

AI Virtual Try – On Using Artificial Intelligence and Augmented Reality: A Light weight CPU – Oriented Web-Based System

Vijilan M, AbinoJoes A, Pabish P, Yogeshwaran R

Department of n Artificial Intelligence and Data Science,

Loyola Institute of Technology & Science Mrs. Mary Suguna J, Assistant Professor

ABSTRACT Virtual Try-On (VTON) technology enables a user to preview a garment on a personal photograph without physically wearing it. This work presents a lightweight, CPU-oriented web-based VTON system designed for academic laboratories and ordinary computers where GPU acceleration is unavailable. Instead of performing end-to-end generative synthesis, the proposed pipeline combines lightweight pretrained inference models with classical geometric image processing. MediaPipe Pose estimates body landmarks; SegFormer provides semantic human parsing; U2NETP extracts the garment foreground; Thin Plate Spline transformation aligns the garment with the target pose; and alpha blending produces the final composite. React/Vite and FastAPI provide the application framework, while lazy loading, model reuse, image resizing, and resolution fallback support resource-aware execution. The design emphasizes modularity, local processing, privacy awareness, and normal operation without project-specific model training.

INDEX TERMS Virtual try-

on, computer vision, MediaPipe Pose, SegFormer, U2NETP, Thin Plate Spline, alpha blending, CPU inference, FastAPI, React, image processing.

I. INTRODUCTION AND MOTIVATION

Virtual Try-On (VTON) is an image-based computer vision application that allows a user to visualize a selected garment on a target-person photograph. The basic interaction is simple: the user supplies a photograph of a person and an image of a garment, and the system produces a composite image in which the garment appears to be worn by the person. This capability is increasingly relevant to online fashion retail, digital catalogs, interactive shopping applications, fashion visualization, and academic research in image processing and artificial intelligence.

The motivation for a virtual try-on system comes from a practical limitation of conventional online shopping. A garment product image normally represents the clothing on a model or against a plain background, while a customer needs to understand how that garment may appear with a different body shape, pose, and photographic context. Physical trial is not always possible in

online shopping. A virtual representation can therefore provide an additional visual reference before a

purchased decision is made. Although an image-based try-on result is not equivalent to a physical fitting, it can improve the interactive character of digital apparel applications.

Modern VTON research has produced increasingly sophisticated methods. Many recent systems use deep generative models, adversarial networks, transformer architectures, or diffusion-based synthesis to generate realistic outputs. These methods can preserve detailed garment appearance and handle difficult image synthesis tasks, but their computational requirements can be significant. Large models, GPU memory, high-resolution processing, and training datasets can make direct deployment difficult on ordinary computers. This creates an important engineering question: can a useful VTON workflow be implemented with lower resource requirements while retaining a clear and understandable processing pipeline?

The proposed project addresses this question through a lightweight CPU-oriented design. Instead of training an end-to-end image generator, the system combines pretrained perception components with classical geometric image processing. MediaPipe Pose is used to estimate body

landmarks. SegFormer is used for semantic human parsing. U2NETP is used to estimate a salient foreground mask for the garment. Thin Plate Spline transformation is used for non-rigid geometric deformation, while alpha blending and OpenCV-based processing are used to form the final image.

This design intentionally separates the VTON task into recognizable stages. A pose estimator does not have to solve garment segmentation, and the garment extractor does not have to synthesize a complete person. Each component has a limited responsibility. This separation is useful for academic development because intermediate outputs can be inspected and failures can be localized. If a garment is misplaced, the pose and geometric stages can be examined. If the source background appears in the output, the garment mask can be examined. If the garment boundary is too hard or transparent, the blending stage can be adjusted.

The project also considers deployment constraints. The frontend is implemented using React and Vite, while the backend uses FastAPI. The backend pipeline controls image validation, resizing, model loading, inference, geometric transformation, blending, and result delivery. Memory-aware techniques include lazy model loading, model reuse, controlled image dimensions, and a resolution fallback strategy from 640 pixels to 384 pixels and then 256 pixels. These techniques are intended to make the application more practical on CPU-oriented systems.

