






# Matlab-based Application for Image Processing and Analysis of Quality Assurance of SPECT Scanners

## Aplicação baseada em Matlab para Análise, Pós-processamento de Imagens e Controle de Qualidade de Imagens de SPECT

Gustavo L. Gonçalves<sup>1</sup>, Eduarda P. Grecco<sup>1</sup>, Carlos E. G. Oliveira<sup>1</sup>,  
Uysha S. Fonda<sup>2</sup>, Emerson N. Itikawa<sup>1</sup>

<sup>1</sup>Grupo de Pesquisa em Imagens e Inteligência Artificial, Instituto de Física/UFG, Goiânia-GO, Brasil

<sup>2</sup>Centro de Medicina Nuclear, Instituto de Radiologia - Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo, São Paulo, Brasil

### Resumo

O objetivo deste trabalho é aprimorar a experiência do usuário em um *toolkit* para pós-processamento de imagens na modalidade SPECT (Tomografia computadorizada por emissão de fóton único), tornando-o também livremente executável para não usuários do MATLAB. O *toolkit* foi desenvolvido dentro do MATLAB, usando o recurso de desenvolvimento de aplicativos autônomos, *MATLAB AppCompiler*, e escrito com o ambiente de aplicativos, *MATLAB AppDesigner*, na versão R2021a. Várias funcionalidades do aplicativo foram corrigidas ou melhoradas (bugs/falhas, problemas de otimização, aperfeiçoamento do *layout*), além de adicionar novos recursos e executar o aplicativo de forma totalmente independente do MATLAB. Em termos de desempenho, o *toolkit* tem um consumo de memória gráfica razoável, mesmo para as máquinas com diferentes especificações de sistema. O programa obteve sucesso em sua execução, além de cumprir sua finalidade para com a medicina nuclear, mostrando-se uma aplicação perfeitamente funcional, para máquinas com diferentes especificações de sistema.

**Palavras-chave:** MATLAB; SPECT; Tempo-morto; Controle de Qualidade; Medicina Nuclear; Processamento de Imagens, Física Médica

### Abstract

*The objective of this work is to improve the user experience within a toolkit for post-processing images in SPECT mode (Single Photon Emission Computed Tomography), making it also freely executable for non-MATLAB users. The toolkit was developed within MATLAB, using the standalone application development feature, MATLAB AppCompiler, and written with the application environment, MATLAB AppDesigner, in version R2021a. Several issues with the app's functionality were resolved (bugs, optimization issues, layout improvement), as well as adding new features and running the application completely independently of MATLAB. In terms of performance, the toolkit has a reasonable graphics memory consumption, even for machines with different system specifications. The program was successful in its execution, in addition to fulfilling its purpose for nuclear medicine, proving to be a perfectly functional application for machines with different system specifications.*

**Keywords:** MATLAB; SPECT; Dead time; Quality Control; Nuclear Medicine, Image Processing, Medical Physics

## 1. Introduction

MATLAB is a simplified programming environment developed by the company MathWorks for mathematical purposes in several areas, working with matrices (1). The versatility of this tool comes from the possibility of working using toolboxes, a set of functions that work around a specific objective.

Another important function of MATLAB is the possibility to create applications in an interaction environment outside your workspace. Initially, the development of applications took place in a graphical user interface called GUIDE (Graphical User Interface Designer), recently, the company gradually migrated to MATLAB AppDesigner, with more utilities in the creation of new apps.

One of the available toolboxes is the Image Processing Toolbox, which allows the acquisition and processing of images in various formats and extensions. Thus, the use of MATLAB for image processing in Nuclear Medicine is justified, since images contain matrix information, and inherent to medical images there is important information such as contrast, resolution and sharpness, which can be

analyzed mathematically. There are many tests, each with a specific frequency of performance, to evaluate some specific capabilities of the machine in use. In this work we implement functionalities for the following tests: Uniformity, Contrast and Dead Time.

The Uniformity test evaluates the intrinsic response of the scintillation chamber to a uniform flow of radiation over the FOV using a symmetrical window centered in the uniform region of the Phantom Jaszczak (2). In the same way, the contrast of the machine can be analyzed using the region of the spheres in the phantom.

