UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

FABIANA TOON DE ARAÚJO

INTERFACE GRÁFICA EM PYTHON PARA AQUISIÇÃO E ANÁLISE DE IMAGENS EM DISPOSITIVO DE TERAPIA FOTODINÂMICA

São Carlos
2023

FABIANA TOON DE ARAÚJO

Interface Gráfica em Python para Aquisição e Análise de Imagens em Dispositivo de Terapia Fotodinâmica

Monografia apresentada ao Curso de Engenharia Elétrica, com ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do Título de Engenheira Eletricista.

Orientador: Prof. Dr. Sebastião Pratavieira

São Carlos
2023

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Araújo, Fabiana Toon
A658i Interface gráfica em Python para aquisição e análise de imagens em dispositivo de terapia fotodinâmica / Fabiana Toon Araújo; orientador Sebastião Pratavieira. São Carlos, 2023.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2023.

1. Terapia Fotodinâmica. 2. Câncer de Pele. 3. Interface Gráfica. 4. Monitoramento. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Fabiana Toon de Araujo

Título: "Avanços na aplicação e monitoramento da terapia fotodinâmica"

Trabalho de Conclusão de Curso defendido e aprovado em 13/12/2023,

com NOTA 9,3 (nove,três), pela Comissão Julgadora:

Prof. Dr. Sebastião Pratavieira - Orientador - IFSC/USP

Prof. Dr. Maximiliam Luppe - SEL/EESC/USP

Prof. Dr. Marlon Rodrigues Garcia - UNESP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Professor Associado José Carlos de Melo Vieira Júnior

Dedico este trabalho à minha mãe, ao meu pai, à minha irmã e ao meu irmão, que sempre foram a base de todos os caminhos que me fizeram chegar até aqui.

AGRADECIMENTOS

Agradeço, imensamente, a todas as pessoas que estiveram presentes de alguma forma durante minha trajetória na universidade. À minha família por serem meu porto seguro quando os tempos pareceram difíceis ao longo desses anos.

Agradeço à minha mãe, Rose, pelo exemplo, pelas palavras de carinho e cuidado, pela nossa família. Ao meu pai, Artur, por ter sempre as respostas para as minhas perguntas, por ter me inspirado a fazer esse curso, para quem sabe um dia eu possa ser a pessoa que tem as respostas das perguntas de alguém. À ambos, gostaria de agradecer pelo apoio e suporte, por me permitirem a oportunidade de estudar nessa universidade.

Agradeço a minha irmã, Gabriele, minha melhor amiga, por todas as vezes que me acolheu, pelas vezes que me fez companhia, mesmo a distância, e pela amizade. Agradeço ao meu irmão, Victor, meu melhor amigo, por todas as palavras de incentivo, pelos conselhos, e por dizer tantas vezes que eu precisava acreditar mais em mim, que eu era capaz.

Agradeço aos meus amigos, em especial ao Felps e ao Murilo, que estiveram comigo desde o primeiro dia, me acolheram, e seguraram a barra junto comigo durante esses anos de faculdade. Eu jamais chegaria onde cheguei e seria quem eu sou sem vocês.

Agradeço também à minha amiga Taís, que esteve comigo em todos os momentos, da graduação ao estágio, que me fez companhia durante a elaboração desse trabalho, nos momentos difíceis e também nas comemorações. Agradeço ao meu amigo João, pelos conselhos, pela amizade, pelos trabalhos em grupo, por tudo que me ensinou e por me obrigar a sair de casa pra ver a luz do dia de vez em quando.

Agradeço a Universidade de São Paulo e ao Departamento de Engenharia Elétrica e da Computação pela oportunidade de crescer academicamente. Agradeço também ao Centro de Pesquisa em Óptica e Fotônica (CePOF) e ao Instituto de Física de São Carlos (IFSC) pela viabilização de toda a pesquisa realizada durante a minha graduação, da Iniciação Científica ao Trabalho de Conclusão de Curso.

Agradeço imensamente ao professor Marlon, por todo o apoio ao longo de 4 anos de pesquisa, da minha iniciação científica à realização desse trabalho, por acreditar em mim desde o começo e por me ajudar a chegar até aqui.

Agradeço também ao meu orientador, professor Sebastião, pela oportunidade, pelo apoio e pelos ensinamentos.

RESUMO

ARAÚJO, F. T. **Interface Gráfica em Python para Aquisição e Análise de Imagens em Dispositivo de Terapia Fotodinâmica**. 2023. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

Atualmente, a Terapia Fotodinâmica (TFD) é um tratamento proeminente para lesões de câncer de pele do tipo não melanoma. Ela consiste na utilização de uma fonte de luz e um agente fotossensibilizante, que, juntamente com oxigênio, ocasionam dano celular e/ou tecidual. O ácido 5-aminolevulínico (5-ALA) é um agente tópico muito utilizado na TFD por ser capaz de induzir a síntese endógena de Protoporfirina-IX (PpIX) nas células, e assim, acarreta a apoptose e/ou necrose celular nas regiões onde a substância é acumulada, após a exposição à luz em determinado comprimento de onda. O sistema desenvolvido por Marlon Garcia é o único capaz de realizar o imageamento da PpIX em lesões de pele excitadas no vermelho e coletadas no início do infravermelho próximo (700 nm). Contudo, possui um sistema integrado de grande porte, controlado por um *software* em *Labview*, que acarreta certa dificuldade para se trabalhar em ambiente clínico pela dimensão dos equipamentos e periféricos, e pelo *software* desenvolvido em *Labview*, que possui limitações. O monitoramento em tempo real da TFD permite um maior controle dos fatores que contribuem para o sucesso do tratamento, visto que é um tratamento que é muito influenciado pelas condições particulares de cada paciente, lesão e tratamento. A fim de tornar o controle do tratamento um processo mais prático, robusto e funcional, foi desenvolvida uma nova interface gráfica, utilizando o *software QtDesigner* e a biblioteca *PyQt5*, da linguagem *Python*. Essa interface possui funcionalidades para controle de usuário, controle de tempo de tratamento, visualização de imagens da lesão e disposição da curva de decaimento da fluorescência em tempo real, e aquisição de imagens. Tal interface pode ser integrada num sistema embarcado, o que permite a utilização de microprocessadores, e consequentemente, a redução da dimensão do equipamento; e também pode ser integrada à funcionalidades complexas de processamento com a utilização da linguagem *Python*. Dessa forma, a manipulação e análise de dados, em tempo real, obtidos durante o tratamento viabiliza a criação de um protocolo customizado adaptado às necessidades de cada paciente, que possuem conjuntos de parâmetros fisiológicos e ópticos específicos.

Palavras-chave: Terapia Fotodinâmica. Câncer de Pele. Interface Gráfica. Monitoramento.

ABSTRACT

ARAUJO, F. T. **Python Graphic Interface for Acquisition and Analysis of Images in Photodynamic Therapy Device**. 2023. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

Currently, Photodynamic Therapy (PDT) is a prominent treatment for non-melanoma skin cancer lesions. It involves the use of a light source and a photosensitizing agent, which, together with oxygen, cause cellular and/or tissue damage. 5-aminolevulinic acid (5-ALA) is a commonly used topical agent in PDT as it can induce the endogenous synthesis of Protoporphyrin-IX (PpIX) in cells, leading to apoptosis and/or cellular necrosis in regions where the substance accumulates after exposure to light at a specific wavelength. The system developed by Marlon Garcia is the only one capable of imaging PpIX in skin lesions excited in the red and collected in the early near-infrared (700 nm). However, it has a large integrated system controlled by LabVIEW software, which poses some challenges for clinical use due to the size of the equipment and peripherals, as well as the limitations of the LabVIEW software. Real-time monitoring of PDT allows greater control over the factors contributing to the treatment's success, as it is highly influenced by the specific conditions of each patient, lesion, and treatment. In order to make treatment control a more practical, robust, and functional process, a new graphical interface was developed using the QtDesigner software and the PyQt5 library from the Python language. This interface includes user control functionalities, treatment time control, visualization of lesion images, real-time fluorescence decay curve display, and image acquisition. Such an interface can be integrated into an embedded system, allowing the use of microprocessors and consequently reducing the equipment's size. It can also be integrated with complex processing functionalities using the Python language. Thus, real-time manipulation and analysis of data obtained during treatment enable the creation of a customized protocol adapted to each patient's needs, considering specific physiological and optical parameter sets.

Keywords: Photodynamic Therapy. Skin Cancer. Graphical Interface. Monitoring.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

TFD	Terapia Fotodinâmica
5-ALA	Ácido 5-aminolevulínico
PpIX	Protoporfirina-IX
ROS	Espécies Reativas de Oxigênio

SUMÁRIO

1 INTRODUÇÃO

A terapia fotodinâmica (TFD) é uma modalidade terapêutica que envolve a utilização de uma fonte de luz e um agente fotossensibilizante para causar danos celulares e/ou teciduais, num processo diretamente dependente do oxigênio molecular (DOLMANS; FUKUMURA; JAIN, 2003). O agente fotossensibilizante é o responsável por transferir energia da luz para o oxigênio, desde que a luz possua o comprimento de onda dentro do espectro de absorção do agente fotossensibilizante (YOUNUS et al., 2023). O processo fotodinâmico possui dois mecanismos diferentes, um deles leva a formação de espécies reativas de oxigênio (ROS), que ocasionam a morte de células cancerígenas através de estresse oxidativo, e o outro leva a formação de Oxigênio Singleto, que possui alta capacidade oxidativa (YOUNUS et al., 2023).

A terapia fotodinâmica tópica é um método utilizado para tratar câncer de pele do tipo não melanoma (BRAATHEN et al., 2007). É uma tecnologia brasileira, desenvolvida nas universidades, e que foi aprovada como sugestão de tratamento pelo Sistema Único de Saúde (SUS) (Ministério da Saúde, 2023). As porfirinas foram os fotossensibilizadores mais estudados, e somente em meados dos anos 1950 que estudos começaram a apontar o uso de porfirinas fluorescente para a localização de tumores em humanos (DOLMANS; FUKUMURA; JAIN, 2003; ACKROYD et al., 2001). Cinquenta anos depois, em 1990, as descobertas de Kennedy e Pottier transformaram os rumos da terapia fotodinâmica através da utilização de uma substância tópica capaz de induzir a produção de porfirinas endógenas (DOUGHERTY et al., 1998). A substância denominada ácido 5-aminolevulínico (5-ALA), exposta a condições apropriadas, é capaz de induzir a síntese do fotossensibilizador denominado Protoporfirina-IX (PpIX) (KENNEDY; POTTIER, 1992). Com a exposição à luz, nos locais onde há a presença dessas substâncias fotossensíveis, pode ocorrer a morte de células tumorais através da apoptose ou necrose celular, causada pelas espécies reativas de oxigênio geradas pelo efeito fotodinâmico (YOUNUS et al., 2023).