The objective is therefore not to claim that a classical compositing pipeline produces higher visual fidelity than state-of-the-art generative VTON. Instead, the project explores a different point in the design space: lower deployment complexity, local processing, modularity, and accessibility. The system is intended as an academic prototype that demonstrates how multiple computer vision components can be integrated into a complete browser-based VTON workflow.

The remainder of this paper presents the project objectives, technical foundations, literature context, limitations of existing approaches, proposed architecture, processing algorithms, software implementation, evaluation methodology, privacy and security considerations, future enhancements, and conclusion. The paper distinguishes intended system behavior from measured experimental results because the available project material does

not provide a controlled benchmark with numerical image-quality scores. A VTON application can be viewed as a sequence of constrained image transformations rather than a single image-generation operation. The target photograph supplies the scene that must largely be preserved, while the garment photograph supplies the visual object that must be inserted. This distinction creates two simultaneous requirements: the system must preserve identity and scene information in the person image, and it must preserve garment appearance while changing its spatial configuration. A successful pipeline therefore needs both recognition and controlled transformation.

From a software-engineering perspective, the project is also an integration problem. The output of a pose model must be expressed in coordinates usable by the geometric stage. The semantic parser must produce a label that can be converted into masks. The garment extractor must produce a mask that has exactly the same spatial dimensions as the garment image. TPS must transform both representations consistently. Finally, the alpha-compositing stage must combine them without destroying protected regions. The overall result depends on the compatibility of these interfaces as much as on the quality of an individual model.

The CPU-oriented objective affects decisions throughout the pipeline. Image resolution, model lifecycle, intermediate buffers, and file handling are not merely implementation details; they influence whether the application can complete a request on ordinary hardware. The proposed architecture therefore treats resource management as part of the system design. A lower-resolution fallback is preferable to an application that fails completely when the preferred resolution exceeds available memory.

The academic value of this approach is its transparency. A demonstration can show the original person image, garment image, pose landmarks, semantic mask, garment mask, warped garment, and final composite. Each intermediate result illustrates a concept from computer vision or image processing. This makes the project suitable for explaining how AI perception and classical geometry can cooperate in a practical application rather than presenting the VTON result as an unexplained black box.

The practical significance of the system is strongest when accessibility is treated as a design constraint from the beginning. A model that produces excellent images but cannot run on the available computer does not satisfy the project's intended deployment context. The proposed workflow instead asks which operations genuinely require learned inference and which can be implemented with deterministic image processing. This separation reduces unnecessary computation and makes the overall system easier to explain. It also creates a useful baseline for later experiments in which a single component can be upgraded and the effect of that upgrade can be observed independently.

Another important motivation is reproducibility. A complete local pipeline can be installed and demonstrated without depending on a continuously available external

inference service. The same input pair can be processed repeatedly while developers inspect intermediate outputs. Reproducibility is particularly important for academic work because visual demonstrations can otherwise depend on hidden service-side model versions or settings. A locally controlled environment provides clearer experimental boundaries and makes troubleshooting more systematic.

II. PROBLEM STATEMENT, OBJECTIVES AND SCOPE

The central problem is the difficulty of providing a usable virtual try-on experience on resource-constrained hardware. High-fidelity VTON systems often depend on computationally expensive neural synthesis. A student or laboratory environment may instead have a conventional CPU-oriented computer with limited memory and no dedicated accelerator. Requiring a large GPU-centered model would increase setup complexity and reduce accessibility. The proposed system therefore treats computational efficiency as a first-class design requirement rather than an afterthought.

