementation, a MRBF network with two outputs was used. The first output represents the class of cracks while the second one the class of brush strokes. Input vectors were two-dimensional and consisted of the hue and saturation values of pixels identified as cracks by the top-hat transform. The number of clusters (hidden units) chosen for each class depends on the overlap between the populations of cracks and brush strokes. If there is a substantial overlap, the number should be increased, in order to reduce the classification error. In our implementation three hidden units have been incorporated. Training was carried out by presenting the network with hue and saturation values for pixels corresponding to cracks and crack-like brush strokes. Data from 24 digitized portable religious icons from the Byzantine era were used for this purpose.

The system trained using this specific training set can be considered to be optimized for paintings of this style and its use on paintings of other style might result in somewhat suboptimal results. However, appropriately selected training sets can be used to train the system to separate cracks from brush strokes on paintings of different artistic styles or content. In order to select pixels corresponding to cracks and crack-like brush strokes the crack detection algorithm presented was applied on these images. Results were subsequently post processed by an expert using the semi-automatic approach presented in Section III. The aim of this post processing step was twofold: to remove pixels that are neither cracks nor crack-like brush strokes and to separate cracks and crack-like brush strokes for the supervised step of the training procedure. In this supervised training step, the network was presented with these labeled inputs, i.e., pairs



Figure 3. Original painting

Of hue-saturation values that corresponded to image pixels that have been identified as belonging to cracks and crack-like brush strokes. After the training session, the MRBF neural network was able to classify pixels identified as cracks by the top-hat transform to cracks or brush strokes. The trained network has been tested on 12 images from the training set and 15 images (of the same artistic style and era) that did not belong to the training set. A thresholded top-hat transform output containing many brush strokes, e.g., hair is illustrated in Fig.4. A great part of these brush strokes is separated by the MRFB

2.2.3 CRACK-FILLING METHODS

After identifying cracks and separating misclassified brush strokes, the final task is to restore the image using local image information (i.e., information from neighboring pixels) to fill (interpolate) the cracks. Two classes of techniques, utilizing order Statistics filtering and anisotropic diffusion are proposed for this purpose. Both are implemented on each RGB channel independently and affect only those pixels which belong to cracks. Therefore, provided that the identified crack pixels are indeed a crack pixel, the filling procedure does not affect the "useful" content of the image. Image in painting techniques like the ones cited in Section I can also be used for crack filling. The performance of the crack filling methods presented below was judged by visual inspection of the results. Obviously, measuring the performance of these methods in an objective way is infeasible since ground truth data (e.g., images depicting the Paintings in perfect condition, i.e., without cracks) are not available. For the evaluation of the results, two restoration experts were asked to inspect several images restored using the various methods and comment, based on their experience, on the quality of the filling results, (i.e., whether cracks were sufficiently filled), whether the color used for filling was the correct one, whether fine image details were retained, etc.



Figure 4.Crack filling by using the modified trimmed mean filter (filter size 5 2 5)

3. DESCRIPTION OF THE PROBLEM

3.1 Existing System

The existing tools such as PhotoShop involve some simple options like blur. In that only a rough pixel transformations will be there. There is no exact crack detection and filling procedure in existing system.

3.2 Proposed System

The proposed system involves exact crack detection and filling procedure. It involves top-hat transformation, region-growing algorithm (grassfire algorithm) and median filter procedures.

4. SYSTEM ANALYSIS

4.1 System Description

This project is entitled as "Detection and removal of cracks and in digitized paintings". The main objective of the project is to remove and detect the cracks. It contains various modules included in the project as follows:

Input Module:

We are used to give the input image (cracked image) from this module.

Gray scale conversion module:

If the image is color image then we have to convert into the common color format like a gray colored image. This work will done the use of gray scale algorithm.

Cracks Detection module:

This is our third module. This module used to find the cracks in the cracked image with the use of surrounded pixels.

Crack filling module:

This module used to fill the color by using median filter and cracks removal algorithm. The cracks will be filled by the surrounded pixel color.

Output module:

The output will produce by this module. Each and every changes in this project will be displayed from the separate forms.