Os tratamentos com terapia fotodinâmica podem ser bastante complexos dada a quantidade de fatores que podem influenciar a eficácia do tratamento (POGUE et al., 2016). Dentre eles, podemos citar a irradiância utilizada (intensidade de luz), fluência de luz (dose de luz), as propriedades ópticas do tecido-alvo, e os parâmetros fisiológicos da pele (CASTANO; DEMIDOVA; HAMBLIN, 2004).

A protoporfirina-IX possui uma grande de absorção, chamado de Banda de *Soret*, com seu máximo em, aproximadamente 405 nm, e outros picos menores, chamados de bandas Q em comprimentos de onda maiores, com máximos de absorção em, aproximadamente, 510, 545, 580 e 630 nm (CALZAVARA-PINTON; VENTURINI; SALA, 2007). A emissão de fluorescência da PpIX possui dois picos, em 635 nm e 700 nm, a depender do comprimento da luz absorvido. Em aplicações onde se quer detectar imagens de fluorescência da PpIX nas lesões da pele, a luz vermelha (633 nm) não é comumente utilizada, visto que,

em seu espectro de fluorescência, o pico de excitação (absorção) e o pico de emissão da fluorescência têm uma sobreposição significativa, o que dificulta sua separação, e consequentemente, dificulta a visualização da fluorescência. Por essa razão, a excitação da Protoporfirina-IX é mais comumente realizada em, aproximadamente, 409 nm, por meio de diodos laser violeta que emitem comprimentos de onda em torno de 405 nm (CELLI et al., 2010).

Embora os picos de absorção sejam menores nas bandas Q, o uso de comprimentos de onda maiores na Terapia Fotodinâmica, entre, aproximadamente, 620 nm-640 nm de vermelho, permite uma maior penetração da luz na pele com absorção e espalhamento reduzidos (CALZAVARA-PINTON; VENTURINI; SALA, 2007; CELLI et al., 2010). Dos equipamentos capazes de fazer o imageamento da PpIX em lesões de pele, excitadas no vermelho e com emissão de fluorescência a 700 nm, o único capaz de tratar e monitorar imagens de fluorescência da lesão em tempo real é o protótipo descrito por Garcia et al. (2019), que utiliza a mesma fonte de excitação, tanto para o tratamento quanto para o imageamento, o que possibilita que ambos sejam feitos ao mesmo tempo. Tal equipamento é dividido em três estruturas principais, como é mostrado no esquemático da figura 1 abaixo.

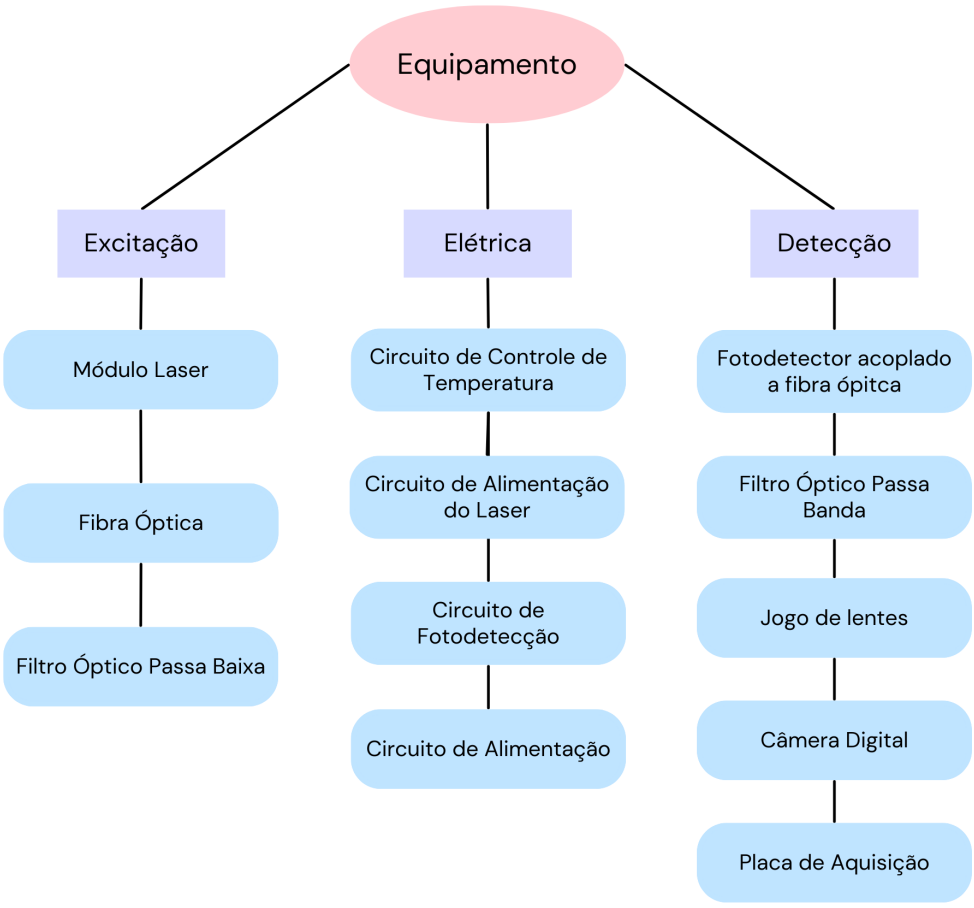


Figura 1 – Diagrama esquemático das partes do equipamento descrito por Garcia et al. (2019).

O sistema desenvolvido por Garcia et al. (2020) excita a região da lesão no vermelho (633 nm) e coleta imagens da fluorescência no início do infra-vermelho próximo (700 nm).

Por conta disso, o volume de tecido excitado é maior que equipamentos convencionais, que utilizam a excitação violeta, e mesmo ao final do tratamento ainda é possível observar a fluorescência no interior da lesão. Tanto o desenvolvimento do protótipo, quanto os resultados de seu experimento em modelo animal, já foram publicados no *Journal of Photodiagnosis and Photodynamic Therapy* (GARCIA et al., 2020).

A Terapia Fotodinâmica possui o benefício de ser um método de tratamento eficiente e não-invasivo para tratamento de câncer de pele do tipo não-melanoma. Contudo, uma das desvantagens está no fato do tratamento ter um protocolo fixo de execução, independente das características individuais de cada paciente e do tecido tumoral, da localização, idade do paciente, e do fototipo. (AKOPOV et al., 2017). Para garantir a eficácia do tratamento, o fotossensibilizador deve ter alta absorção no comprimento de luz utilizado, deve alcançar tecidos profundos para garantir a abrangência de toda a lesão, deve ser bem distribuído na lesão e ter uma alta razão entre sua concentração em tecidos lesionados e tecidos normais (MORTON et al., 2002). Atualmente, o maior desafio enfrentado na aplicação de terapias como a TFD é no desenvolvimento de uma forma de monitorar em tempo real o tratamento, de forma que seja possível utilizar estratégias, durante a execução da terapia, para combater possíveis respostas insatisfatórias no tratamento (CELLI et al., 2010).

Nesse sentido, no sistema desenvolvido por Garcia et al. (2020) é possível acompanhar a intensidade da fluorescência no interior das lesões. Definindo uma região de interesse (ROI) que engloba toda a região da lesão, e uma segunda região de interesse que inclui somente a região central, é possível observar, na figura 2, o decaimento da fluorescência no sistema comercial (em azul) e no sistema desenvolvido por Garcia et al. (2020) (em vermelho).

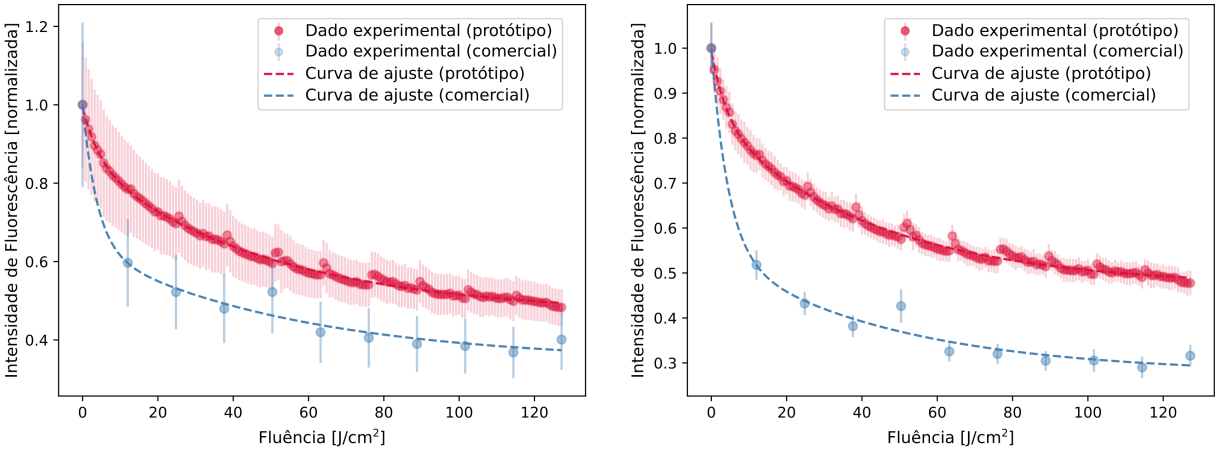


Figura 2 – Intensidade de fluorescência para todo o tumor (a), e para a região central do tumor (b), adquirida com o sistema desenvolvido por Garcia et al. (2020) (pontos vermelhos), e com o sistema comercial (pontos azuis). As curvas de ajuste calculadas por uma equação bi-exponencial, pelo método dos mínimos quadrados, também são mostradas (GARCIA et al., 2020).