In almost all radiation detector systems, there is a minimum time interval for the detection of two events, that is, in general, due to the processes inside the detector, some events can be lost in this time, which is called dead time (3). The dead-time test measures the ratio of count losses to detector dead-time at a high-count rate. The two-source method is commonly used method to estimate the dead-time of gas-filled detector system (4). The count rate is plotted as a function of source activity (5). The advantage of using MATLAB in these tests is the mathematical precision

and image processing power of the Image Tool toolbox.

The objective of this work was to implement an image processing application for quality control in nuclear medicine to be run independently of MATLAB, in order to improve the user experience, working with functionalities for Uniformity, Contrast and Dead Time tests in SPECT.

## 2. Materials and methods

The application development is divided into major and minor updates including mainly:

- update of the application development environment
- code errors/fails resolution (*bugs*)
- code optimization
- running the application regardless of MATLAB being installed on the user's machine, using the MATLAB App Compiler

All changes were made in version R2021.A. The machine configurations used were: Intel Core i5-4570 CPU, 16 GB RAM, Intel HD Graphics 4000 GPU.

## 3. Results

### 3.1 Graphical User Interface

The first major update applied to the code was the migration of the user interface from GUIDE to AppDesigner. In this step, some functions are not recognized, as they have been updated, so it was necessary to use the GuideToAppConvert function in all callbacks (elements that respond to user action). Still, some functions such as delete (used in the close plugin button) were implemented because they have a greater synergy with AppDesigner.

### 3.2 Optimization

In terms of code optimization, it has been done removing idle functions and migrating local and global scope variables to properties (thus using "app." to call the variable in any code callback). Another minor optimization was the reordering of the TAB key to switch between callback elements.

Figure 1 shows a bug that was related to the Close all windows button, which resulted in the application being re-opened in a loop, which was resolved by implementing the delete function in a "for" loop structure. In this way, the button closes all windows without additional memory cost.

```
% Button pushed function: closeWindows,
...closeWindows_2
function closeWindows_Callback(app,event)
% close all figure created
for j=1:app.n
delete (figure(j));
end
end
```

Figure 1 - code referring to the windows closing callback.

A detail in the image acquisition that could result in an unexpected bug was the fact that the dicomlookup function (function that investigates all DICOM image headers) limitedly inspects the image modality type, allowing the selection of any image from DICOM format. This was resolved by changing the selection method to exclusively choose a SPECT image, when checking the image's metadata, and returning an error window otherwise (Fig. 2).

```
if info.(dicomlookup('0008', '0060')) == 'NM'
showDicom(app, dicom);
f = msgbox(['Check the volume image, ' ...
' then press Uniformity to start'], ...
'Checking the volume');
drawnow
waitfor(f);
else
errorDlg('Dataset must be SPECT', 'ERROR');
pause;
end
```

Figure 2 - code that checks the type of acquisition modality.

With the application open, the Uniformity and Contrast test buttons were functional even without an image file loaded, resulting in a critical malfunction. To solve this, the most elegant solution was to add a new variable, called varcheck, created whenever the image to be worked on was chosen. When calling the following callbacks, the existence of this variable is always checked first, to ensure that the function will only be executed if there is material for it (in this case, the user's image). Otherwise, a pop-up window appears warning the user that he must previously choose an image.

UNIFORMITY TEST		
Average Counts of the 3 slices (of Figure 4):	6464.21 counts	
Standard deviation:	37.01 %	
Minimum Counts:	8824.00 counts	
Maximum Counts:	1.00 counts	
Center2Edge ratio:	1.02 counts	
Integral Uniformity (CFOV):	7.13 %	
Integral Uniformity (UFOV):	20.36 %	
Differential Uniformity (CFOV):	3.42 %	
Differential Uniformity (UFOV):	10.20 %	
CONTRAST RESOLUTION		
Reference Uniformity slice = 7619.79		
	RC max	RC average
Sphere 1	0.94	0.41
Sphere 2	0.92	0.32
Sphere 3	0.76	0.24
Sphere 4	0.52	0.18
Sphere 5	0.30	0.13
Sphere 6	0.21	0.09

Figure 3 - Text file with Uniformity and Contrast example results

Another optimization update was the creation of the "Open .txt report" button that instantly accesses the data resulting from the tests performed (Fig. 3), since previously this file was only created and stored in a less intuitive way in the application folder.