The input to the system consists primarily of a target-person image and a garment image. The person image provides the body geometry, pose, visible regions, and background context. The garment image provides the clothing appearance that should be transferred. Before these images can be combined, the system must determine where the body is located, where important semantic regions occur, which pixels belong to the garment, how the garment should deform, and which parts of the original person image should remain visible.

The first objective is body-aware processing. MediaPipe Pose provides landmarks that can be used as geometric references for the upper body. These landmarks are not a complete 3-D body model, but they provide useful 2-D structural information for locating shoulders, arms, and other body points. The second objective is semantic protection. SegFormer provides human parsing information so that regions such as the face, hair, arms, and background can be treated differently from the clothing region.

The third objective is garment isolation. A clothing photograph may contain a background that should not appear in the final image. U2NETP is used to estimate a salient foreground representation so that the garment can be separated from its original surroundings. Morphological processing can be used to reduce small holes or isolated artifacts in the mask. The garment RGB image and its mask are then treated as a coordinated pair during geometric transformation.

The fourth objective is geometric alignment. A garment cannot normally be transferred successfully using only translation and scaling because clothing changes shape according to body orientation and pose. Thin Plate Spline transformation provides a smooth non-rigid mapping controlled by corresponding source and target points. The garment and its mask are transformed together so that the boundary remains aligned with the garment appearance.

The fifth objective is controlled compositing. Alpha blending combines the transformed garment with the original person image according to a mask. Protected regions can be restored after blending so that the garment does not accidentally cover areas that should remain visible. The final image is then returned to the frontend for display.

The project scope is deliberately limited. Normal operation does not include training a new VTON generator. The system is focused on

image-based upper-body garment transfer and a lightweight web workflow. It does not claim physical cloth simulation, accurate 3-D reconstruction, automatic body measurement, or guaranteed photorealistic synthesis. These capabilities are treated as possible future enhancements rather than existing functions.

A further scope requirement is modularity. The architecture should allow an individual component to be replaced without rewriting the complete application. This is important for future experimentation because pose estimation, segmentation, garment extraction, and refinement models continue to evolve. The current system therefore serves both as an application prototype and as a base platform for later research.

A useful requirement is that every stage should have a measurable responsibility. Input validation is responsible for producing readable images. Pose estimation is responsible for body landmarks. Human parsing is responsible for semantic regions. Garment extraction is responsible for foreground isolation. TPS is responsible for geometric deformation. Alpha blending is responsible for composition. The orchestrator is responsible for sequencing these operations and handling failures. This responsibility-based design reduces ambiguity when the final output is unsatisfactory.

Another requirement is coordinate consistency. The person image, garment image, pose landmarks, semantic mask, garment mask, and warped output must all agree on the working resolution. If an image is resized after landmarks are detected without transforming the coordinates, the garment can be placed incorrectly. Similarly, if the garment mask is generated at one resolution and the RGB garment at another, alpha compositing becomes invalid. The implementation should therefore define a single working coordinate system for each request.

The project also has a usability requirement. A technically correct backend is not sufficient if the user cannot determine which image is the person input and which is the garment input, whether processing has started, or whether the result has completed. The React/Vite layer therefore has a practical role in communicating the state of the pipeline. Progress information is especially useful on CPU-oriented hardware because processing may take longer than a simple browser interaction.

This scope boundary is important for evaluation. The system is intended for an approximate image-based try-on, not for guaranteeing real garment fit. It does not measure fabric elasticity, body circumference, garment dimensions, or physical comfort. A visually plausible image should therefore be interpreted as a visualization result. Any future physical-fit feature would require additional measurements and validation beyond the current image-compositing workflow.

The objectives can also be translated into acceptance criteria for the software. A valid input should pass preprocessing and reach inference. The pose stage should produce usable landmarks for ordinary target images. The garment stage should produce a foreground representation rather than a complete rectangular image.

The geometric stages should produce a transformed garment whose dimensions match the working

image. The blending stages should return an encoded image that the br

owser can display. These criteria make the project easier to test because every objective corresponds to an observable artifact.