Observando o decaimento da intensidade de fluorescência, é possível inferir que o de-

caimento no sistema comercial acontece mais rapidamente do que no sistema desenvolvido por Garcia et al. (2020), estando de acordo com o fato de que o volume de tecido excitado por esse equipamento é maior que em equipamentos convencionais. O monitoramento realizado é capaz de extrair informações do tratamento, como as curvas de decaimento, da intensidade de fluorescência, e das regiões onde a PpIX foi sintetizada, de forma que é possível inferir sobre o efeito do tratamento na lesão que está sendo tratada.

A equação que descreve a intensidade de fluorescência na região da lesão é dada por (GARCIA et al., 2020):

$$F(t) = F_{0f}e^{-t/\tau_f} + F_{0s}e^{-t/\tau_s} + I_{bg}, \tag{1.1}$$

no qual $F(t)$ é a intensidade de fluorescência num tempo t, em segundos, F_{0f} e F_{0s} são as intensidades de fluorescência inicias do decaimento rápido e lento, respectivamente, e I_{bg} é a intensidade do sinal de fundo adquirido (*background*). τ_f e τ_s são os tempos médios de decaimento rápido e lento, respectivamente. O tempo médio de decaimento total (τ) é calculado através da média entre τ_f e τ_s, ponderada pelos valores de F_{0f} e F_{0s}.

Medir a intensidade da fluorescência, por exemplo, permite fazer correções imediatas durante o tratamento caso haja perda de intensidade de fluorescência muito rápida ou lentamente. A perda da intensidade da fluorescência leva a diminuição da eficácia da terapia fotodinâmica, visto que a substância fotossensibilizadora está sendo consumida e o oxigênio do tecido está sendo esgotado (CELLI et al., 2010). Dessa análise, é possível inferir sobre o tratamento específico de cada paciente, de acordo com a resposta de cada organismo, do tipo de lesão, da forma como o fotossensibilizante foi administrado, e a partir disso, propor mudanças em tempo real no tratamento.

O equipamento desenvolvido por Garcia et al. (2020) possui um sistema integrado de grande porte, contendo uma ponta de prova responsável pelo imageamento e direcionamento do laser, um laptop onde um *software* em *Labview* controla o tratamento através do gerenciamento dos dispositivos periféricos, e uma caixa preta contendo toda a parte elétrica, o módulo laser e a placa de aquisição. Tal estrutura acarreta certa dificuldade para se trabalhar em ambiente clínico, tanto pela dimensão dos equipamentos e periféricos, quanto pelo *software* desenvolvido em *Labview* que exige um pós processamento das imagens adquiridas durante o tratamento, o que faz com que a análise do imageamento das lesões não seja feita durante o tratamento.

Outro ponto a ser destacado na manipulação do equipamento é a forma como ocorre o controle do módulo laser. Atualmente, o controle do laser é feito de forma analógica, e seria vantajoso que o controle da intensidade óptica do laser fosse feito através de um sistema digital com eletrônica chaveada. Por essa razão, foi desenvolvido um trabalho acadêmico, de autoria de Igor Cordeiro Santa Bárbara, que projeta um *driver* para controle do módulo laser, mas que não foi integrado à interface gráfica do sistema (BÁRBARA, 2020).

A fim de tornar o controle do tratamento um processo mais prático, robusto e funcional, é possível utilizar o *software* Qt, cuja principal utilidade está no desenvolvimento de interfaces gráficas para sistemas embarcados. O desenvolvimento dessa interface é condição para o embarcamento do sistema, visto que, ela é integrada, através da linguagem *Python*, a qualquer processamento que se queira para conferir funcionalidades à ela. Assim, utilizando de todas as vantagens que a linguagem *Python* oferece, é possível proporcionar o embarcamento do sistema, o que garantiria redução considerável do peso e das dimensões do sistema, além de permitir o controle do *driver* do *laser*, a estimação dos parâmetros da equação 1.1, e o uso de inteligência artificial para estimar resultados dos tratamentos a partir dos parâmetros calculados.

1.1 Objetivos

Como objetivo geral, nesse trabalho pretende-se aprimorar o sistema de monitoramento e tratamento da terapia fotodinâmica descrito por Garcia et al. (2020), de forma a se desenvolver uma interface gráfica para controle e automatização do sistema, por meio da linguagem de programação *Python*. Para o desenvolvimento da interface gráfica de controle do sistema será utilizado o *software QtDesginer* juntamente com a biblioteca *PyQt5* da linguagem *Python*.

Como objetivos específicos, buscou-se desenvolver os tópicos listados a seguir:

- Desenvolver habilidades no uso de *PyQt5* e *QtDesigner* para a implementação de interfaces gráficas;
- Integrar funcionalidades à interface utilizando a linguagem *Python* e a biblioteca *PyQt5*;
- Controlar uma câmera, a princípio com uma *webcam*, para mostrar imagens das lesões de pele direto na interface gráfica;
- Realizar o controle de usuários da interface para separar as informações de cada tratamento, através de ferramentas da linguagem *Python*;
- Capturar imagens durante o tratamento e armazená-las na respectiva seção de cada paciente;
- Armazenar dados do tempo de aquisição de cada imagem adquirida durante o tratamento;
- Imprimir em um gráfico em tempo real a média dos valores dos *pixels* da região da lesão.

A seguir, nos próximos capítulos serão tratados os materiais utilizados para desenvolvimento desse trabalho, os detalhes da implementação, os resultados obtidos e a respectiva discussão, e a conclusão. Por fim, uma seção para detalhar participação em eventos e publicações de trabalhos que foram frutos das pesquisas realizadas para elaboração desse trabalho.

2 MATERIAIS E MÉTODOS

O *Qt* é uma estrutura multiplataformas desenvolvida em C++ que permite a implementação de interfaces gráficas, na qual um único conjunto de algoritmos pode ser utilizado em diversos sistemas operacionais e embarcados (BLANCHETTE; SUMMERFIELD, 2006). Atualmente, diversos ambientes *desktop*, interfaces embarcadas e *mobile*, e aplicativos possuem as ferramentas do *Qt* como base do desenvolvimento, como *Ableton Live, Adobe Photoshop, Autodesk Maya, EAGLE, Google Earth, Spyder*, entre outros (Wikipedia contributors, 2023). Assim, essa plataforma pode contribuir para a elaboração do projeto, de forma que, possibilita o desenvolvimento de interfaces gráficas modernas e de fácil utilização, e também um controle mais eficiente e mais integrativo do equipamento e periféricos do sistema.

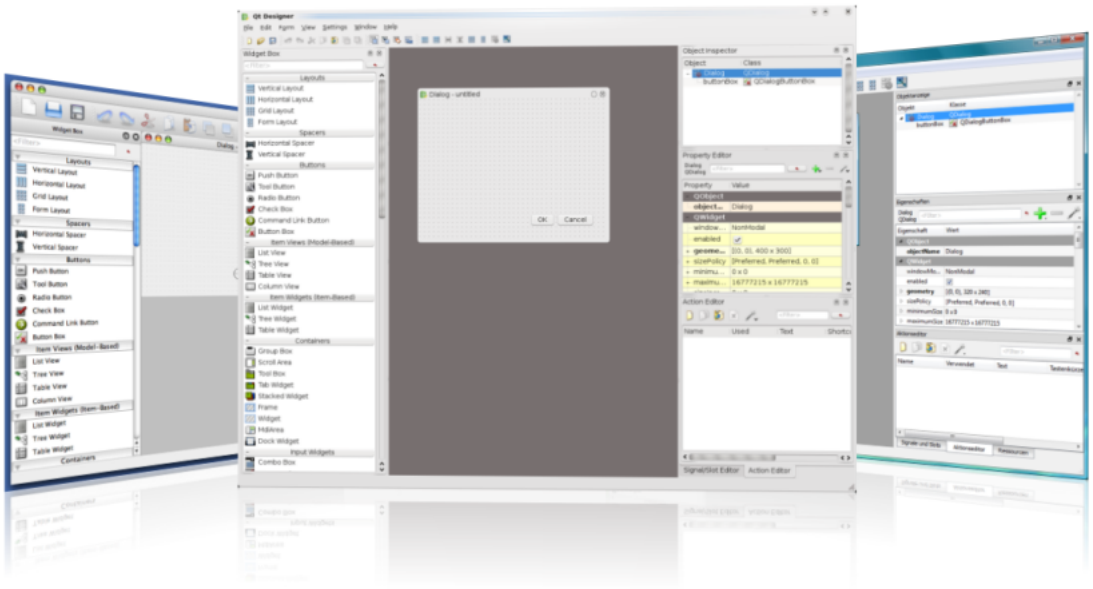
PyQt5, por sua vez, é a biblioteca que possibilita a ligação entre a linguagem *Python* e o *Qt*. Essa biblioteca possui mais de 35 módulos, que permitem que o *Python* seja uma linguagem alternativa ao C++ para o desenvolvimento de aplicações. Dessa forma, combinando a linguagem *Python* com o *Qt*, se incorpora as vantagens do conjunto de ferramentas C++ para criar aplicações, como a facilidade de criar interfaces flexíveis, com a simplicidade e a grande variedade de módulos que já existem no *Python* (WILLMAN; WILLMAN, 2021).

A construção de interfaces pode ser feita de duas formas, utilizando o *QtDesigner*, ou totalmente por vias de código. A ferramenta *QtDesigner*, mostrada na figura 3, é um *software* da *The Qt Company* (Espoo, Uusimaa, Finlândia) cuja funcionalidade se destina exclusivamente ao desenvolvimento de interfaces gráficas baseadas em *QtWidgets*. Os *QtWidgets* são um conjunto de *widgets* disponibilizados pelo *Qt*, e são elementos primários na criação de interface, podendo exibir dados e informações de status, receber dados do usuário e fornecer um contêiner para outros *widgets* que devem ser agrupados. A ferramenta do *Qt* possibilita a elaboração de interfaces através do modo "arrastar e soltar", de forma que, o próprio usuário, através do posicionamento dos *widgets* na interface, customize e componha a interface gráfica, aumentando a eficiência e rapidez no desenvolvimento.

No desenvolvimento de interfaces, o *widget* principal que compõe qualquer interface é o *QMainWindow*, mostrado na figura 4. Essse *widget* possui sua própria topologia, na qual é possível inserir componentes, e, portanto, tem a função de agrupar outros *widgets* que comporão a interface gráfica. A partir dele, é possível criar qualquer interface inserindo os *widgets* manualmente na tela para atender aos requisitos do projeto. Os *widgets* disponibilizados pelo *QtDesigner* podem ser vistos na figura 5.

