Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Hierarchy Super-Re-solution-Oriented Re-painting

Veeresh B. Hadli
(BE.., (MTech), Dept of CSE &VTU,
Belgum, India

Smt. Shobha
(BE.., ME), Dept of CSE & VTU,
Beglum, India

Abstract— This paper introduces a novel framework for examplaar-Oriented inpainting. It consists in performing first the inpainting on a coarse version of the input image. A Hierarchy super-Re-solution algorithm is then used to recover details on the missing areas. The advantage of this approach is that it is easier to inpaint low-Re-solution pictures than high-Re-solution ones. The gain is both in terms of computational complexity and visual quality. However, to be less sensitive to the parameter setting of the inpainting method, the low-Re-solution input picture is inpainted several times with different configurations. Results are efficiently combined with a loopy belief propagation and details are recovered by a single-image super-Re-solution algorithm. Experimental results in a context of image editing and texture synthesis demonstrate the effectiveness of the proposed method. Results are compared to five state-of-the-art inpainting methods.

Keywords— Examplaar-Oriented inpainting, single-image super-Re-solution.

I. INTRODUCTION

Image Re-painting refers to methods which consist in filling in missing regions (holes) in an image. Existing methods can be classified into two main categories. The first category concerns diffusion-Oriented approaches which propagate linear structures or level lines (so-called isophotes) via diffusion Oriented on partial differential equations and variational methods. The diffusion-Oriented methods tend to introduce some blur when the hole to be filled in is large. The second family of approaches concerns examplaar-Oriented methods which sample and copy best matching texture patches from the known image neighbourhood. These methods have been inspired from texture synthesis technique and are known to work well in cases of regular or repeatable textures. The first attempt to use examplaar-Oriented techniques for object removal has been reported. The authors improve the search for similar patches by introducing an a priori rough estimate of the Re-painted values using a multi-scale approach which then results in an iterative approximation of the missing regions from coarse-to-fine levels. The two types of methods (diffusion- and examplaar-Oriented) can be efficiently combined, e.g. by using structure tensors to compute the priority of the patches to be filled. A recent approach combines an examplaar-Oriented approach with super-Re-solution. It is a two-steps algorithm. First a coarse version of the input picture is Re-painted. The second step consists in creating an enhanced Re-solution picture from the coarse Re-painted image.

II. EISTING SYSTEM

Existing methods can be classified into two main categories. The first category concerns diffusion-Oriented approaches which propagate linear structures or level lines via diffusion Oriented on partial differential equations and variation methods. Unfortunately, the diffusion-Oriented methods tend to introduce some blur when the hole to be filled-in is large. The second family of approaches concerns exemplar-Oriented methods which sample and copy best matches texture patches from the known image neighborhood. These methods have been inspired from texture synthesis techniques and are known to work well in cases of regular or repeatable textures. The first attempt to use exemplar-Oriented techniques for object removal has been reported in. Authors in improve the search for similar patches by introducing an a priori rough estimate of the in painted values using a multi-scale approach which then results in an iterative approximation of the missing regions from coarse to fine levels.

Disadvantages of existing system

A. In existing methods not get clear image.
B. Old Re-painting method have take lot of time

Proposed System

In proposed system two main components are the in-painting and the super-Re-solution algorithms. More specifically, the following steps are performed:

1. A low-Re-solution image is first built from the original picture;
2. An in-painting algorithm is applied to fill-in the holes of the low-Re-solution picture;
3. The quality of the in-painted regions is improved by using a single-image SR method.

Advantages of proposed system

1. Capable of Re-painting large regions.
2. Most of texture of image can be rebuild
3. Most of structure if the image can be rebuilds.

Scope:

Re-painting is art of reconstructing the missing portions of images in order to make it more legible and restore its unity.

Brief outline of the project

The works carried out at each project phase are outlined below:-

Learning & Analysis Phase

This phase includes:

- Gathering knowledge about existing communicating techniques.
- Well understanding of the project design review from the client.
- Learning tools, technologies & programming Language for coding purpose.

Design & Implementation

This phase Includes:

- Designing the overall functional view i.e. system architecture of the project.
- Describing the language, platform used in the project implementation.
- Identification and design of the modules for implementing.
- Implementing the applications for accessing and controlling the different types of services.

Testing Phase

This phase includes:

- Writing the test cases for testing the implemented modules.
- Executing the test cases manually, comparing and evaluating the actual result with the expected result.

Organization of the project report

The technical aspects, system requirements and the organization of the project report are discussed as follows. The project report mainly consists of total 9 chapters, references and appendices. Chapter 1 gives the overall information related to the project. Chapter 2 gives information about the referred papers and websites. Chapter 3 gives a detailed study of background related to the project and also highlights some of the drawbacks related to the background. Chapter 4 gives details about several analysis that are performed to facilitate taking decision of whether the project is feasible enough or not. Chapter 5 includes some basic design concept, methodology, some necessary diagrams to outline the project easily. Chapter 6 describes implementation of modules and also dictates about the tools and technologies used for implementation. Chapter 7 defines various testing methods used for testing the different modules in the project. Chapter 8 is related to snapshots of the project along with interpretation. Chapter 9 concludes the dissertation work and summarizes the project. It also highlights some future enhancement scopes of the project.