The system should also degrade gracefully. A CPU-oriented application cannot assume that every machine has identical memory capacity or processing speed. If the preferred resolution cannot be processed, the fallback mechanism should attempt a smaller resolution rather than terminating immediately. If a request is invalid, the services should reject it before expensive model execution. Graceful degradation therefore applies both to resource limitations and to input errors.

Another requirement concerns maintainability. The project combines front-end, back-end, machine-learning, and image-processing technologies. Without clear module boundaries, a small change in one stage can produce unexpected effects elsewhere. Interfaces between modules should therefore define expected image dimensions, data types, mask conventions, coordinate systems, and error states. This is especially important for TPS because geometric errors can be caused by preprocessing or coordinate conversion rather than by the transformation algorithm itself.

The scope also protects the evaluation from unrealistic expectations. The project should be demonstrated using representative images that match its assumptions and should include difficult examples to show limitations. A successful academic evaluation does not require every possible pose or garment to work perfectly. It requires the system's intended operating conditions, strengths, failure cases, and resource trade-offs to be described accurately.

III. TECHNICAL FOUNDATIONS AND RELATED WORK

VTON research can broadly be understood as a combination of image understanding, geometric alignment, and image synthesis. The person image must be analyzed to determine body structure and visible semantic regions. The garment image must be represented in a form that can be transformed. The transformed garment must then be integrated into the target photograph without producing obvious discontinuities. Different research systems place different amounts of computation into these stages.

VITON established an influential image-based formulation in which a garment is transferred to a person image. Later methods explored improved geometric matching, texture preservation, and high-resolution synthesis. CP-VTON introduced explicit geometric transformation using Thin Plate Splines and demonstrated the value of separating clothing alignment from subsequent image generation. VITON-HD and HR-VITON addressed high-resolution quality and misalignment or occlusion-related problems. These systems provide important context for the present project even though the proposed implementation does not reproduce their full learned synthesis architecture.

The supplied reference journal emphasizes three recurring problems in VTON: inaccurate geometric alignment, loss of garment details, and unnatural garment-body blending. These problems are visible around shoulders, sleeves, collars, waist regions, and patterned garments. The reference also demonstrates a dual-stage philosophy in which geometric alignment and garment-body integration are treated as separate concerns. The present project adopts the general principle of separating responsibilities, but implements the stages with lightweight models and cla

ssical image processing rather than the reference paper's specialized learned refinement networks.

Thin Plate Spline transformation is particularly relevant because clothing deformation is non-

rigid. A rigid transformation assumes that all points move according to the same rotation and scale. Clothing does not behave this way in an image.

Different regions can move differently while still requiring a smooth overall transformation. TPS uses control points and a radial basis function to generate a smooth mapping while penalizing excessive bending. This makes it suitable for approximating changes in garment shape.

Pose estimation and semantic parsing provide complementary information. A pose model supplies sparse structural landmarks. A semantic parser supplies dense region labels. The combination can be used to estimate where a garment should be placed and where it should not be placed. This distinction is important because a correct garment mask alone does not guarantee correct body placement, while pose points alone do not provide sufficient information about occlusion and protected regions.

U2NETP is used in the proposed pipeline for salient garment extraction. Its role is narrower than a complete VTON generator: it estimates a foreground mask so that the garment can be separated from its source background. The mask can then be cleaned and warped along with the garment. This approach avoids training a custom segmentation network for the project while still providing an AI-based image understanding component.

SegFormer provides transformer-based semantic segmentation. In this project it is used to identify relevant human regions rather than to generate the final try-on image. MediaPipe Pose similarly acts as a specialized perception module. The use of pretrained components reduces the need for project-specific training and allows the main engineering effort to focus on orchestration, geometry, memory management, and user interaction.