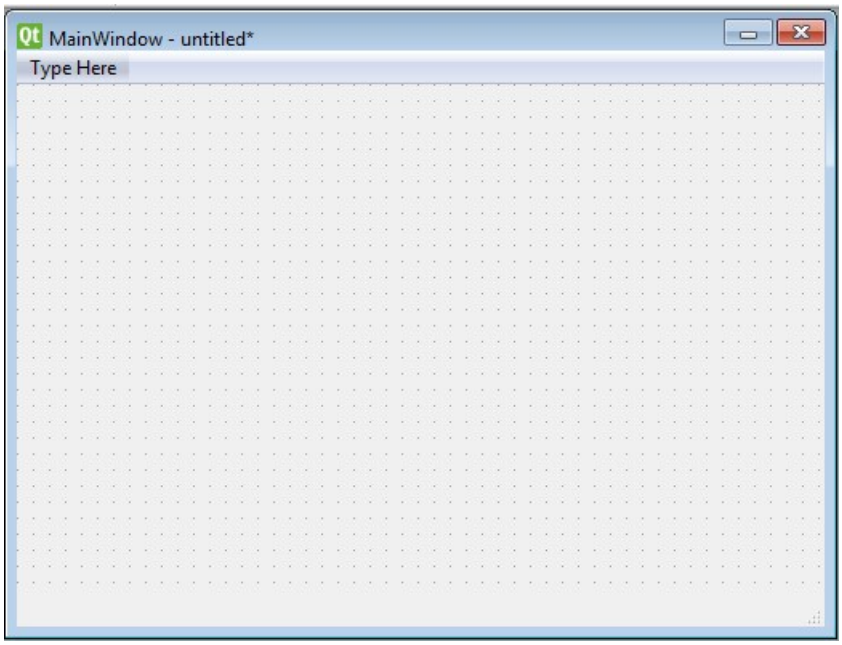
A biblioteca *PyQt5*, por sua vez, possui diversas extensões que permitem a utilização do *QtDesigner* via código em *Python*, tornando possível a utilização de interfaces gráficas, criadas com o *QtDesigner*, em projetos desenvolvidos em *Python*. Para dar funcionalidade aos *widgets*, faz-se necessário também a utilização de bibliotecas *Python*, que possibilitam

Figura 3 – *Software QtDesigner* utilizado para o desenvolvimento da interface gráfica.



Fonte: Imagem retirada do *website qt.io.*

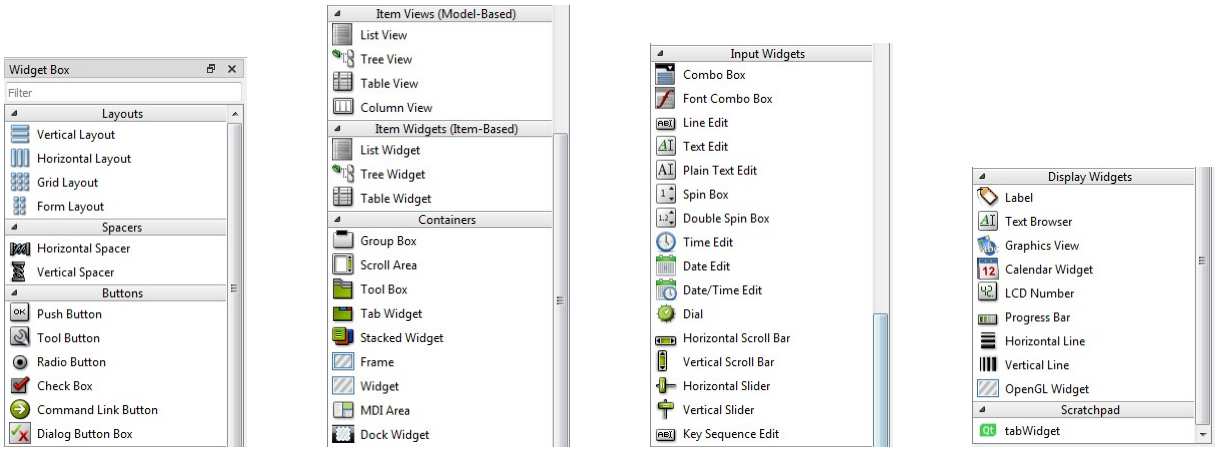
Figura 4 – Layout do *QMainWindow*.



Fonte: *QtDesigner*.

que a interface tenha capacidade de utilizar ferramentas da linguagem para desempenhar processos mais complexos.

Figura 5 – *Widgets* disponíveis no *QtDesigner* para serem adicionados na *MainWindow*.



Fonte: *QtDesigner*.

3 IMPLEMENTAÇÃO

Como proposto anteriormente, o desenvolvimento da interface gráfica se baseia na utilização do *software Qt Designer*, e na utilização da linguagem *Python* para conferir a ela funcionalidades.

3.1 Interface Gráfica

O desenvolvimento da interface e seu *layout* foi feito utilizando o software *Qt Designer* e se inicia com a criação de uma *MainWindow*, que consiste na janela principal onde todos os elementos da interface são contidos. Cada funcionalidade da interface é definida através de um *widget* disponibilizado na janela principal, e toda a parte de captação de informações, processamento, e disponibilização de resultados na interface é feita através da linguagem *Python*.

Na parte de controle de usuário, é necessário utilizar *widgets* que recebem informações externas, como o *QLineEdit* e o *QDateEdit*. Nesses *widgets*, o usuário é o responsável por fornecer as informações que posteriormente são utilizadas para o gerenciamento do tratamento de cada paciente.

Para disponibilização em tempo real da imagem da lesão adquiridas pela câmera, o *widget QLabel* foi utilizado na tela principal. Esse elemento é capaz de incorporar diversos tipos de objetos que são responsáveis pela manipulação de imagens, como o *QPixmap*, cuja função é disponibilizar imagens na tela de forma otimizada.

Outro objeto muito comum no desenvolvimento de interfaces, e que foi muito utilizado nesse projeto, é o *QPushButton*. Esse objeto dá comandos para o computador performar determinadas ações, e pode ser útil para controlar o tratamento.

Para integrar as funcionalidades da biblioteca *matplotlib* do *Python* à interface, para que seja possível imprimir gráficos em tempo real, foi utilizado um *widget* básico, denominado *QWidget*, que deve ser promovido a um *mplwidget*. Assim, quando promovido, ele passa a ter a configuração de um gráfico típico do *matplotlib*.

Depois de finalizado o desenvolvimento da interface, é criado um arquivo, de extensão *.ui*, que deve ser carregado no algoritmo em *Python* no qual as funcionalidades da interface serão tratadas. Nesse ponto, a biblioteca responsável pelas conexões entre a linguagem e o Qt é a *PyQt5*.

Manter a interface responsiva e em pleno funcionamento exige que ela seja a execução principal do programa, ou seja, todos os processamentos realizados ao mesmo tempo devem ser processos paralelos, para que eles não interrompam o processamento da interface principal. À seguir, são detalhados os processos paralelos que a interface deve realizar e como esses processos podem ser desempenhados.

3.2 Controle de Usuário

Como o tratamento fotodinâmico das lesões de pele deve ser monitorado para cada paciente, para que futuramente possa existir um protocolo customizado, é possível fazer um controle de usuário através de informações como nome do paciente e data de realização da sessão do tratamento.

Os dados são inseridos na interface pelo usuário, e, a partir deles, é possível criar diretórios nos quais todos os arquivos gerados, referentes ao mesmo paciente, podem ser armazenados para acompanhamento da evolução do tratamento.

Para manipular pastas e arquivos, e interagir com o sistema operacional, utilizou-se a biblioteca *os* do *Python*, juntamente com a biblioteca *datetime*, que viabiliza a manipulação de dados como a data do tratamento realizado.

3.3 Controle da Câmera

A fim de acompanhar o avanço do tratamento e o consumo do fármaco na região da lesão, utiliza-se uma câmera para capturar imagens e mostrá-las em tempo real na tela. Nesse caso, utilizou-se uma *webcam* para a aquisição das imagens, para simplificação do projeto. Porém, a câmera monocromática originalmente utilizada no sistema de Garcia et al. (2019) pode ser utilizada no lugar da *webcam* para a aquisição das imagens de fluorescência no infravermelho próximo. O *OpenCV* é uma biblioteca dedicada a computação visual em tempo real, e portanto, nesse trabalho, essa biblioteca foi utilizada para controlar as funcionalidades da *webcam* e a aquisição de imagens.

Para manter a interface gráfica responsiva, ao mesmo tempo que outras tarefas de longa duração são executas, como manter a aquisição de imagens em tempo real durante todo o tratamento, é necessário fazer o uso de *threads*. Separar essas tarefas em *threads* garante fluxos separados de execução, de forma que, enquanto o programa mantém a câmera em funcionamento mostrando uma imagem a cada atualização de *frame*, ainda é possível interagir com a interface gráfica sem que esta congele e tenha suas funcionalidades interrompidas. O mesmo serve para as demais tarefas a serem executas pelo programa, como controle de câmera e aquisição de imagens, processamento de imagem e dados, controle de tempo e armazenamento de dados, cada fluxo de informação deve estar localizado em uma *thread*, que podem se relacionar ou não, para que nenhuma afete o funcionamento das demais e funcionem simultaneamente à *thread* principal, que mantém a interface em funcionamento.

A biblioteca *PyQt5* possui seu próprio módulo de gerenciamento de *threads*, denominado *QThread*. A *thread* principal de execução é a própria *thread* que lida com a interface, e é essa que é sempre inicializada quando o programa é executado. As demais *threads* são inicializadas quando se fazem pertinentes ao longo da execução do programa, e são desenvolvidas em *threads* individuais que herdam do objeto *QThread*. O controle da câmera

deve ser feito em uma *thread* separada, e portanto, em uma classe diferente da principal na qual a interface é definida.

Para que haja comunicação entre as duas classes diferentes, a da câmera e a da interface gráfica, faz-se necessário o uso do sistema de *Slots* e *Signals* disponibilizadas pelo Qt. Os *widgets* têm a capacidade de emitir sinais a medida que determinado evento ocorre, e esses sinais são recebidos e performam ações de acordo com o que foi definido em seus respectivos *slots*. Assim, a cada sinal emitido, uma informação pode ser enviada para fora da *thread*, e é recebida pelo seu *slot* onde vai ser tratada.