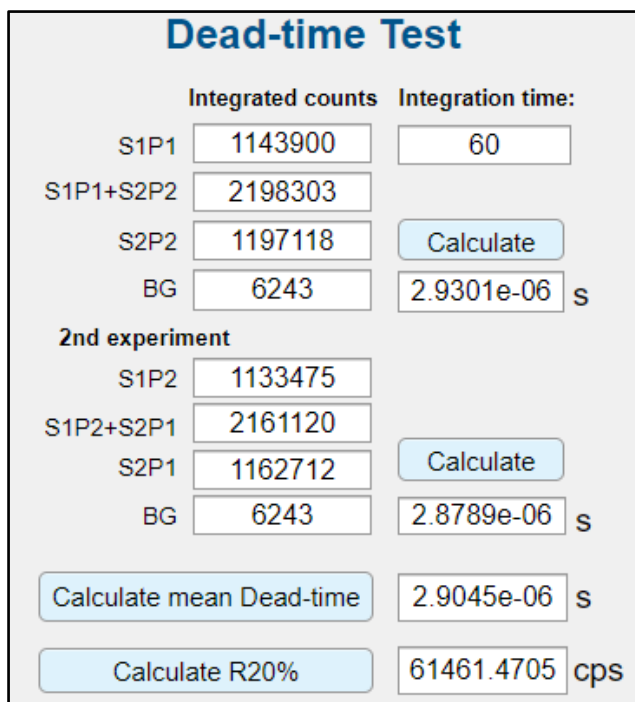


Figure 4 - Dead-time test example results, second tab

### 3.3 Functionality

As far as image processing is concerned, the main tab of the application works on two fronts: Uniformity and Contrast test.

For the correct use of the application, the user must select a SPECT test image in DICOM format and inform the phantom model used.

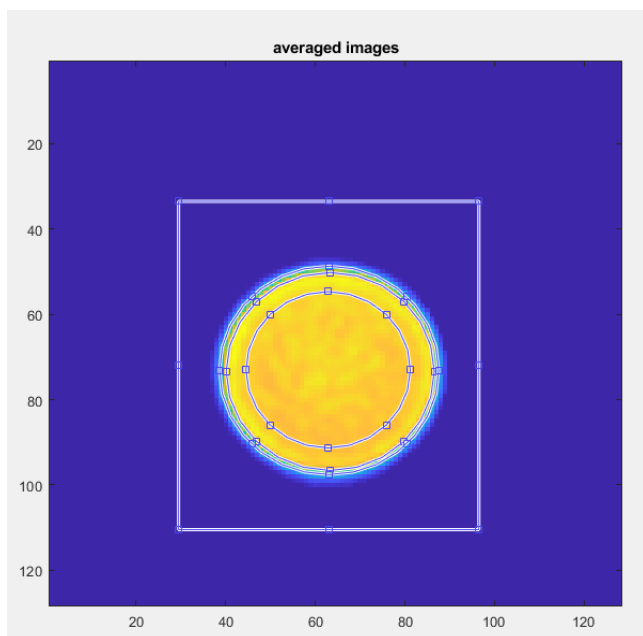


Figure 5 – Roi selection inside the uniformity test.

The uniformity test uses data collected in three different user-defined ROIs: two circular, with 75% and 95% of the phantom's cross-section radius, and one square, selected to optimize the 3D visualization provided by the application (Fig. 5).

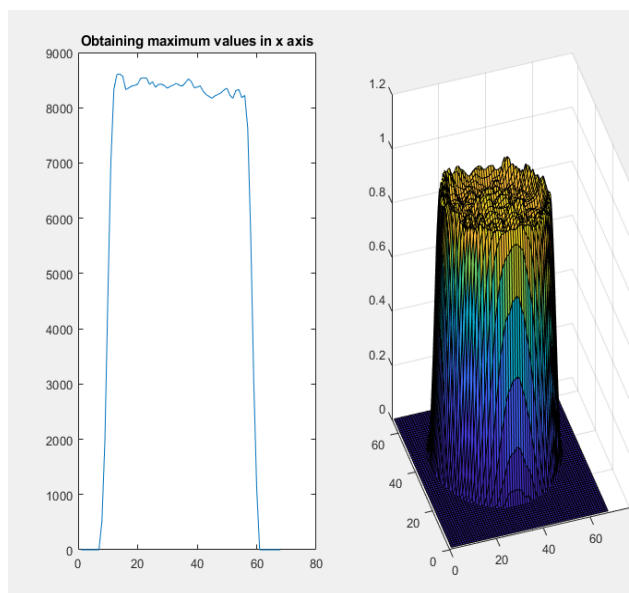


Figure 6 – Graphic representation of results

The contrast test considers the pixel data of the phantom's hot spheres (either standard or deluxe) for its calculations. The application asks for the selection of an ROI for each sphere (Fig. 6) to return the values as shown in figure 3.