4.2 UML DiagramS: USE CASE MODEL

Use case diagram for our system:

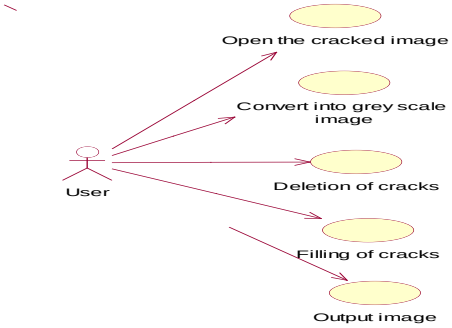


Figure 5. Use case diagram

ACTIVITY DIAGRAM

Activity diagrams provide a way to model the workflow of a business process, code-specific information such as a class operation. The transitions are implicitly triggered by completion of the actions in the source activities. The main difference between activity diagrams and state charts is activity diagrams are activity centric, while state charts are state centric. An activity diagram is typically used for modeling the sequence of activities in a process, whereas a state chart is better suited to model the discrete stages of an object's lifetime.

An activity represents the performance of task or duty in a workflow. It may also represent the execution of a statement in a procedure. You can share activities between state machines. However, transitions cannot be shared.

An action is described as a "task" that takes place while inside a state or activity

Actions on activities can occur at one of four times:

on entry---The "task" must be performed when the object enters the state or activity.

on exit---The "task" must be performed when the object exits the state or activity.

do---The "task" must be performed while in the state or activity and must continue until exiting the state.

on event---The "task" triggers an action only if a specific event is received.

The following tools are used on the activity diagram toolbox to model activity diagram

An end state represents a final or terminal state on an activity diagram or state chart diagram.

A start state (also called an "initial state") explicitly shows the beginning of a workflow on an activity diagram.

Swim lanes can represent organizational units or roles within a business model. They are very similar to an object. They are used to determine which unit is responsible for carrying out the specific activity. They show ownership or responsibility. Transitions cross swim lanes

- Synchronizations enable you to see a simultaneous workflow in an activity diagram Synchronizations visually define forks and joins representing parallel workflow.
- A fork construct is used to model a single flow of control that divides into two or more separate, but simultaneous flows. A corresponding join should ideally accompany every fork that appears on an activity diagram. A join consists of two of more flows of control that unite into a single flow of control. All model elements (such as activities and states) that appear between a fork and join must complete before the flow of controls can unite into one.
- An object flow on an activity diagram represents the relationship between an activity and the object that creates it (as an output) or uses it (as an input).

Activity diagram for our system:

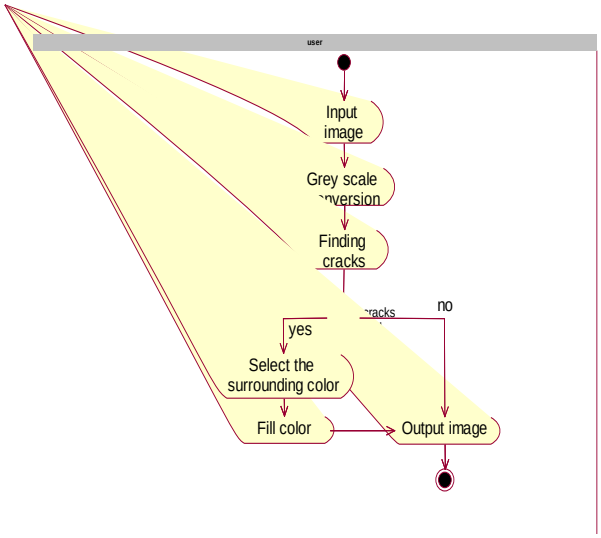


Figure 6. Activity Diagram

5. SYSTEM DESIGN

5.1 SEQUENCE DIAGRAM

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Elements of Sequence Diagrams:

There are mainly five elements in sequence diagrams. Three of them are common to the two interaction diagrams and two are for sequential diagrams. They are -

An object is a concrete manifestation of a class to which a set of operations can be applied and which has a state that stores the effects of the operations. Objects are instances of classes.

A link is a semantic connection among objects. In general, a link is an instance of an association. Whenever a class has an association to another class, there may be a link between the instances of the two classes; whenever there is a link between two objects, one object can send a message to the other object.

A message is the specification of a communication among objects that conveys information with expectation that activity will ensure. The receipt of message instance may be considered an instance of an event. When a message is passed, the action that results is an executable statement. An action may result in a change in state.

The focus of control is a tall, think rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

An object lifeline is a vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in an interaction diagram will be in existence of an object over a period of time. Most objects that appear in an interaction diagram will be in existence for the duration of the interaction.

Message to self: A message to self is a tool that sends a message from one object back to the same object. It does not involve other objects because the message returns to the same object. The sender of a message is the same as the receiver.