3.4 Controle de Tempo de Armazenamento das Imagens

Para analisar a evolução do tratamento e armazenar dados de cada paciente, algumas das imagens das lesões geradas durante o tratamento podem ser armazenadas nos diretórios criados na seção de controle de usuário. Além de salvar os arquivos de imagens das lesões, também é necessário o armazenamento de dados do tempo no qual essas imagens foram capturadas, e o controle do tempo decorrido do tratamento.

Desta forma, uma nova *thread* deve ser implementada, de forma que ela realize a contagem do tempo, enquanto a interface se mantém em execução, e salve as informações de tempo relevantes para o tratamento. Para conectar essa *thread* com a *thread* principal, utiliza-se o mesmo sistema de *Slots* e *Signals*.

Para realizar a contagem do tempo e o armazenamento dos dados adquiridos num arquivo csv, utilizou-se a biblioteca *time* do *Python* juntamente com a biblioteca *csv*. Para salvar as imagens adquiridas a cada intervalo de tempo, utiliza-se a biblioteca *OpenCV*.

3.5 Cálculo do Decaimento da Fluorescência e Impressão em Tempo Real

Depois de salvar a imagem da lesão no sistema de aquivos, é possível calcular, a partir dessa mesma imagem, o decaimento da fluorescência, e imprimir num gráfico o decaimento em função do tempo decorrido do tratamento.

O cálculo para se obter o gráfico de decaimento da fluorescência é feito a partir de uma função da biblioteca *numpy* do *Python*, que calcula a média da intensidade dos *pixels* da imagem. Os valores então obtidos devem ser salvos em um novo arquivo csv, utilizando a biblioteca *csv*, que será utilizado como base de dados para a impressão do gráfico de decaimento da fluorescência.

De posse dos valores da média dos *pixels* das imagens, e dos valores do intervalo de tempo na qual cada imagem foi obtida, é possível plotar o gráfico no *widget* correspondente. As funções que lidam com o gráfico em tempo real devem estar numa *thread* separada, na qual todos os métodos e funcionalidades do *widget* estejam definidos. Assim, fica garantido que as execuções sejam feitas de forma paralela e a interface se mantenha responsiva enquanto a impressão é feita.

Para fazer a leitura dos arquivos csv para construção do gráfico, utiliza-se a biblioteca *Pandas* do *Python*, que permite não apenas a leitura do arquivo csv, como a manipulação dos dados para que seja possível utilizá-los diretamente na impressão do gráfico.

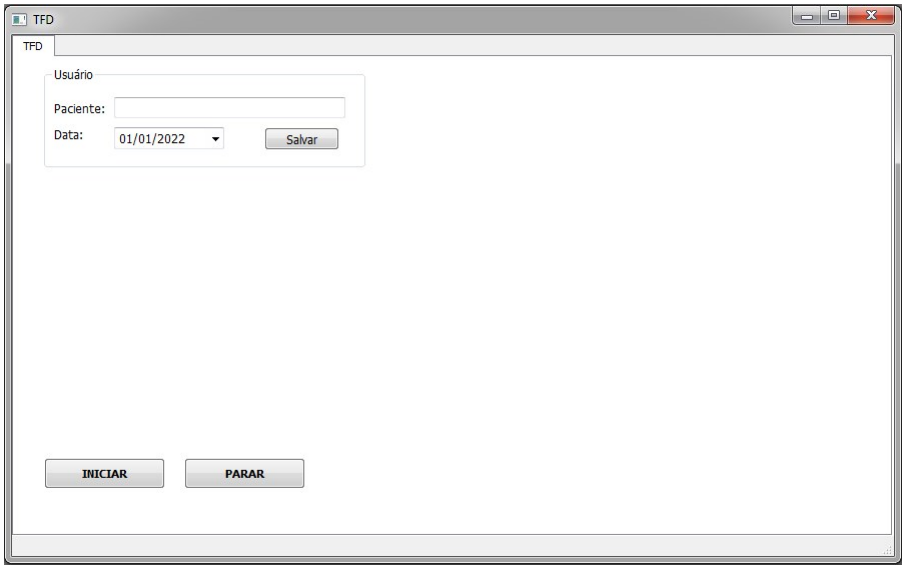
4 RESULTADOS

Três arquivos foram desenvolvidos, os quais contém as partes que integram a interface gráfica, sendo eles os arquivos *controle.py*, *mplwidget.py* e *interface.ui*. O código fonte e demais arquivos são disponibilizados no *github* (ARAUJO, 2023).

4.1 Desenvolvimento da interface gráfica

No arquivo *interface.ui* está a interface gráfica desenvolvida no *software QtDesigner*. Como detalhado anteriormente, para a interface em questão, desenvolvida para este projeto, necessita-se de um controle de usuário, um *widget* para disposição da imagem da lesão, outro para disposição do gráfico de decaimento da fluorescência na região da lesão, e botões para controle do tratamento e do equipamento. No *QtDesigner*, foi desenvolvido o *layout* da interface gráfica, conforme pode ser visto na figura 6.

Figura 6 – Interface gráfica desenvolvida no *software QtDesigner*



Fonte: Própria.

Na região de controle de usuário foi inserido um *QGroupBox*, que agrupa todos os *widgets* da seção, como o *QLineEdit*, que recebe o nome do paciente, o *QDateEdit* que recebe a informação do dia em que determinado tratamento foi realizado, e o *QPushButton* que realiza a ação de salvar os dados inseridos. Nos espaços centrais estão posicionados os *widgets QLabel*, à esquerda, e *QWidget*, à direita, destinados à imagem da lesão adquirida pela câmera, e o gráfico de decaimento exponencial da fluorescência da lesão.

A biblioteca *PyQt5*, através do método *uic.loadUI()*, permite que essa interface seja carregada num código em *Python*, para que seja possível conferir ações e funcionalidades aos *widgets* nela inseridos. A classe principal da interface, que herda de *QMainWindow*, deve

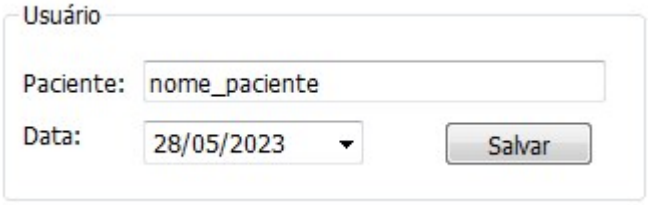
ser implementada, e em sua inicialização ocorre o carregamento do arquivo da interface, a conexão dos botões às suas respectivas funções, a instânciação de todas as *threads* que serão executadas pela interface, e a definição das variáveis que serão utilizadas.

A funcionalidade desempenhada por cada elemento da interface é definida em funções individuais. Assim, quando ocorre o evento do usuário interagir com esses elementos, eles são diretamente conectados à essas funções, de acordo com as conexões definidas na inicialização da classe da interface principal, que executam determinadas tarefas. Com a utilização da linguagem *Python*, essas funções são simples de implementar, e permitem o desenvolvimento de tarefas complexas com pouco tempo depreendido para elas.

4.2 Controle de Usuário

Para iniciar o tratamento, o usuário da interface deve primeiro inserir os dados do paciente, para que tenha definido o local onde as informações de cada paciente serão armazenadas. A área para controle de usuário inserida na interface é mostrada na figura 7.

Figura 7 – Área para Controle de Usuário na interface gráfica.



Fonte: Própria.

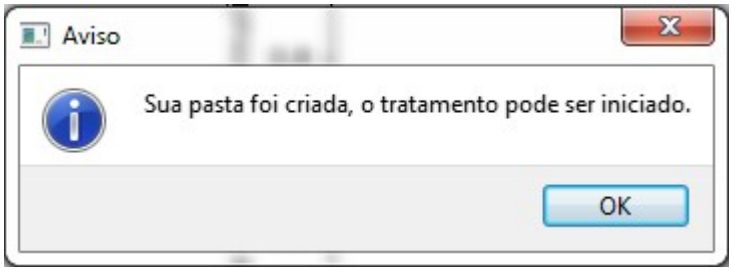
A partir dos dados informados pelo usuário, a biblioteca *os* é utilizada para criar uma pasta com nome do paciente. Dentro dessa pasta, outra pasta é criada com a data da realização do tratamento. Todos os arquivos gerados, referentes ao mesmo paciente, são inseridos nessas pastas para acompanhamento da evolução do tratamento.

Depois de informar os dados, o usuário deve clicar no botão "salvar"para que ocorra a criação das pastas. Porém, nesse momento, antes da criação da pasta, é feita uma verificação para conferir se já existe uma pasta referente a esse paciente, ou referente a data definida. Caso seja inserido um nome de um paciente que já tenha registro prévio, é criada somente uma pasta nova dentro desse diretório com a data do tratamento, como mostra a figura 8. Caso seja um registro novo, toda a estrutura de pastas é criada, e uma notificação em *pop-up* é mostrada na tela, como mostra a figura 9, para indicar ao paciente que o tratamento já pode ser inicializado.

Figura 8 – Pasta criada dentro do diretório com o nome do paciente.

Fonte: Própria.

Figura 9 – *Pop-up* com aviso de confirmação



Fonte: Própria.

