This document only specifies the functions that need to be modified to add new techniques/algorithms to any of the modules. Please refer Appendix-A of the report for a detailed description of the various modules and their associated classes and functions.

Adding new parameters: The application reads the initial parameter values from a text file called init_parameters.txt through a class called `SystemParameters`, defined in `systemParameters.hpp` and `systemParameters.cpp`. So, when a new parameter is added to any module, corresponding entries must be made in this text file and also in the corresponding member function of the `SystemParameters` class called `readInitial**Params`, where `**` refers to the name of the concerned module. To ensure that each parameter value is read into the correct variable, the parameter numberings in the text file and the reading function must match.

Adding new techniques/algorithms: Following is a list of the main system modules along with the procedure to add new methods to each:

1. User interface module: This module is responsible for reading and writing input and output videos. Input can be taken either from a video file or a camera through a `CvCapture` object called `video_reader`. Output is in the form of images displayed during program execution and also videos that are written using `CvVideoWriter` objects called `bounding_box_video_writer`, `combined_video_writer` and `static_mask_video_writer`. Following are some of the changes that can be made:

i. The input video file location and name can be changed in `initInputInterface`.
ii. To increase the number of output videos/images, the corresponding `CvVideoWriter` or IplImage objects must be added to the `UserInterface` class and following functions must be edited:
    - `UserInterface` (`constructor`): all objects must be initialized to NULL.
    - `initOutputInterface`: add `cvCreateVideoWriter` or `cvCreateImage` statements to allocate memory for video/image objects.
    - `clearOutputInterface`: add `cvReleaseVideoWriter` and `cvReleaseImage` statements to free up the allocated memory.
iii. To make new parameters adjustable in real time, the corresponding track bar should be added to `initIOWindow` or `initProcWindow` depending on the nature of the parameter.

2. Pre-processing module: This module is implemented by `PreProcessing` class defined in `preprocessing.hpp` and `preprocessing.cpp`.

i. Additional contrast enhancement techniques can be added to `performContrastEnhancement` and a corresponding entry should be added to `updateContrastEnhancementMethod` along with an update to the `contrastM` track bar in `initWindow`.
ii. Additional noise reduction techniques can be added to `performNoiseReduction`and a corresponding entry should be added to `updateNoiseReductionMethod`along with an update to the `noiseM` track bar in `initWindow`.

3. BGS module: This module currently utilizes four different classes, one for each BGS technique: `GrimsonGMM`, `ZivkovicAGMM`, `AdaptiveMedianBGS` and `WrenGA`. To modify any of these, changes should be made to the corresponding cpp/hpp files. One point to be noted is that all of these classes implement (or extend) the virtual class `Bgs` (defined in `Bgs.hpp`) and therefore if any functions are to added to any of them that need to be called directly from the main function, corresponding virtual function must be defined in `Bgs.hpp` and must also be instantiated in each of these 4 classes (possibly with empty functions in case it is not needed in one or more of these).Some of the parameters common to all these classes are managed by a class called `BgsParams` (`BgsParams.hpp`).

A single wrapper class called `BgsModels` (`bgsModels.hpp/cpp`) is used to manage all of these classes and must be modified if any new BGS techniques are to be added.

i. Add initialization statements to `initBGSMethod`.
ii. Additional functions must be written to initialize, update and print the parameters for the new method. These should be named (if convention is to be followed) as `init**Params`, `update**ParamsLocal`, `update**ParamsGlobal` and `print**Params` where `**` should be replaced by the name of the new technique.
iii. Additional track bars should be added to `initWindow` and an existing track bar called `bgsMethod` should also be updated.
iv. Additional parameters pertaining to the BGS module in general can be added to either `bgs_struct` or `bgs_toggle_struct` structure depending on whether they are to be used for adjusting the performance of specific methods or for turning them on/off.
v. New parameters for individual BGS techniques should be added to the corresponding structure (`gmm_struct`, `running_gaussian_struct` or `adaptive_median_struct`)
vi. To make new parameters adjustable in real time, the corresponding track bars should be added to `initWindow or initMorphWindow` respectively for the above two cases.