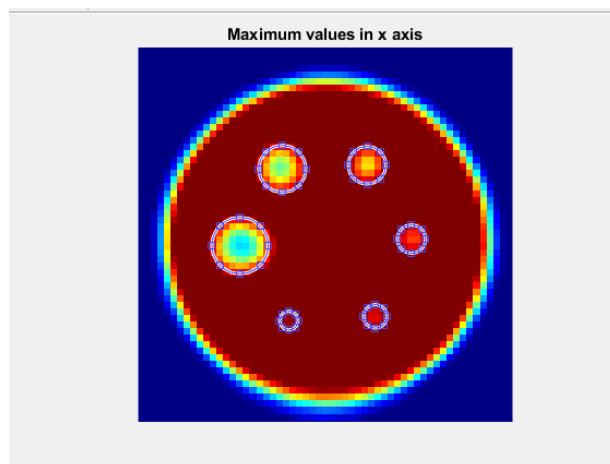


Figure 7 – Roi selection inside the contrast test.

The results can be instantly accessed by the button "Open .txt report".

### 3.4 Compilation

The ability to install and run the program on any computer was implemented, free of charge, as long as the installer file is available. This could be done within the AppDesigner itself, compiling the finished code and application, adding data such as version, loading screen, icon, developers and readme.txt, in addition to the MATLAB Runtime in the installed package, which allows the program to be executable on any computer, even if it does not have MATLAB installed.

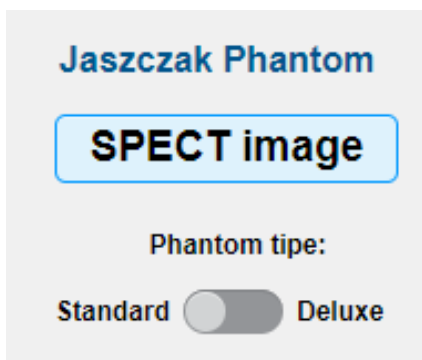


Figure 8 – Phantom type switch button.

Finally, new features were added such as: results summary button, switch button to choose between standard and deluxe phantom (Fig. 7), closing multiple windows, integration time selector and updated dialog windows.

### 3.5 Application performance analysis

Data collected from 6 different images in Uniformity of Contrast tests on two different machines, each with an evaluator, showed a relative error of 0.02% between them.

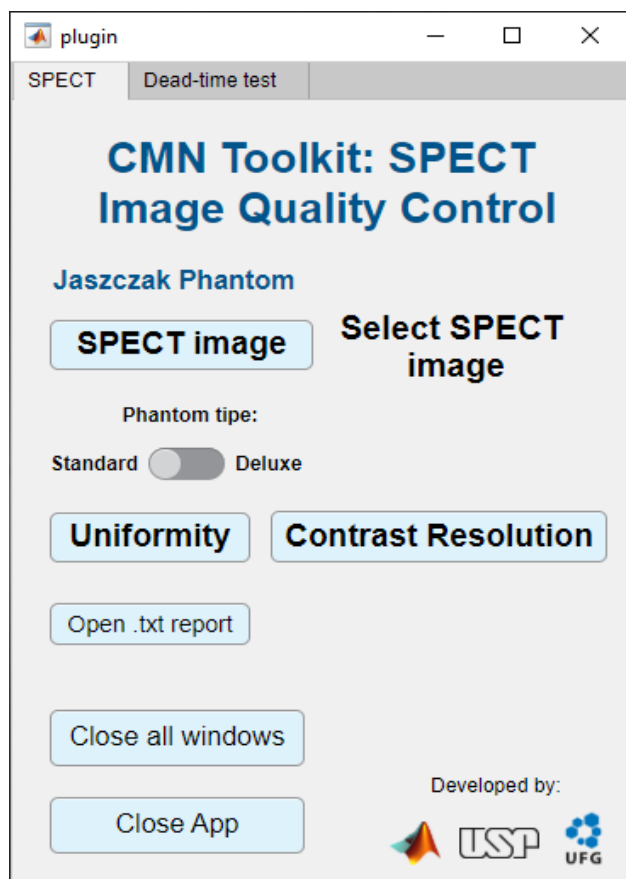


Figure 9 - Final layout of the toolkit's main tab.