4.3 Controle de Câmera

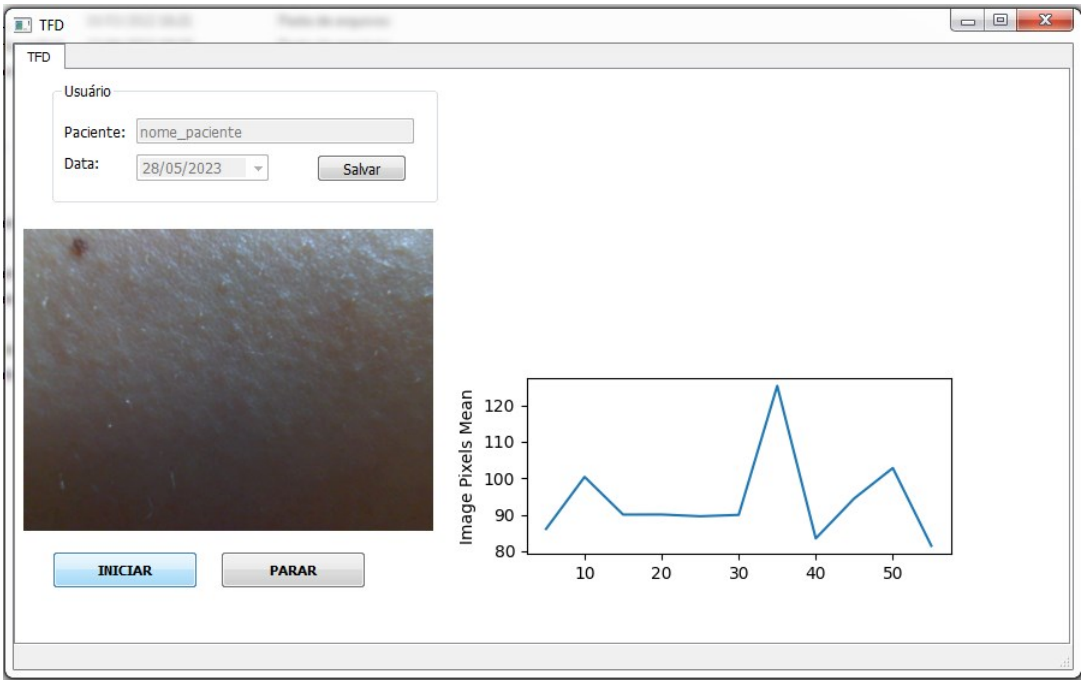
Para incorporar a imagem da *webcam* na interface, o código que a controla usando a biblioteca *OpenCV* está em uma *thread* diferente da principal. Nessa *thread* estão definidos os sinais que serão emitidos dela, com os dados que serão passados para fora da *thread*, além da função principal que mantém a *webcam* ligada.

Assim que o botão "INICIAR"é pressionado, a função que inicializa as *threads* é chamada, e a *thread* que controla a câmera é inicializada, de forma que a imagem captura é mostrada na tela no *widget* determinado. O funcionamento da câmera juntamente com a interface pode ser visto na figura 10.

Na *thread* da câmera, a cada vez que o *frame* é atualizado um sinal é emitido, utilizando o *pyqtSignal*, com a imagem correspondente àquele *frame*. Esse sinal é captado numa função na *thread* principal da interface, e essa imagem é tratada para que possa ser disponibilizada adequadamente no *widget*. Nessa função, a imagem é convertida do espaço de cor BGR para o RGB, já que a biblioteca *OpenCV* utiliza o BGR como padrão, e para o formato do *widget QPixmap* onde ela é inserida na interface. Ao final, essa imagem tratada é atualizada no *widget* e permite uma visualização fluida do que é capturado pela câmera.

Assim, a *thread* da câmera se mantém funcional em todo o tratamento, atualizando o *widget* a cada atualização de *frame*, permitindo a visualização da lesão em tempo real durante o tratamento, sem que a interface congele e não seja responsiva.

Figura 10 – Interface gráfica sendo executada juntamente com *webcam* em tempo real.



Fonte: Própria.

4.4 Controle de Tempo de Armazenamento das Imagens.

O controle do tempo do tratamento também é feito em uma *thread* separada da principal, na qual se estabelece um intervalo de tempo para captura de imagens e um tempo limite para o tratamento. Essa *thread* é inicializada juntamente com a *thread* da câmera e nela estão definidos o sinal que será emitido e a função principal. A *thread* inicia a contagem de tempo do tratamento, e a cada intervalo de tempo estabelecido, salva a informação do tempo que se decorreu e um sinal é emitido. No *slot* onde esse sinal é recebido existe uma função que salva no sistema de arquivos a imagem contida naquele instante na interface.

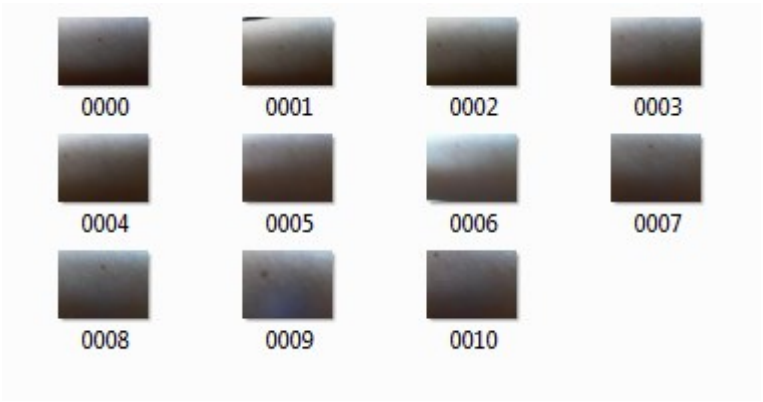
A captura das imagens é feita a cada intervalo de tempo definido, de forma que somente algumas imagens do tratamento são salvas para futuras análises. A biblioteca *time* permite a contagem de tempo decorrido entre dois pontos, sendo eles o começo do tratamento, e cada nova aquisição de imagem. Com esse intuito, a função *sleep()* é utilizada para garantir o intervalo de tempo entre a aquisição das imagens. Quando essa função é utilizada, ela congela a execução do programa pelo tempo definido, em segundos. Por esse motivo, é necessário que essa contagem de tempo seja feita numa *thread* separada, do contrário, a execução de todo o programa seria congelada e a interface não seria responsiva, e dentro da *thread*, somente aquela unidade de processamento específico é interrompida.

Passado o intervalo de tempo definido, um arquivo csv é criado, a informação do tempo que se passou é salva nele, e um sinal é emitido para notificar que uma imagem deve ser salva nesse instante. Nesse sinal específico, nenhuma informação precisa ser passada da

thread, sua única utilidade é chamar a função, na *thread* principal, que salva a imagem do *frame* naquele instante.

Na função que salva a imagem, utiliza-se a biblioteca *OpenCV* para salvar os arquivos das imagem nos *frames* requeridos, e o nome de cada arquivo é definido por numerais em ordem crescente, como mostrado na figura 11.

Figura 11 – Imagens adquiridas e salvas durante a execução do tratamento.

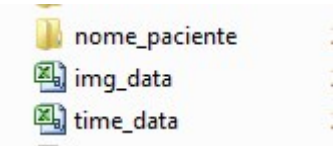


Fonte: Própria.

4.5 Cálculo do Decaimento da Fluorescência e Impressão em Tempo Real

Para o cálculo do decaimento da fluorescência, é feito o cálculo da média dos *pixels* da imagem da lesão, utilizando a função *mean()*, da biblioteca *Numpy* do Python. Esse cálculo é feito na mesma função que salva a imagem no sistema de arquivos, de forma que, logo após ser salva, a mesma imagem já é processada e o resultado obtido do cálculo da média pode ser salvo num novo arquivo csv. Ao final, são obtidos dois arquivos csv, contendo a informação do tempo em que as imagens foram adquiridas e a média dos *pixels* das mesmas, além da pasta com os arquivos armazenados, mostrados na figura 12.

Figura 12 – Arquivos gerados durante o tratamento.

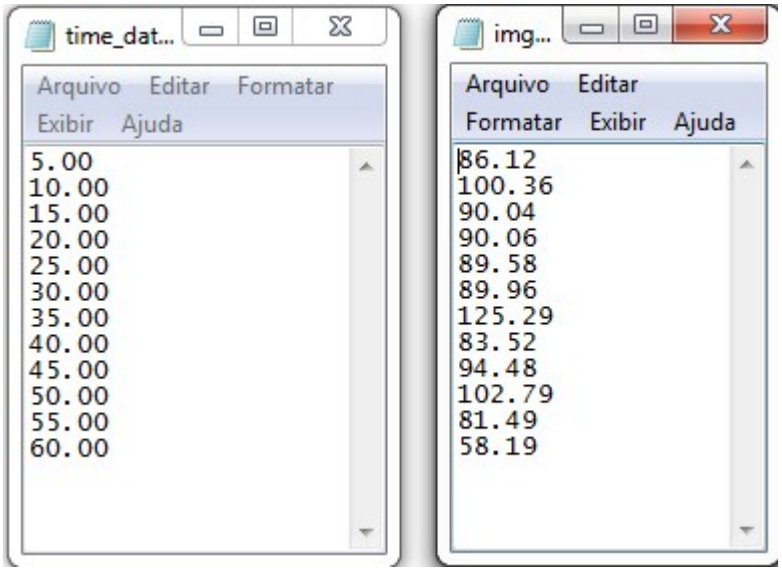


Fonte: Própria.

De posse dos dados da média dos *pixels* da imagem e do tempo em que cada dado foi obtido, nos arquivos csv mostrados na figura 13, é possível gerar o gráfico com essas informações. Para isso, uma outra função é utilizada, responsável por fazer a leitura dos

arquivos csv contendo os dados a serem plotados, por convertê-los em listas e por chamar o método que plota esses dados no gráfico da interface.

Figura 13 – Arquivos csv gerados durante o tratamento.

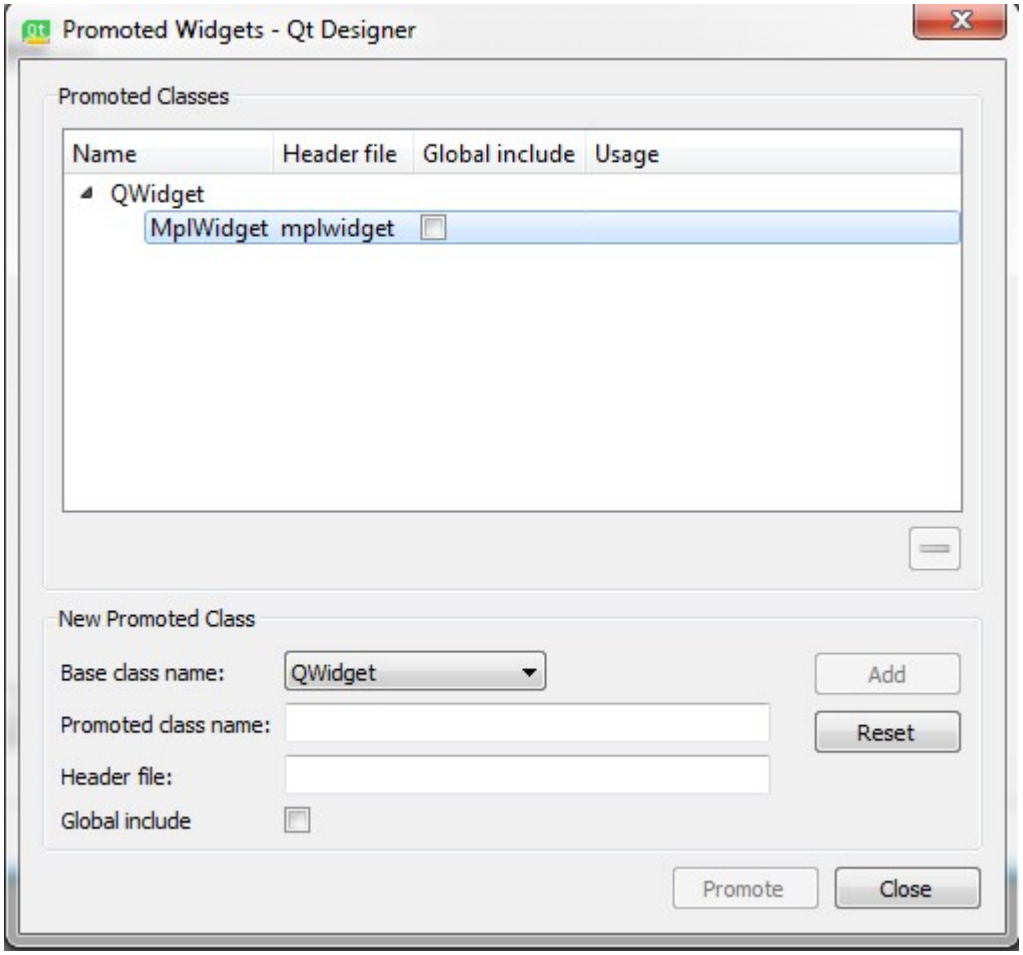


Fonte: Própria.

