# Objective

The objective of this proposal is to provide details of a method to determine the nature and extent of damage (cracks, spalling, rebar exposure etc.) to concrete walls based on the processing of high resolution images. The client has provided a set of five training images which have been pre-analyzed showing the extents of damage in a section of the image. The client would like to develop a process to automate the process and apply it to a new set of test images to extract information about the damage therein.

# Proposed Method

## Literature Survey

## Overview

We propose a method that primarily uses a machine learning classifier such as Support Vector Machines to identify pixels that correspond to various damage states. To train the classifier, we intend to construct a feature vector that consists of the color and gradient intensities of the pixel under consideration as well as the histogram of similar quantities from a *kernel* or a subgrid of pixels (e.g. 3x3 or 5x5 pixels) that surround the pixel under consideration. By using the kernel histograms in addition to the pixel color/gradient intensities, we expect the classifier performance to be better since this allows the classifier to normalize the features w.r.t its surroundings and therefore identify damage better across a range of images which may have been taken under different conditions (e.g. different types of walls, lighting conditions etc.). Once we have made a first pass at classifying the pixels, we would then use a method such as the Hough Transformation to connect adjoining pixels that have been identified with similar damage (e.g. cracks) to extract further meta information.

The steps involved for the proposed approach are hereby detailed:

1. **Preparing the training dataset**. This first involves overlaying the analysis images on top of the original images with the approporiate offsets. This allows us to build a 1-to-1 map of the pixels indicating damage (and also pixels that do not indicate damage) in the provided training images.
2. **Constructing a feature vector from a pixel**. This would involve the transformation of the pixels to other color spaces (e.g. YCrCb, grayscale etc) and using the intensities along these dimensions as feature quantities. We would also attempt to use Canny edge detection algorithms to use the pixel gradient magnitudes as pixel features. This step might involve some experimentation to identify the best color-spaces and transformations that provide the most accurate classifier performance.
3. **Appending a feature vector from a kernel.** In this step, we obtain the local features corresponding to the region surrounding the pixel under consideration. In some cases, average quantities will be applicable and in other cases, a full histogram of quantities from all pixels in the kernel will be used. These kernel features will be appended to the pixel features previously developed.
4. **Principal Component Analysis (PCA).** Once we have a complete feature vector from pixels as well as the kernels, we will also consider performing a PCA to extract the most important features and discard those that do not affect the classification. This step may prove to be helpful in reducing the noise in the classification and improve accuracy.
5. **Training a classifier**. In this step, we will train a classifier to provide the most accurate classification possible. We will start by using a Support Vector Machine Classifier and this step will involve calibrating the hyperparameters C and alpha that define the smoothness of the classification boundaries.
6. **Application on sample images**. Once we have a trained classifier, we will apply it on the test (sample) images to identify the pixels containing the damage classes that we have trained the classifer for. In this case, a subgrid of pixels (of the same size as the previously used kernel) will be sent and the classifier will return a class identification (e.g. no-damage, crack, spall etc.)
7. **Post-processing to extract meta-information**. Once we have identified pixels that correspond to various damage classes, we perform the postprocessing step to connect adjacent pixels of similar damage (with sufficient threshold on gaps) to obtain an estimate of the size and extent of the damage.

## Process Deployment

The entire workflow detailed in the previous section will be implemented in Python. The main modules and libraries to be used are listed below:

* **Python 3** either onMS Windows or Linux.
* **OpenCV**: Computer vision and image processing library.
* **Scikit-learn**: Library providing support-vector-machines and other classifiers.
* **MatplotLib**: plotting library.
* **Scipy:** Scientific library for optimization, PCA etc.
* **Numpy**: numerical library for matrix operations and such.

Depending on the clients needs, the entire workflow can be deployed in terms of a Jupyter notebook which provides a convenient web interface to modify the workflow and follow the intermediate results or we can pursue a compiled application which can be deployed even on machines that do not have the aforementioned libraries installed. Later down the line, we could also consider deploying this process on a mobile application that can provide a real-time assessment of damage states based on camera images being captured on the phone.

# Scope of Work

## Tasks and Timeline

## Budget

# Proposer Information & Prior Experience

## Principal Investogators

## Proposing Firm