1. The first example is entertaining helping systems. The second example is usage of the technologies beneath for other applications. The third example is tracking features that could be used for tracking objects and thus for manipulating virtual objects with the real ones.
2. As far as I know, there are multiple options. From the OpenCV's perspective, there are at least a couple of them: Mean (threshold value is the mean of neighbourhood area) and Gaussian (threshold value is the weighted sum of neighbourhood values where weights are a gaussian window).
3. I would say that the thresholding algorithms are different and used for different purposes. However, I'd rather say that the good one should discard problems caused by illumination changes. In that section the binary global thresholding was used because the initial image was already thresholded with adaptive thresholding.
4. FAST was initially developed as an algorithm for real-time because SIFT and SURF were not efficient enough. BRIEF is just a feature extractor and matcher but a bit faster then others. It means that this tandem is quite efficient for real-time. In addition, ORB Oriented FAST and Rotated BRIEF (ORB) is a good option for that because contains improvements for both algorithms included.
5. As far as I know, the algorithm is an alternative for monocular SLAM thus it only uses a monocular camera and no dense sensors. Regarding the blur problem, authors claim that the availability of a dense scene model, all the time, enables many simplifications of issues with pointbased systems, for instance with regard to multiple scales and rotations, occlusions or blur due to rapid motion. How do they achieve this can read in their article and is out of the thesis scope.
6. Since the Sudoku solver was the first try with computer vision, at that time it has been done as follows. If the detection has failed, number is recognized incorrectly or detected where there is no number the solving will fail and that is it. Regarding the image quality, vast majority of modern smartphones and web cams are sufficient enough. The optimal resolution is 1024x768 or close to that. Illumination conditions and non-linear distortions cause more problems for the app.
7. Yes, I have tested under the bad lighting conditions. No it will not, cube has different colors on faces thus some of them will be detected correctly and some of them will fail. This will happen at face detection step. Color detection step is an unpredictable thing.
8. Yes, I do. Maybe it is possible to add some kind of back processing though if bootstrapping has failed play with parameters programmatically until the plane is found. On the other hand it can be mixed with the previous two approaches to achieve cube's corners computation but not for the face detection.
9. I did know how to solve the cube before this project. That was one of the reasons to pick the beginners method to solve the cube programmatically because I was able to test this approach gradually by myself.