III. LITERATURE SURVEY

Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can workout. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project. A variety of research has been done on Fault tolerance for applications. Following section explores different references that discuss about several topics related to Fault Tolerance.

Literature Survey

The SR problem is ill-posed since multiple high-Re-solution images can produce the same low-Re-solution image. Solving the problem hence requires introducing some prior information. The prior information can be an energy functional defined on a class of images which is then used as a regularization term together with interpolation techniques [11]. This prior information can also take the form of example images or of corresponding LR-HR (Low Re-solution – High Re-solution) pairs of patches learned from a set of un-related training images [12] or from the input low Re-solution image itself [13]. This latter family of approaches is known as examplaar-Oriented SR methods [12]. An examplaar-Oriented superRe-solution method embedding Knearest neighbours found in an external patch database has also been described in [14]. Instead of constructing the LR-HR pairs of patches from a set of un-related training images, the authors in [13] extract these correspondences by searching for matches across different scales of a multi-Re-solution pyramid constructed from the input low-Re-solution image.

The proposed method builds upon the super-Re-solutionOriented Re-painting method proposed in [10] which is Oriented on examplaar-Oriented Re-painting (in particular Criminisi-like approach [4]) and single-image examplaar-Oriented superRe-solution [13]. The main novelty of the proposed algorithm is the combination of multiple Re-painted versions of the input picture. The rationale behind this approach is to cope with the sensitivity of examplaar-Oriented algorithms to parameters such as the patch size and the filling order. Different combinations have been tested and compared. Besides this major point, different adjustments regarding examplaar-Oriented Re-painting and SR methods are described such as the use of the coherence measure to constrain the candidate search [15].

Summary

This chapter mainly discusses about the papers, websites that are referred while making this dissertation report. All these papers and websites provide information related to Fault tolerance, their existing solutions, protocol used and also their advantages & limitations.

IV. SYSTEM REQUIREMENT SPECIFICATION

Software requirement Specification is a fundamental document, which forms the foundation of the software development process. It not only lists the requirements of a system but also has a description of its major feature. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and nonfunctional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:-

1) Image restoration done accordingly in his project.
2) Re-solution also done.

Non-functional Requirement

Non functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

1. Product Requirements 2. Organizational Requirements 3. User Requirements 4. Basic Operational Requirements

Product Requirements

Portability: Since the software is developed in matlab it can be executed on any platform for which the matlab is available with minor or no modifications.

Correctness: It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

Ease of Use: The front end is designed in such a way that it provides an interface which allows the user to interact in an easy manner.

Modularity: The complete product is broken up into many modules and well-defined interfaces are developed to explore the benefit of flexibility of the product.

Robustness: This software is being developed in such a way that the over all performance is optimized and the user can expect the results within a limited time with utmost relevancy and correctness.

Non functional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

Organizational Requirements

Process Standards: IEEE standards are used to develop the application which is the standard used by the most of the standard software developers all over the world.

Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

The design of the system is perhaps the most critical factor affecting the quality of the software and has a major impact on the later phases, particularly testing and maintenance. We have to design the product with the standards which has been understood by the developers of the team.

User Requirements

- The user must be able to visualize the user interface using GUIDE.
- The user must be able to configure all the parameters with neat GUI.

Basic Operational Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points:-

Mission profile or scenario: It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.

Performance and related parameters: It points out the critical system parameters to accomplish the mission

Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

Operational life cycle: It defines the system lifetime.

Resource Requirement

Mat Lab 2010:

MATLAB (**mat**rix **lab**oratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Oriented Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia.[2] MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

The MATLAB application is built around the MATLAB language, and most use of MATLAB involves typing MATLAB code into the Command Window (as an interactive mathematical shell), or executing text files containing MATLAB code and functions.[6]

MATLAB can call functions and subroutines written in the C programming language or Fortran. A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "MEX-files" (for **M**ATLAB **ex**ecutable).[13][14]

Libraries written in Java, ActiveX or .NET can be directly called from MATLAB and many MATLAB libraries (for example XML or SQL support) are implemented as wrappers around Java or ActiveX libraries. Calling MATLAB from Java is more complicated, but can be done with a MATLAB extension,[15] which is sold separately by MathWorks, or using an undocumented mechanism called JMI (Java-to-MATLAB Interface),[16] which should not be confused with the unrelated Java Metadata Interface that is also called JMI.

As alternatives to the MuPAD Oriented Symbolic Math Toolbox available from MathWorks, MATLAB can be connected to Maple or Mathematica.[17]

Libraries also exist to import and export MathML

MATLAB has a number of competitors.[21] Commercial competitors include Mathematica, Maple, and IDL. There are also free open source alternatives to MATLAB, in particular GNU Octave, FreeMat, and Scilab which are intended to be mostly compatible with the MATLAB language. Among other languages that treat arrays as basic entities (array programming languages) are APL, Fortran 90 and higher, S-Lang, as well as the statistical language S. There are also libraries to add similar functionality to existing languages, such as IT++ for C++, Perl Data Language for Perl, and SciPy for Python.

You can use MATLAB for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing.