Sequence diagram for our system:

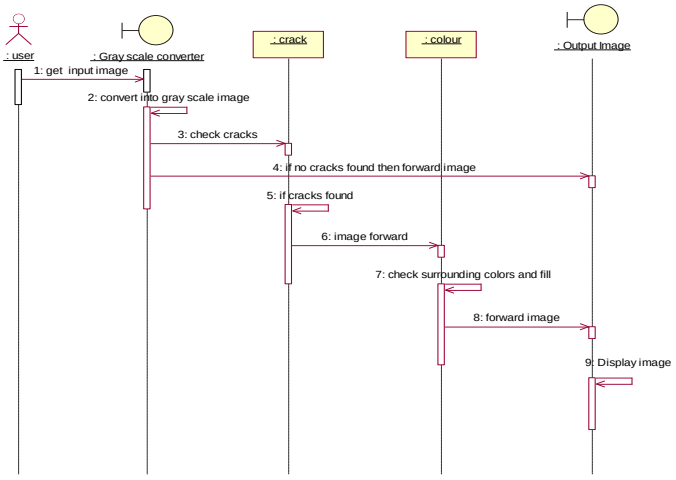


Figure 7.Sequence Diagram

5.2COLLABORATION DIAGRAM

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time-based sequence. Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

Two types of Numbering Sequences are:

1. Flat Sequence
2. Decimal Sequence

Differences between sequence and Collaboration diagrams are:

1. Sequence diagram is easy to read.
2. Collaboration diagram can be used to indicate how objects are statically connected.
3. There is no numbering in sequence diagram.
4. Sequence diagram shows the links between objects in a time based sequence.
5. Collaboration diagram shows how the objects associate with each other

Collaboration Diagram for our system:

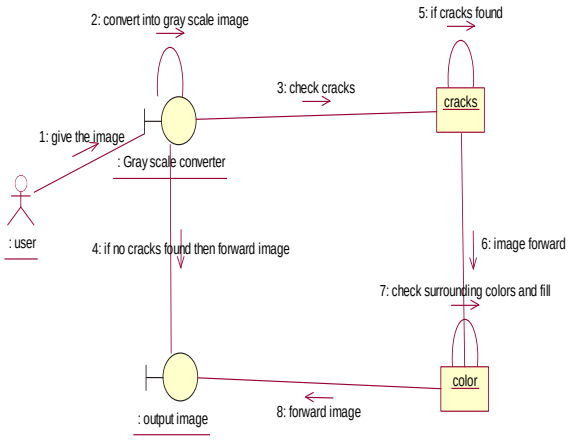


Figure 8 Collaboration Diagram

5.3 Class Diagram

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items.

There are 4 approaches for identifying classes:

1. Noun phrase approach:
2. Common class pattern approach.
3. Use case Driven Sequence or Collaboration approach.
4. Classes , Responsibilities and collaborators Approach
5.

Class diagram for our System:

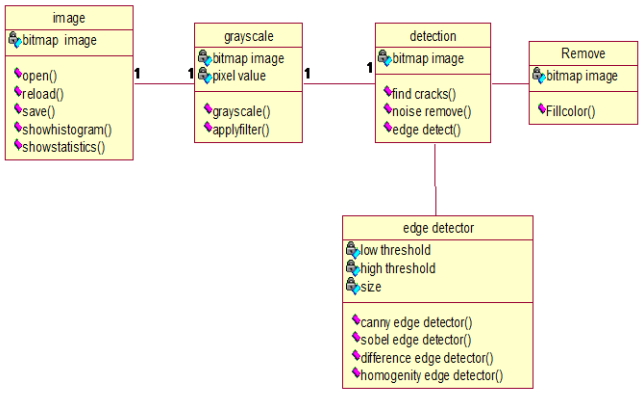


Figure 9. Class Diagram

6. SYSTEM IMPLEMENTATION

6.1.SYSTEM MAINTENANCE

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or software system. This is called the maintenance of the system. Nowadays there is the rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes. In our project the process can be added without affecting other parts of the system. Maintenance plays a vital role. The system is liable to accept any modification after its implementation. This system has been designed to favor all new changes. Doing this will not affect the system's performance or its accuracy.

6.2.SYSTEM TESTING

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently. In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing. Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. Inadequate testing or non-testing leads to errors that may appear few months later.

This will create two problems:

- Time delay between the cause and appearance of the problem
- The effect of the system errors on files and records within the system

The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits. The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and

on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing.
In the project system testing is made as follows: The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated.

This is the final step in system life cycle. Here we implement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports.