Como discutido anteriormente, o *widget* onde o gráfico é inserido deve ser um *widget* promovido a um *widget* do *matplotlib*, como é mostrado na figura 14. Ao integrar seu desenvolvimento no código em *Python*, ele possui uma classe específica onde seu layout e seus métodos são definidos. Assim, sua classe herda de *QWidget*, e tem seu layout definido logo que a classe é inicializada. O método responsável por adicionar o gráfico na figura também é definido nessa classe, assim, quando ela é instanciada na classe principal da interface gráfica isso permite que os métodos sejam chamados de dentro da classe principal.

Na função que faz a leitura dos arquivos csv, o método que faz a impressão do gráfico é chamado, passando os dados que foram lidos. A cada vez que uma imagem é salva, sua média e calculada, e a impressão é feita na interface, assim, a cada nova atualização do tratamento, um novo dado é inserido no gráfico que é visto na interface, e é possível acompanhar em tempo real os dados que são obtidos durante o tratamento. A impressão dos dados juntamente com a imagem em tempo real da câmera pode ser visto na figura 10.

Figura 14 – *Widget* promovido a um *widget* do *matplotlib*.



Fonte: Própria.

5 CONCLUSÃO

Do exposto acima, e através do aprofundamento e do estudo sobre as funcionalidades do *software Qt-Designer* e da linguagem *Python*, é possível depreender os direcionamentos do desenvolvimento de uma nova interface gráfica para aprimoramento do monitoramento da Terapia Fotodinâmica. O software da Qt é utilizado em diversas aplicações em tempos atuais, se mostrando uma alternativa em alta para o desenvolvimento de interfaces e dispositivos embarcados. A linguagem *Python* é também uma linguagem de alto nível que vem cada vez mais sendo utilizada para o desenvolvimento de *softwares*, conferindo menor tempo de desenvolvimento e alta capacidade de processamento.

O desenvolvimento da interface gráfica utilizando o *software Qt-Designer* confere rapidez e facilidade na estruturação do *layout* da mesma, visto que, o *software* é intuitivo e simples de utilizar. Assim, a escolha do ambiente de desenvolvimento permitiu que a interface fosse desenvolvida de forma direta, de tal maneira que, quando incorporada ao código de desenvolvimento em *Python*, bastou conectar as partes da interface às funcionalidades em *Python*, sem a necessidade de ajustar os elementos por linhas de código, o que tornaria o processo muito mais trabalhoso, demorado e pouco produtivo.

Do uso da linguagem *Python*, juntamente com a biblioteca *PyQt5* para integrar a interface gráfica, foi possível conferir funcionalidades à interface de forma simplificada. O uso de bibliotecas em *Python* para agregar diferentes etapas no processamento de dados se fez bastante útil, dada a facilidade de implementação e robustez de resultados. O controle da *webcam*, através da biblioteca *OpenCV*, e o uso das funções das bibliotecas *Numpy* e *matplotlib* para obtenção e exposição de dados contribui para a facilidade de desenvolvimento e utilização da interface, visto que, são bibliotecas amplamente utilizadas e de grande conhecimento difundido.

Realizar o controle de usuário da interface, bem como o armazenamento de dados do tratamento, contribui grandemente para o controle do tratamento e acompanhamento dos resultados. O uso da linguagem *Python* para tratamento e manipulação dos dados permitiu maior facilidade e simplicidade para lidar com os mesmos, visto que, a linguagem possui bibliotecas específicas para tratar dados armazenados no sistema de arquivos.

Dentre as dificuldades enfrentadas no desenvolvimento da interface, pode-se destacar a adaptação do *layout* da mesma para disponibilizar os dados calculados. A linguagem *Python* utilizada permite grande capacidade de processamento, e suas funcionalidades são facilmente implementadas no desenvolvimento do *software*. Contudo, a disponibilização de dados como a imagem da *webcam* e o gráfico de decaimento da fluorescência são tarefas complexas, pois os dados precisam ser ajustados no *layout* da interface para correta visualização. Diante disso, a imagem adquirida da *webcam* precisou ser ajustada para o *layout* do *widget* utilizado, e o *layout* do gráfico disponibilizado através do *matplotlib* também recebeu ajustes específicos de *layout*.

A escolha do ambiente de desenvolvimento e da linguagem se mostra eficiente para promover maior usabilidade do sistema, além de conferir uma integração maior das partes, maior facilidade de obtenção de dados e resultados, e dá abertura para integração com diversas outras possibilidades de processamento. Essas melhorias impactam a aplicação e o monitoramento da terapia fotodinâmica, de forma que, o uso de ferramentas da linguagem *Python* permite a interpretação e análise dados do tratamento em tempo real. Assim, a manipulação das informações obtidas durante o tratamento fotodinâmico viabiliza a criação de um protocolo customizado adaptado às necessidades de cada paciente, que possuem conjuntos de parâmetros fisiológicos e ópticos específicos.

Como possíveis direcionamentos futuros, faz-se interessante integrar o trabalho desenvolvido por Igor Cordeiro Santa Bárbara na interface, para que seja possível controlar via *software* o módulo laser, e exibir dados de controle de corrente, por exemplo, diretamente na interface. Além disso, o embarcamento da interface no equipamento se faz interessante para extensão dos resultados obtidos, de forma que seria possível efetivamente utilizar a interface em ambiente clínico e implementar outras funcionalidades como controle do *driver* do laser, estimação dos parâmetros e o uso de inteligência artificial para estimar resultados dos tratamentos a partir dos parâmetros calculados

5.1 Participações em eventos científicos

- Participação no VIII Simpósio de Iniciação Científica da Engenharia Elétrica, 2020 (ARAÚJO et al., 2020a).
- Participação no Simpósio Internacional de Iniciação Científica e Tecnológica da USP, 2020, com indicação para a fase internacional (ARAÚJO et al., 2020b).
- Participação no Simpósio Internacional de Iniciação Científica e Tecnológica da USP, 2022, na Semana Integrada do Instituto de Física de São Carlos (ARAÚJO et al., 2022).

REFERÊNCIAS

ACKROYD, R. et al. The history of photodetection and photodynamic therapy¶. *Photochemistry and photobiology*, Wiley Online Library, v. 74, n. 5, p. 656–669, 2001.

AKOPOV, A. L. et al. Endobronchial photodynamic therapy under fluorescence control: Photodynamic theranostics. *Photodiagnosis and Photodynamic Therapy*, v. 19, p. 73–77, 2017. ISSN 1572-1000. Disponível em: https://www.sciencedirect.com/science/article/pii/S1572100016302836.

ARAUJO, F. T. *PDT Monitoring Graphical Interface*. 2023. Disponível em: https://github.com/fabianaaraujo/pdt_monitoring.

ARAÚJO, F. T. d. et al. Desenvolvimento de uma interface gráfica para monitoramento online de terapia fotodinâmica. In: *Simpósio de Iniciação Científica da Engenharia Elétrica - SICEEL*. Universidade de São Paulo - USP, Escola de Engenharia de São Carlos - EESC, 2020. Disponível em: http://soac.eesc.usp.br/index.php/SICEEL/viiisiceel/paper/view/2605/1594.

ARAÚJO, F. T. d. et al. Interface gráfica e controle do laser para equipamento que realiza terapia fotodinâmica. In: . Universidade de São Paulo - USP, 2020. Disponível em: https://uspdigital.usp.br/siicusp/siicPublicacao.jsp?codmnu=7210f.

ARAÚJO, F. T. d. et al. Avanços na aplicação e monitoramento da terapia fotodinâmica. In: . Universidade de São Paulo - USP, 2022. Disponível em: https://repositorio.usp.br/directbitstream/3d4a912c-938b-4677-80a1-8366335f351a/3046554.pdf.

BÁRBARA, I. C. bathesis, *Controle de Driver de Laser em Sistema de Aplicação e Monitoramento da Terapia Fotodinâmica de Câncer de Pele do Tipo Não-Melanoma*. São Carlos: [s.n.], 2020. 53 p.

BLANCHETTE, J.; SUMMERFIELD, M. *C++ GUI programming with Qt 4*. [S.l.]: Prentice Hall Professional, 2006.

BRAATHEN, L. R. et al. Guidelines on the use of photodynamic therapy for nonmelanoma skin cancer: An international consensus. *Journal of the American Academy of Dermatology*, v. 56, n. 1, p. 125–143, 2007. ISSN 0190-9622. Disponível em: https://www.sciencedirect.com/science/article/pii/S019096220601718X.

CALZAVARA-PINTON, P.; VENTURINI, M.; SALA, R. Photodynamic therapy: update 2006 part 1: Photochemistry and photobiology. *Journal of the European Academy of Dermatology and Venereology*, v. 21, n. 3, p. 293–302, 2007. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-3083.2006.01902.x.

CASTANO, A. P.; DEMIDOVA, T. N.; HAMBLIN, M. R. Mechanisms in photodynamic therapy: part one-photosensitizers, photochemistry and cellular localization. *Photodiagnosis and photodynamic therapy*, p. 279–293, dec 2004. Disponível em: https://doi.org/10.1016/S1572-1000(05)00007-4.