Hardware Requirements

Processors	:	Pentium IV.
RAM	:	64 MB.
Storage	:	20GB.
Monitor	:	15”

Keyboard : Standard 102 keys
Mouse : 3 buttons

Software (Tools &Technologies) Requirements

Platform : Windows XP.
Language : MATLAB
IDE/tool : MATLAB 2010Ra

Summary

This chapter gives details of the functional requirements, non-functional requirements, resource requirements, hardware requirements, software requirements etc. Again the non-functional requirements in turn contain product requirements, organizational requirements, user requirements, basic operational requirements etc.

V. SYSTEM ANALYSIS

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately this condition does not prevail in practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case there are three primary areas of interest:-

Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Normally, the OS is windows XP. The main theme of this project is to allocate path channels Oriented on the hot spot and clod spot. Performance analysis is done to find out whether our algorithm is more efficient. It is essential that the process of performance analysis and definition must be conducted in parallel. We measure the parameter called Packet delivery to measure the effectiveness of the approach.

Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organization's technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis there are several points to focus on:-

Changes to bring in the system: All changes should be in positive direction, there will be increased level of efficiency and better customer service.

Required skills: Platforms & tools used in this project are widely used. So the skilled manpower is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the user's point of view.

Economical Analysis

Economical analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not have high performance servers. All the functions of implemented through software modules. In this system we are not using any physical devices for connection. So the system is economically feasible enough.

Summary

The main aim of this chapter is to find out whether the system is feasible enough or not. For these reasons different kinds of analysis, such as performance analysis, technical analysis, economical analysis etc is performed.

VI. SYSTEM DESIGN

Design is a creative process; a good design is the key to effective system. The system "Design" is defined as "The process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization". Various design features are followed to develop the system. The design specification describes the features of the system, the components or elements of the system and their appearance to end-users.

Fundamental Design Concepts

A set of fundamental design concepts has evolved over the past three decades. Although the degree of interest in each concept has varied over the years, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. The fundamental design concepts provide the necessary framework for "getting it right". The fundamental design concepts such as abstraction, refinement, modularity, software architecture, control hierarchy, structural partitioning, data structure, software procedure and information hiding are applied in this project to getting it right as per the specification.

Input Design

The input Design is the process of converting the user-oriented inputs in to the computer-Oriented format. The goal of designing input data is to make the automation as easy and free from errors as possible. Providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of the project. Input design is a part of overall system design which requires very careful attention. Often the collection of input data is the most expensive part of the system, which needs to be route through number of modules .

Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other systems through outputs. It is most important and direct source information to the user. Efficient and intelligent output improves the systems relationship with source and destination machine.

System development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

Model phases

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

Requirement Analysis: This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.

System Design: Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm**,** data structure**,** software architecture etc.

Coding: In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

Testing: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

Reason for choosing waterfall model as development method

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification
- Better resource allocation.
- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

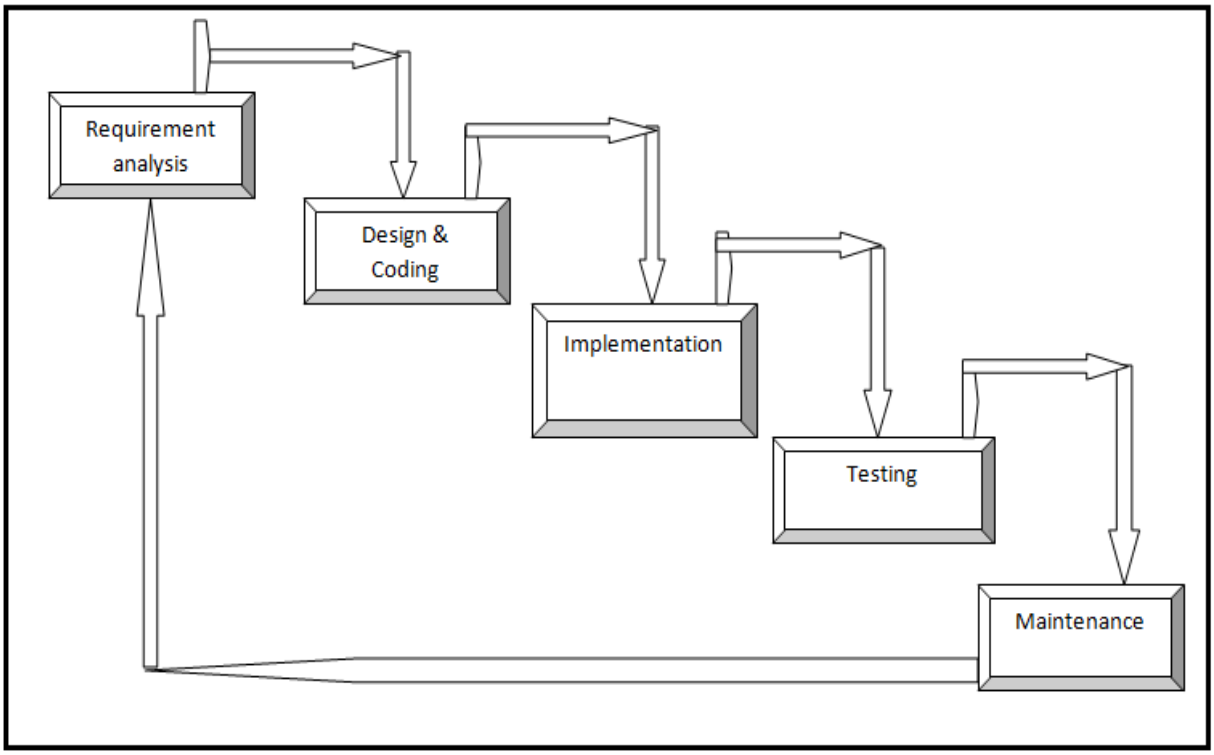
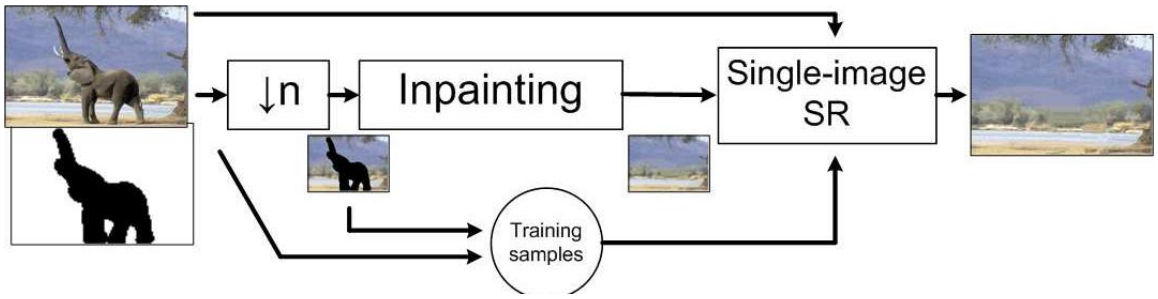