Regarding the consumption of the machine's processing capacity, the maximum central processing used during the tests was 45% and an average of 15% (Only with the toolkit running in the foreground) while, for the operation inherent to MATLAB, it obtained a peak of 79% and an average of 51%.

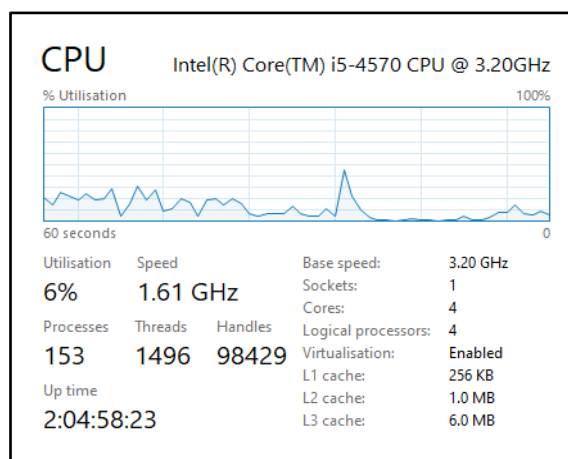


Figure 10 - Central processing data during testing.

## 4. Discussion

In this work, we update the graphical interface of an in-house application whose purpose is to control the quality of SPECT images. We also performed bug fixes and mainly made the application independent of MATLAB, which is an application whose license is paid for by the user.

The change in the graphical interface allowed greater versatility, as there is a tab to switch between the code and the app's layout, easily accessing callbacks and functions without the need to open additional windows, in addition to the aesthetic advantage in the application. This step also represented an optimization feature of the application, making it functional on machines with simpler hardware specifications. Overall, the functionality implementations applied represented a substantial improvement in the user experience.

The most important of the updates was the compilation of the application in a standalone (desktop application that is independent of the simultaneous execution of another program). This allows any potential user to install the program on their machine for free and independently.

The plugin provided by the IAEA, however, needs a third-party software (FIJI), based on JAVA, for its execution. In this first moment, this work proposed to update the interface and fix bugs related to the functioning of the application. The next step will be the validation of the results obtained by our application, in comparison with the values obtained by the IAEA plugin, the 'NM Toolkit', considering the implemented tests, inter-user comparison and the intuitiveness provided by the user-interface interaction.

## 5. Conclusions

The application did what it set out to do quickly and satisfactorily, obtaining reproducible and detailed results in the Uniformity, Contrast and Dead Time tests without consuming large amounts of memory and processing. In the future update package, we propose to implement the Spatial Resolution test. Application source code will be available upon reasonable request for academic purposes.

## References

1. Lyra M, Ploussi A, Georgantzoglou A. MATLAB as a Tool in Nuclear Medicine Image Processing. In: Ionescu, Clara, editors. MATLAB: a ubiquitous tool for the practical engineer. Rijeka: Intech; 2011. p. 477-500
2. International Atomic Energy Agency. Quality assurance for SPECT systems. Vienna: International Atomic Energy Agency; 2009.
3. Beers Y. A Precision Method of Measuring Geiger Counter Resolving Times. Rev Sci Instrum. 1942 Feb;13(2):72-6.
4. Akyurek T. A new dead-time determination method for gamma-ray detectors using attenuation law. Nucl Eng Technol. 2021 Dec;53(12):4093-7.
5. AAPM. Acceptance Testing and Annual Physics Survey Recommendations for Gamma Camera, SPECT, and SPECT/CT Systems The Report of AAPM Task Group 177 [Internet]. 2019 [cited 2022 Oct 13]. Available from: [https://www.aapm.org/pubs/reports/RPT\\_177.pdf](https://www.aapm.org/pubs/reports/RPT_177.pdf).

## Contact:

Gustavo Lopes Gonçalves.  
Universidade Federal de Goiás, Instituto de Física.  
Avenida Esperança, s/n, Campus Samambaia,  
74.690-900.  
gustavo.lopes@discente.ufg.br