CELLI, J. P. et al. Imaging and photodynamic therapy: mechanisms, monitoring, and optimization. *Chem Rev*, v. 110, n. 5, p. 2795–2838, May 2010.

DOLMANS, D.; FUKUMURA, D.; JAIN, R. Photodynamic therapy for cancer. *Nature Reviews Cancer*, v. 3, p. 380–387, May 2003.

DOUGHERTY, T. J. et al. Photodynamic Therapy. *JNCI: Journal of the National Cancer Institute*, v. 90, n. 12, p. 889–905, 06 1998. ISSN 0027-8874. Disponível em: https://doi.org/10.1093/jnci/90.12.889.

GARCIA, M. R. et al. Assembly of a pdt device with a near-infrared fluorescence monitoring of ppix during treatment. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Optical Methods for Tumor Treatment and Detection: Mechanisms and Techniques in Photodynamic Therapy XXVIII*. [S.l.], 2019. v. 10860, p. 108600T.

GARCIA, M. R. et al. Development of a system to treat and online monitor photodynamic therapy of skin cancer using ppix near-infrared fluorescence. *Photodiagnosis and Photodynamic Therapy*, Elsevier, p. 101680, 2020.

KENNEDY, J. C.; POTTIER, R. H. New trends in photobiology: Endogenous protoporphyrin ix, a clinically useful photosensitizer for photodynamic therapy. *Journal of Photochemistry and Photobiology B: Biology*, v. 14, n. 4, p. 275–292, 1992. ISSN 1011-1344. Disponível em: https://www.sciencedirect.com/science/article/pii/1011134492851087.

Ministério da Saúde. *Portaria SECTICS/MS no 46*. 2023. Disponível em: https://www.gov.br/conitec/pt-br/midias/relatorios/portaria/2023/portaria-sectics-ms-no-46.

MORTON, C. A. et al. Guidelines for topical photodynamic therapy: report of a workshop of the british photodermatology group. *British Journal of Dermatology*, Blackwell Science Ltd Oxford, UK, v. 146, n. 4, p. 552–567, 2002.

POGUE, B. W. et al. Revisiting photodynamic therapy dosimetry: reductionist & surrogate approaches to facilitate clinical success. *Physics in Medicine & Biology*, IOP Publishing, v. 61, n. 7, p. R57, mar 2016. Disponível em: https://dx.doi.org/10.1088/0031-9155/61/7/R57.

Wikipedia contributors. *Qt (software) — Wikipedia, The Free Encyclopedia*. 2023. [Online; accessed 19-December-2023]. Disponível em: https://en.wikipedia.org/w/index.php?title=Qt_(software)&oldid=1190284780.

WILLMAN, J.; WILLMAN, J. Overview of pyqt5. *Modern PyQt: Create GUI Applications for Project Management, Computer Vision, and Data Analysis*, Springer, p. 1–42, 2021.

YOUNUS, L. A. et al. Photodynamic therapy in cancer treatment: properties and applications in nanoparticles. *Brazilian Journal of Biology*, Instituto Internacional de Ecologia, v. 84, p. e268892, 2023. ISSN 1519-6984. Disponível em: https://doi.org/10.1590/1519-6984.268892.

APÊNDICE

Código desenvolvido em linguagem *Python* para integração da interface gráfica.

```
# imported modules and libraries
import sys, os
from PyQt5 import QtWidgets, QtCore, uic, QtGui
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import pyqtSignal, pyqtSlot, Qt, QThread
from PyQt5.QtWidgets import QMessageBox, QApplication, QWidget,
    QVBoxLayout
from datetime import datetime
from threading import Thread
import cv2
import numpy as np
from matplotlib import pyplot as plt
from mplwidget import MplWidget
import pandas as pd
import time
import csv

# time counter thread
class TimeCounterThread(QtCore.QThread):
    writerow_signal = pyqtSignal(float)

    def __init__(self):
        super().__init__()
        self.run_flag = True

    def run(self):
        y= 0
        # creates a csv file to store time data
        while self.run_flag:
            start = time.time()
            time.sleep(5)
            end=time.time()
            x = end-start
            y = x + y
            with open('time_data.csv','a', newline='') as results:
```

```
                dados = csv.writer(results)
                dados.writerow(["{:.2f}".format(y)])
            self.writerow_signal.emit(y)
            if y>=60:
                self.run_flag = False
    def stop(self):
        self.run_flag = False
        self.wait()

# camera thread
class ThreadClass(QtCore.QThread):
    change_pixmap_signal = pyqtSignal(np.ndarray)

    def __init__(self):
        super().__init__()
        self._run_flag = True

    def run(self):
        # capture from web cam
        cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
        while self._run_flag:
            ret, cv_img = cap.read()
            if ret:
                self.change_pixmap_signal.emit(cv_img)
        # shut down capture system
        cap.release()

    def stop(self):
        # set run flag to False and waits for thread to finish
        self._run_flag = False
        self.wait()

# main thread that handles the UI thread
class App(QtWidgets.QMainWindow):

    def __init__(self, parent=None):
        super().__init__(parent)
        uic.loadUi("Interface.ui", self)
        #connect actions to methods
```

```
        self.salvar.clicked.connect(self.criar_pasta)
        self.btnIniciar.clicked.connect(self.plotar_grafico)

        self.threadclass = ThreadClass()
        # connect its signal to the update_image slot
        self.threadclass.change_pixmap_signal.connect(self.
            update_image)

        self.timecounterthread = TimeCounterThread()
        self.timecounterthread.writerow_signal.connect(self.
            save_image)

        self.path2 = None
        self.paciente = None
        self.data = None
        self.label = 0
        self.cv_img = None

        self.MplWidget = MplWidget()
        self.mpl_container.layout().addWidget(self.MplWidget)

    # aquire user data to create folders
    def criar_pasta(self):
        self.paciente = self.pacienteinput.text()
        self.data = datetime.strftime(self.dateinput.date().
            toPyDate(),'%y-%m-%d')
        self.path = os.getcwd()
        default = "21-01-01"

        if not self.paciente:
            QMessageBox.warning(self,'Erro', 'Insira␣o␣nome␣do␣
                usu rio␣para␣prosseguir.')
        elif self.data == default:
            QMessageBox.warning(self,'Erro', 'Insira␣a␣data␣de␣
                hoje␣para␣prosseguir.')
        else:
            self.pacienteinput.setEnabled(False)
            self.dateinput.setEnabled(False)
            if not os.path.exists(self.path +'/'+ self.paciente):
```

```
            os.mkdir(self.path +'/'+ self.paciente)
            os.mkdir(self.path +'/'+ self.paciente + '/' + self
                .data)
            QMessageBox.information(self,'Aviso','Sua␣pasta␣foi
                ␣criada,␣o␣tratamento␣pode␣ser␣iniciado.')
        else:
            os.mkdir(self.path +'/'+ self.paciente + '/' + self
                .data)
            QMessageBox.information(self,'Aviso','Sua␣pasta␣foi
                ␣criada,␣o␣tratamento␣pode␣ser␣iniciado.')
    self.path2 = self.path +'/'+ self.paciente + '/' + self.
        data
    return self.path2

# starts the camera and the time counter threads
def plotar_grafico(self):
    self.threadclass.start()
    self.timecounterthread.start()

# deal with the closing UI event and stop threads
def closeEvent(self, event):
    self.threadclass.stop()
    self.timecounterthread.stop()
    event.accept()

# receive signals and show the frame images from the webcam on
    the screen
@pyqtSlot(np.ndarray)
def update_image(self, cv_img):
    """Updates the image_label with a new opencv image"""
    qt_img = self.convert_cv_qt(cv_img)
    self.image.setPixmap(qt_img)

# adjust image format
def convert_cv_qt(self, cv_img):
    self.cv_img = cv_img
    """Convert from an opencv image to QPixmap"""
    rgb_image = cv2.cvtColor(cv_img, cv2.COLOR_BGR2RGB)
    h, w, ch = rgb_image.shape
```

```
        bytes_per_line = ch * w
        convert_to_Qt_format = QtGui.QImage(rgb_image.data, w, h,
            bytes_per_line, QtGui.QImage.Format_RGB888)
        return QPixmap.fromImage(convert_to_Qt_format)

    @pyqtSlot(float)
    def save_image(self):
        cv2.imwrite(self.path2 +'/'+ f'{self.label:04d}' +'.jpg',
            self.cv_img)
        self.label = self.label + 1
        # Calculates image pixel mean
        with open('img_data.csv','a', newline='') as data:
            dados_imagem = csv.writer(data)
            img_mean = np.mean(self.cv_img)
            dados_imagem.writerow(["{:.2f}".format(img_mean)])
        self.plot_graph()

    def plot_graph(self):
        dados_x = pd.read_csv('time_data.csv', header = None,
            names = ["Time"])
        dados_y = pd.read_csv('img_data.csv', header = None, names
            = ["Data"])
        dados_x = dados_x["Time"].tolist()
        dados_y = dados_y["Data"].tolist()
        self.MplWidget.update_data(dados_x, dados_y)


# main code
if __name__ =='__main__':              #Running the file directly
    qt = QApplication(sys.argv)        #Instance of the QApplication
        Class
    window = App()                     #Instance of the App Class
    window.show()                      #Calling the method show()
        through the instance
    sys.exit(qt.exec_())               #Calling the method exec_()
```

Código desenvolvido em linguagem *Python* para o *plot* em tempo real dos dados de decaimento de fluorescência.

```
from PyQt5.QtWidgets import *
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
    as FigureCanvas
from matplotlib import pyplot as plt
from matplotlib.figure import Figure

class MplWidget(QWidget):

    def __init__(self, parent = None):

        QWidget.__init__(self, parent)
        self.figure = Figure()
        self.grafico = FigureCanvas(self.figure)
        layout = QVBoxLayout()
        layout.addWidget(self.grafico)
        self.setLayout(layout)
        self.grafico.setSizePolicy(QSizePolicy.Expanding,
            QSizePolicy.Expanding)

    def update_data(self, x, y):
        self.figure.clear()
        ax = self.figure.add_subplot(111)
        ax.set_xlabel('Time [s]')
        ax.set_ylabel('Image Pixels Mean')
        ax.plot(x, y)
        self.grafico.draw()
```