Fig 6.4:- Waterfall model

System Architecture

System architecture is the conceptual design that defines the structure and behavior of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system.

The System architecture is shown below.



MODULE DESCRIPTION:

Image in painting

In painting is the process of reconstructing lost or deteriorated parts of images and videos. For instance, in the museum world, in the case of a valuable painting, this task would be carried out by a skilled art conservator or art restorer. In the digital world, in painting refers to the application of sophisticated algorithms to replace lost or corrupted parts of the image data.

Image restoration

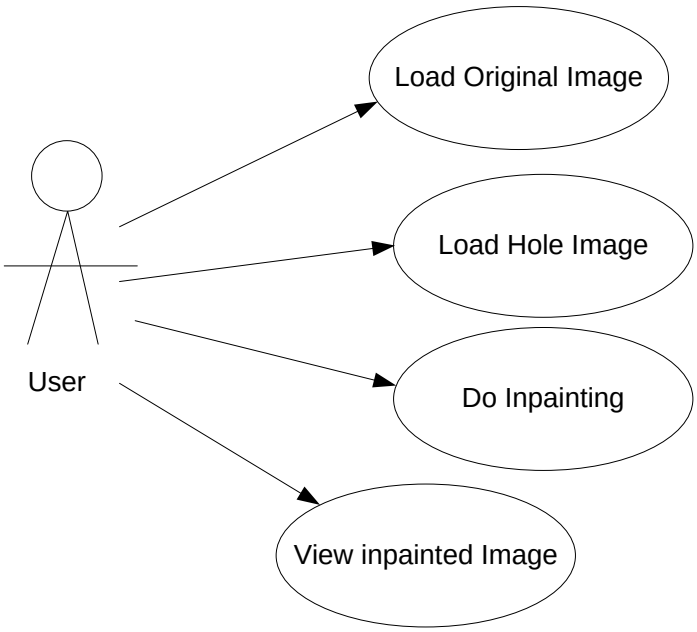
Image restoration is the operation of taking a corrupted/noisy image and estimating the clean original image. Corruption may come in many forms such as motion blur, noise, and camera miss focus.

Super-Re-solution

Super Re-solution (SR) is a class of techniques that enhance the Re-solution of an imaging system. In some SR techniques—termed *optical* SR—the diffraction limit of systems is transcended, while in others—*geometrical* SR—the Re-solution of digital imaging sensors is enhanced.