4. Foreground analysis module: This module uses the `ForegroundProc` class defined in `foregroundProcessing.hpp` and `foregroundProcessing.cpp`.

i. New method for detecting quick lighting change (or any other king of false foreground) can be added by altering `removeFalseForeground` function to replace (or augment) the default function `detectLightingChange`.
ii. New method for detecting cast shadows can also be added by altering the `removeFalseForeground` function and setting the `isShadow` function pointer to point to the new function.
iii. Additional parameters can be added to the `frg_struct` or `morph_struct` structure depending on whether they are to be used for adjusting the performance of specific methods or for turning them on/off.
iv. To make new parameters adjustable in real time, the corresponding track bars should be added to `initWindow or initMorphWindow` respectively for the above two cases.

5. Blob extraction module: This module consists of a frontend class `BlobDetection` (defined in `blobDetection.hpp` and `blobDetection.cpp`) that uses functions from a modified version of

the online blob extraction/manipulation library cvBlobsLib located in the `blob_detection` subfolder. Changes to the existing blob extraction method can therefore be made by altering the `BlobResult.h/cpp` and `blob.h/cpp` files located in this directory. It is important to note that the solution `cvblobslib.sln` must be opened in Visual Studio and recompiled before any such changes actually take effect since the application directly utilizes the libraries created when this solution is compiled (rather than using the actual source).

i. To add a new blob extraction method, changes must be made in the `getBlobs` function of `BlobDetection` class.
ii. Additional parameters should be added to the `blob_detection_struct` structure and `initParams` should be modified accordingly.
iii. To make new parameters adjustable in real time, the corresponding track bar should be added to `initWindow` and some related changes might also be required to `updateParams`.

6. Blob tracking module: This uses the `BlobTracking` class defined in `blobTracking.hpp/cpp` files.

i. To add a new method of occlusion detection either `isOccluded` should be modified or its call in `updateTrackedBlobs` should be replaced by the new function.
ii. To add new methods for calculating the area of a blob or the distance between two blobs, `updateBlobAreas` and `updateBlobDistances` should respectively be modified.
iii. To change the criteria of matching two blobs, matchBlobs must be changed.
iv. To change the criterion for comparing the appearance of two blobs `updateMovingAverageOfDifference` should be changed or its call in `updateTrackedBlobs` should be replaced.
v. To alter the criteria of adding/removing blobs or changing the status of existing blobs, changes should be made to `processTrackedBlobs`.
vi. To alter the manner in which tracked blobs are displayed to the user, `updateBlobImage` can be changed.
vii. To change what blobs are used as feedback to the BGS module, make changes to `updateTrackedStaticMask`.
viii. Additional parameters should be added to either `match_struct` or `track_struct` structure depending on whether these influence the criteria for blob correspondence establishment or the addition/removal/modification of blobs to the tracking system.
ix. To make new parameters adjustable in real time, the corresponding track bar should be added to either `initWindowMatch` or `initWindowTrack`depending respectively for the above two cases.

7. Abandonment analysis module: This module uses the class `AbandonmentAnalysis` defined over `abandonmentAnalysis.hpp/cpp` and `regionGrowing.cpp`.

i. To add a new approach to using region growing, alter the functions `detectRemovedObjectsUsingRegionGrowing` and `evaluateRegionGrowingPixelCount`. To add a new method of comparing a new pixel

with existing region (to decide whether or not to add it), the function pointer `evaluatePointSimilarity` must be made to refer to the new function in `updateSimilarityMethod`. Alternatively the call to `evaluatePointSimilarity` in `performRegionGrowing` can also be changed.

ii. To change the way the gradient images are calculated (for edge detection) make changes to `detectRemovedObjectsUsingGradientEdgeDetection`, `getThresholdedGradientImages` and `getCombinedThresholdedGradientImage`.

iii. A different method of abandonment analysis can be added by modifying the abandonment track bar in initWindow and adding the requisite parameters to `abandonment_struct` structure.

8. Blob filtering module: This one uses the class `BlobFilter` defined in `blobFilter.hpp/cpp`.

i. To alter the method of calculating the distance between two blobs, update the function `getSquaredDistance`. Existing methods can be altered by changing `getSquaredEuclideanDistance` or `findMinimumBoundingBoxDistance`.

ii. To change the criteria for finding a match between a new blob and an existing one, update findMatchingeObject. Alternatively, the individual functions `matchObjectLocation`, `matchObjectSize` and `matchObjectAppearance` can also be changed to update the corresponding criterion.

iii. To change the interface presented to the user for adding/removing candidate objects, changes are needed in `addCandidateRemovedObjects` and `getObject`.

iv. Any additional parameters can be added to filter_struct structure and can be made adjustable by adding the corresponding track bars in `initWindow`.