UNIT TESTING:

Unit testing verification efforts on the smallest unit of software design, module. This is known as "Module Testing". The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

INTEGRATION TESTING:

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

VALIDATION TESTING:

To uncover functional errors, that is, to check whether functional characteristics confirm to specification or not

Test Case for Grey Scale Conversion:

Test Case: Grey Scale Conversion
Test Description: If input image is binary
Pre Conditions: A cracked image is taken.
Action Performed: Binary image is converted
Expected Results: A Grey scale image is
Conditions Verified: yes
Result: Success

Test Case for Cracks Removal:

Test Case: Cracks Removal
Test Description: Image is converted to grey
Pre Conditions: Grey Scale image is
Action Performed: Cracks can be detected
Expected Results: Cracks are removed from
Conditions Verified: yes
Result: Success

7. CONCLUSION & FUTURE WORK

Cracks are detected by using top-hat transform, whereas the thin dark brush strokes, which are misidentified as cracks, are separated either by an automatic technique (MRBF networks) or by a semi-automatic approach. The methodology has been

applied for the virtual restoration of images and was found very effective by restoration experts. However, there are certain aspects of the proposed methodology that can be further improved. In this project we have presented a strategy for detection and filling of cracks, Crack interpolation is performed by appropriately modified order statistics filters or controlled anisotropic diffusion

The crack-detection stage is not very efficient in detecting cracks located on very dark image areas, since in these areas the intensity of crack pixels is very close to the intensity of the surrounding region. A possible solution to this shortcoming would be to apply the crack-detection algorithm locally on this area and select a low threshold value. Another situation where the system does not perform as efficiently as expected is in the case of cracks that cross the border between regions of different color. In such situations, it might be the case that part of the crack in one area is filled with color from the other area, resulting in small spurs of color in the border between the two regions. However, this phenomenon is rather seldom and, furthermore, the extent of these erroneously filled areas is very small (2–3 pixels maximum). A possible solution would be to perform edge detection or segmentation on the image and confine the filling of cracks that cross edges or region borders to pixels from the corresponding region. Another improvement of the crack filling stage could aim at using properly adapted versions of nonlinear multi-channel filters (e.g., variants of the vector median filter) instead of processing each color channel independently. These improvements will be the topic of future work on this subject

REFERENCES

Abas F and Martinez K. 2002. "Craquelure analysis for content-based retrieval," in *Proc. 14th Int. Conf. Digital Signal Processing*, vol. 1, ,pp. 111–114.

Barni M, Bartolini F and Cappellini V. 2000. "Image processing for virtual restoration of artworks," *IEEE Multimedia*, vol. 7, no. 2, pp. 34–37, Jun.

Bertalmio M, Vese L, Sapiro G and Osher S. 2003. "Simultaneous structure and texture image inpainting," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 882–889, Aug.

Bertalmio M, Sapiro G, Caselles V and Ballester C. 2000. "Image inpainting," in *Proc. SIGGRAPH*, pp. 417–424.

Chan T and Shen J. 2001. "Non-texture inpaintings by curvature-driven diffusions," *J. Vis. Commun. Image Represen.*, vol. 12, no. 4, pp. 436–449.

Efros A and Leung T. 1999. "Texture synthesis by nonparametric sampling," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 1033–1038.

Esedoglu S and Shen J. 2002. "Digital inpainting based on the Mumford-Shah-Euler image model," *Eur. J. Appl. Math.*, vol. 13, pp. 353–370.

Giakoumis I and Pitas I. 1998. "Digital restoration of painting cracks," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, pp. 269–272. Screen shots.

"Interpolation of missing data in image sequences," *IEEE Trans. Image Process.*, vol. 4, no. 11, pp. 1509–1519, Nov. 1995.

Joyeux L, Buisson O, Besserer B and Boukir S. 1999. "Detection and removal of line scratches in motion picture films," in *Proc. IEEE Int.* Conf.*Computer Vision and Pattern Recognition*, 1999, pp. 548–553.

Kokaram A, Morris R, Fitzgerald W and Rayner P. 1995. "Detection of missing data in image sequences," *IEEE Trans. Image Process.*, vol. 4, no. 11, pp. 1496–1508, Nov.

Masnou S and Morel JM. 1998. "Level lines based disocclusion," in *Proc. IEEE Int. Conf. Image Process.*, vol. III, pp. 259–263.

Object oriented analysis and design using UML- Simon Benett, Steeve MC Robb, Ray Farmer, PP 22-50,June 2003.