Use case Diagram of the system

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

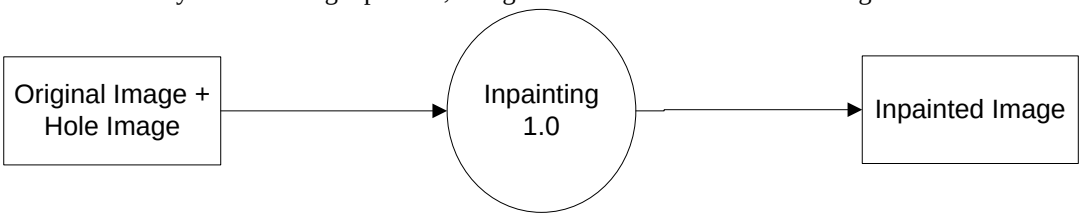


Data Flow Diagram of the system

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

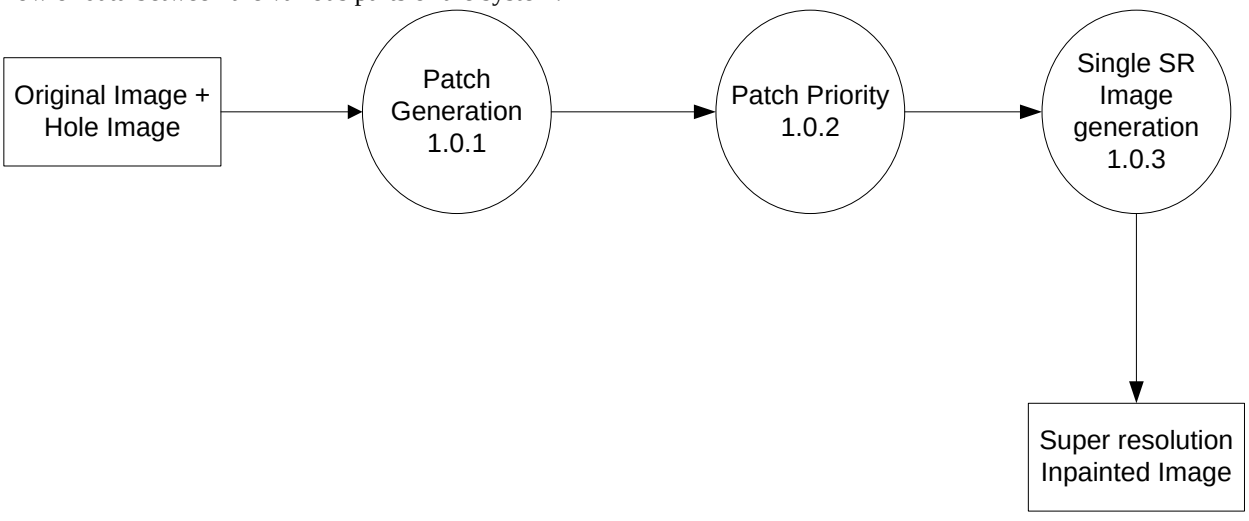
Level 0 Data flow diagram

A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization

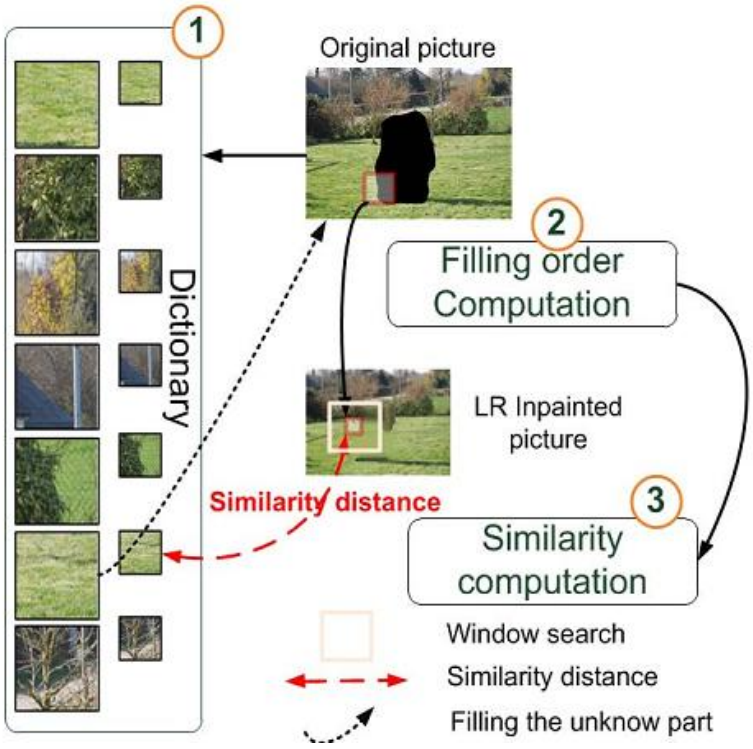


Level 1 Data flow diagram

The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



Flow chart of Super Re-solution Algorithm



Summary

This chapter mainly concentrates on few fundamental design concepts such as input & output design, algorithm used in the project, system development methodology, system architecture, use-case diagram, data flow diagram etc.

VII. IMPLEMENTATION

Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage the main workload and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned and controlled, it can cause chaos and confusion.

The implementation stage requires the following tasks.

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform
- Appropriate selection of the language for application development

.

Language used for implementation

Implementation phase should perfectly map the design document in a suitable programming language in order to achieve the necessary final and correct product. Often the product contains flaws and gets ruined due to incorrect programming language chosen for implementation.

In this project, for implementation purpose mat lab is chosen as the programming language. Few reasons for which mat lab is selected as a programming language can be outlined as follows:-MATLAB has several advantages over other methods or languages:

- Its basic data element is the matrix. A simple integer is considered an matrix of one row and one column. Several mathematical operations that work on arrays or matrices are built-in to the Matlab environment. For example, cross-products, dot-products, determinants, inverse matrices.
- Vectorized operations. Adding two arrays together needs only one command, instead of a for or while loop.
- The graphical output is optimized for interaction. You can plot your data very easily, and then change colors, sizes, scales, etc, by using the graphical interactive tools.
- Matlab's functionality can be greatly expanded by the addition of toolboxes. These are sets of specific functions that provided more specialized functionality. Ex: Excel link allows data to be written in a format recognized by Excel, Statistics Toolbox allows more specialized statistical manipulation of data (Anova, Basic Fits, etc)

Matlab is an interpreted language for numerical computation. It allows one to perform numerical calculations, and visualize the results without the need for complicated and time consuming programming. Matlab allows its users to accurately solve problems, produce graphics easily and produce code efficiently.

Platform used for Implementation

A platform is a crucial element in software development. A platform might be simply defined as "a place to launch software". In this project, for implementation purpose Windows XP platform is used & reasons for choosing this platform are Integrated Networking support, More stable and secure than previous version, Contain remote desktop connection and restore option, Enhanced device driver verifier, Dramatically reduced reboot scenarios, Improved code protection, Side-by-side DLL support, Windows File Protection, Preemptive multitasking architecture, Scalable memory and processor support, Encrypting File System (EFS) with multi-user support, IP Security (IPSec), Kerberos support, Smart card support, Internet Explorer Add-on Manager, Windows Firewall, Windows Security Center, Fresh visual design.

Algorithms

Super-Re-solution algorithm

Once the in painting of the low-Re-solution picture is completed, a single-image super-Re-solution approach is used to reconstruct the high Re-solution of the image. The idea is to use the low-Re-solution in painted areas in order to guide the texture synthesis at the higher Re-solution. The problem is to find a patch of higher-Re-solution from a database of examples.

1. Dictionary building: it consists of the correspondences between low and high Re-solution image patches. The unique constraint is that the high-Re-solution patches have to be valid, i.e. entirely composed of known pixels. In the pro- posed approach, high-Re-solution and valid patches are evenly extracted from the known part of the image. The size of the dictionary is a user-parameter which might influence the overall speed/quality trade-off. An array is used to store the spatial coordinates of HR patches (DHR). Those of LR patches are simply deduced by using the decimation factor;
2. Filling order of the HR picture: the computation of the filling order is similar to the one described in Section 3. It is computed on the HR picture with the sparsity-Oriented method. The filling process starts with the patch HRp 10 Olivier Le Meur and Christine Guillemot having the highest priority. This improves the quality of the in painted picture compared to a raster-scan filling order;
3. For the LR patch corresponding to the HR patch having the highest priority, its K-NN in the in painted images of lower Re-solution are sought. The number of neighbors is computed as described in the previous section. The similarity metric is also the same as previous;
4. Weights wp, pj are calculated by using a non-local means method as if we would like to perform a linear combination of these neighbors. However, the similarity distance used to compute the weights is composed of two terms: the first one is classical since this is the distance between the current LR patch and its LR neighbors, noted d(LRp , LRp,pj). The second term is the distance between the known parts of the HR patch HRp and the HR patches corresponding to the LR neighbours of LRp . Say differently, the similarity distance is the distance between two vectors composed of both pixels of LR and HR patches. The use of pixel values of HR patches allows to constraint the nearest neighbour search of LR patches.
5. A HR candidate is finally deduced by using a linear combination of HRpatches with the weights previously computed: HRp =Xpj2DHRwp,pj × p,pj (4) with the usual conditions $0 \leq wp,pj \leq 1$, andPk wp,pk = 1.
6. Stitching: the HR patch is then pasted into the missing areas. However, as an overlap with the already synthesized areas is possible, a seam cutting the overlapped regions is determined to further enhance the patch blending. The minimum error boundary cut [21] is used to find a seam for which the two patches match best. The similarity measure is the Euclidean distance between all pixel values in the overlapping region. More complex metrics have been tested but they do not substantially improve the final quality. At most four overlapping cases (Left, Right, Top and Bottom) can be encountered. There are sequentially treated in the aforementioned order. The stitching algorithm is only used when all pixel values in the overlapping region are known or already synthesized. Otherwise, the stitching is disabled. After the filling of the current patch, priority value is recomputed and the afore-mentioned steps are iterated while there exist unknown areas.

Summary

This chapter gives implementation details of the two major subsystems which are developed for this project. With the help of data flow diagram, it also specifies the logic of implementation for the different modules that have been specified during the system design. Along with these, this chapter also highlights some of the important features of the platform and language used for implementation purpose.

VIII. TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-Oriented system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions .The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. Testing is the final verification and validation activity within the organization itself. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as a solution to the original problem.
- To provide operational reliability of the system.

During testing the major activities are concentrated on the examination and modification of the source code.

Unit Testing

Here each module that comprises the overall system is tested individually. Unit testing focuses verification efforts even in the smallest unit of software design in each module. This is also known as "Module Testing". The modules of the system are tested separately. This testing is carried out in the programming style itself. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. In this step each module is found to work satisfactorily as regard to the expected output from the module. This testing is done to check for the individual block codes for their working. It is done so that when we carry out functional testing then the units which are part of these functionalities should have been tested for working.

The following unit testing table shows the functions that were tested at the time of programming. The first column lists all the functions which were tested and the second column gives the description of the tests done.

Table 8.1:- Unit testing table

Function	Tests done	Remarks
Browse GUI	Tested to check whether the browse function opens file chooser window to choose call input file.	Success
Working of Re-painting	Tested to check whether the original and holed images were loaded and do impainting. Finally view the impainted image.	Success

Integration

After successful completion of unit testing or module testing, individual functions are integrated into classes. Again integration of different classes takes into place and finally integration of front-end with back-end occurs.

- **Integration of functions into classes**
  At the start of coding phase only the functions required in different parts of the program are developed. Each of the functions is coded and tested independently. After verification of correctness of the different functions, they are integrated into their respective classes.

- **Integration of different classes**
  Here the different classes are tested independently for their functionality. After verification of correctness of outputs after testing each class, they are integrated together and tested again.

- **Integration of front-end with back-end**
  The front-end of the project is developed in mat lab environment. The user interface is designed to facilitate the user to input various commands to the system and view the system's normal and faulty behavior and its outputs. The back-end code is then integrated with the GUI and tested.

Integration Testing

Data can be lost across interface. One module can have an adverse effect on another. Sub functions when combined, should not reduce the desired major function. Integration testing is a systematic technique for constructing the program structure. It addresses the issues associated with the dual problems of verification and program construction. The main objective in this testing process is to take unit tested modules and build a program structure that has been dictated by design.

After the software has been integrated, a set of high order tests are conducted. All the modules are combined and tested as a whole. Here correction is difficult, because the isolation of errors is complicated by the vast expanse of the entire program.

Top down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward, beginning with the main program module. Modules that subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from bottom to up, processing required for modules subordinate to a given level is always available. Therefore in this case the need for stubs is eliminated. The following integration testing table shows the functions that were combined into different classes and the class as a whole tested for its functionality. This is important to check for error-free interaction between various classes, and maintenance of data integrity.

Table 8.2:- Integration testing table

Classes integrated	Functions integrated in each class	Tests done	Remarks
Class: Main	browseOriginalImage() browseHoleImage() doRe-painting()	Class tested to check whether all commands that were applied are working correctly and appropriately or not.	Success
Class: Inpainitng	generatePatch() patchPriority() singleSRImageGeneration() superRe-solution()	Class tested to check whether all operations are calling appropriate function whenever necessary.	Success

Validation Testing

At the culmination of integration testing, software is completed and assembled as a package. Interfacing errors are uncovered and corrected. Validation testing can be defined in many ways. Here the testing validates the software function in a manner that is reasonably expected by the customer.

Table 8.3:- Validation testing table

Functionality to be tested	Input	Tests done	Remarks
Working of Front-End	User interaction with help of a mouse and keyboard	Appropriate forms open when buttons are clicked	Success
Working of Re-painting	User has to upload the original image and holed image. And start inpaining.	Re-painting of image done.	Success

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Therefore the output testing involves first of all asking the users about the format required by them and then to test the output generated or displayed by the system under consideration. The output format is considered in 2 ways: –

(1) On screen (2) Printed format

User Acceptance Testing

User Acceptance of a system is the key factor to the success of any system. Performance of an acceptance test is actually the user's show. User motivation and knowledge are critical for the successful performance of the system.

The system under consideration is tested for user acceptance by constantly in touch with the prospective system users at time of developing and making changes wherever required in regard to the following point:

- Input Screen design
- Output Screen design
- Menu driven system

White box testing

White box testing (clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases Oriented on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. While white box testing is applicable at the unit, integration and system levels of the software testing process, it is typically applied to the unit. While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test.

Though this method of test design can uncover an overwhelming number of test cases, it might not detect unimplemented parts of the specification or missing requirements, but one can be sure that all paths through the test object are executed. Using white box testing we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity

Black box testing

Black box testing focuses on the functional requirements of the software. It is also known as functional testing. It is a software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs.

The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications. It enables us to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing is an alternative to white box technique. Rather it is a complementary approach that is likely to uncover a different class of errors in the following categories:-

- Incorrect or missing function.
- Interface errors.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

Advantages

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

Preparation of Test Data

Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by using test data, errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to suggest data for test from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files that they have entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing and although the realistic data that will show how the system will perform for the typical processing requirement. Assuming that the live data entered are in fact typical; such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true system test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an "umbrella activity" that is applied throughout the engineering process. Software quality assurance encompasses:-

- Analysis, design, coding and testing methods and tools
- Formal technical reviews that are applied during each software engineering
- Multitiered testing strategy
- Control of software documentation and the change made to it.
- A procedure to ensure compliance with software development standards.
- Measurement and reporting mechanisms.

Quality Factors

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- ✓ Factors that can be directly measured.
- ✓ Factors that can be indirectly measured

These factors focus on three important aspects of a software product

- Its operational characteristics
- Its ability to undergo changes
- Its adaptability to a new environment.
- Effectiveness or efficiency in performing its mission
- Duration of its use by its customer.

Generic Risks

A risk is an unwanted event that has negative consequences. We can distinguish risks from other project events by looking for three things:

- A loss associated with the event.
- The likelihood that the event will occur.
- The degree to which we can change the outcome

The generic risks such as the product size risk, business impact risks, customer–related risks, process risks, technology risks, development environment risks, security risks etc. This project is developed by considering all these important issues.

Security Technologies & Policies

The software quality assurance is comprised of a variety of tasks associated with seven major activities:-

- Application of technical methods.
- Conduct of formal technical reviews
- Software testing
- Enforcement of standards
- Control of change
- Measurement
- Record keeping and reporting

Summary

This chapter deals with several kinds of testing such as unit testing which is a method of testing the accurate functioning of a particular module of the source code. It is also referred to as module testing. It also gives a brief detail about different kinds of integration testing in which individual software modules are combined and tested as a group. Other then these main two kinds of testing, many other types such as validation testing, output testing, user acceptance testing and preparation of test data also discussed here. This chapter also focuses on assuring quality of the software.

IX. INTERPRETATION OF RESULT

The following snapshots define the results or outputs that we will get after step by step execution of all the modules of the system.

Interpretation:-Once application is started.

Summary

This chapter gives a brief interpretation of the expected and obtained result when each and every module is executed in their proper sequence

X. CONCLUSION & FUTURE SCOPE

Conclusion

A novel Re-painting approach has been presented in this paper. The input picture is first down sampled and several Re-paintings are performed. The low-Re-solution Re-painted pictures are combined by globally minimizing an energy term. Once the combination is completed, a Hierarchy single image super Re-solution method is applied to recover details at the native Re-solution. Experimental results on a wide variety of images have demonstrated the effectiveness of the proposed method. One interesting avenue of future work would be to extend this approach to the temporal dimension. Also, we plan to test other SR methods to bring more robustness to the method. But the main important improvement is likely the use of geometric constraint and higher-level information such as scene semantics in order to improve the visual relevance.

REFERENCES

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Re-painting," in Proc. 27th Annu. Conf. Comput. Graph. Interact. Tech., Jul. 2000, pp. 417–424.

[2] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with PDEs: A common framework for different applications,"IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 4, pp. 506–517, Apr. 2005.

[3] T. Chan and J. Shen, "Variational restoration of non-flat image features: Models and algorithms," SIAM J. Appl. Math., vol. 61, no. 4, pp. 1338–1361, 2001.

[4] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by examplaar Oriented image Re-painting," IEEE Trans. Image Process., vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

[5] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-Oriented image completion,"ACM Trans. Graph., vol. 22, no. 2003, pp. 303–312, 2003.

[6] P. Harrison, "A non-Hierarchy procedure for re-synthesis of complex texture," in Proc. Int. Conf. Central Eur. Comput. Graph., Vis. Comput. Vis., 2001, pp. 1–8.

[7] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing,"ACM Trans. Graph., vol. 28, no. 3, p. 24, Aug. 2009.

[8] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," inProc. 7th IEEE Comput. Vis. Pattern Recognit., Sep. 1999, pp. 1033–1038.

[9] O. Le Meur, J. Gautier, and C. Guillemot, "Examplaar-Oriented Re-painting Oriented on local geometry," inProc. 18th IEEE Int. Conf. Image Process., Sep. 2011, pp. 3401–3404.

[10] O. Le Meur and C. Guillemot, "Super-Re-solution-Oriented Re-painting," in Proc. 12th Eur. Conf. Comput. Vis., 2012, pp. 554–567.

[11] S. Dai, M. Han, W. Xu, Y. Wu, Y. Gong, and A. Katsaggelos, "SoftCuts: A soft edge smoothness prior for color image super-Re-solution,"IEEE Trans. Image Process., vol. 18, no. 5, pp. 969–981, May 2009.

[12] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-Oriented superRe-solution," IEEE Comput. Graph. Appl., vol. 22, no. 2, pp. 56–65, Mar.–Apr. 2002.

[13] D. Glasner, S. Bagon, and M. Irani, "Super-Re-solution from a single image," in Proc. IEEE Int. Conf. Comput. Vis., vol. 10. Oct. 2009, pp. 349–356.

[14] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-Re-solution through neighbor embedding," inProc. IEEE Comput. Vis. Pattern Recognit.,vol.1. Jun.–Jul. 2004, pp. 275–282.

[15] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," in Proc. IEEE Comput. Vis. Pattern Recognit., Jun.–Jul. 2004, pp. I-120–I-127.

[16] Z. Xu and J. Sun, "Image Re-painting by patch propagation using patch sparsity," IEEE Trans. Image Process., vol. 19, no. 5, pp. 1153–1165, May 2010.

[17] S. Di Zenzo, "A note on the gradient of a multi-image," Comput. Vis., Graph., Image Process., vol. 33, no. 1, pp. 116–125, 1986.

[18] J. Weickert, "Coherence-enhancing diffusion filtering,"Int. J. Comput. Vis., vol. 32, nos. 2–3, pp. 111–127, 1999.

[19] J. Kopf, W. Kienzle, S. Drucker, and S. B. Kang, "Quality prediction for image completion," ACM Trans. Graph., vol. 31, no. 6, p. 131, 2012.

[20] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro, "A comprehensive framework for image Re-painting," IEEE Trans. Image Process., vol. 19, no. 10, pp. 2634–2644, Oct. 2010.

[21] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in Advances in Neural Information Processing System. Cambridge, MA, USA: MIT Press, 2000.

[22] A. Buades, B. Coll, and J. Morel, "A non local algorithm for image denoising," in Proc. IEEE Comput. Vis. Pattern Recognit.,vol.2. Jun. 2005, pp. 60–65.

[23] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in Proc. SIGGGRAPH, 2003, pp. 313–318.

[24] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning,"IEEE Trans. Image Process., vol. 16, no. 11, pp. 2649–2661,