The literature therefore supports the architectural decision to separate perception, geometric alignment, and synthesis. The research gap for this project is an accessible implementation that combines these ideas into a complete local web application with CPU-oriented resource controls. The proposed system should be evaluated on both visual behavior and computational feasibility rather than only on photorealistic quality.

The literature can also be organized according to where intelligence is placed in the pipeline. Some methods place most of the intelligence in a generative synthesis network. Others make geometric alignment an explicit learned or mathematical stage and then use a synthesis network to refine the result. The present project places recognition intelligence in pretrained specialist models and uses an explicit mathematical transformation for deformation. This difference explains both the computational advantage and the visual limitations of the proposed design.

Thin Plate Spline is especially appropriate when a transformation must be smooth but cannot be represented by a single affine matrix. A garment may need to move differently from its lower region. TPS allows the transformation to respond to multiple control points while maintaining smoothness between them. The method

is therefore a useful bridge between sparse pose landmarks and a dense pixel image. Its effectiveness still depends on selecting meaningful correspondences.

The reference dual-stage VTON paper provides a useful comparison because it treats geometric alignment and garment-body integration as separate concerns. The proposed student system adopts this conceptual separation but uses a much lighter implementation. The comparison should therefore be understood as architectural inspiration rather than a claim of equivalent performance. The reference system is designed for high-fidelity synthesis, while the present project is designed for CPU-oriented accessibility and modular academic experimentation.

The combination of MediaPipe Pose, SegFormer, and U2NETP also illustrates model specialization. A single model is not forced to solve every task. Pose estimation is optimized for landmarks, semantic segmentation for region understanding, and salient-object extraction for foreground separation. This specialization reduces the need to train a project-specific model, while allowing each stage to be inspected and potentially replaced in future versions.

A useful distinction in the literature is between alignment and synthesis. Alignment answers where the garment should be placed and how its visible geometry should change. Synthesis answers how the final photograph should look after that placement, including missing regions, lighting, texture continuity, and occlusion. The present project concentrates on alignment and controlled compositing. This makes the system computationally simpler but also explains why it cannot reproduce all of the visual effects of a learned synthesis network.

The reference research on dual-stage VTON is relevant because it explicitly identifies geometric alignment and garment-body integration as separate technical problems. In a lightweight implementation, this separation can be retained even when the solutions are simpler. TPS acts as the coarse geometric mechanism, while semantic masks and alpha blending act as the integration mechanism. Future learned refinement can then be introduced without changing the conceptual decomposition.

The use of pretrained models also changes the training requirement. The project does not need to collect a new fashion dataset or train an end-to-end VTON generator for normal operation. The model weights provide general visual capabilities, while the project-specific work lies in preprocessing, orchestration, transformation, and compositing. This reduces development time and hardware requirements and makes the prototype more accessible to academic users.

At the same time, pretrained components introduce their own dependency considerations. Model versions, weight files, framework compatibility, and preprocessing assumptions can affect behavior. A reproducible implementation should record the model identifiers, library versions, input normalization rules, and image-size assumptions. Doing so is essential if the project is later compared with an upgraded model or reproduced on another computer.

IV. EXISTING SYSTEM, LIMITATIONS AND RESEARCH GAP

Existing VTON systems commonly use learned neural pipelines. A typical architecture estimates representations of the person and

garment, aligns the clothing to the target body, and generates a final image. High-capacity models can learn complex relationships between body pose, garment appearance, texture, lighting, and occlusion. This provides strong visual potential, particularly when models are trained on large datasets and executed at high resolution.

However, computational requirements can become a barrier. Large models can require significant GPU memory and long inference times on CPU-only hardware. Training adds another level of cost because datasets, preprocessing, accelerator resources, and repeated optimization are required. Even when a trained model is available, environment configuration may involve framework versions, model weights, CUDA dependencies, and hardware-specific optimization.

Another issue is deployment architecture. Cloud-based image APIs can simplify computation because the remote service performs inference, but they require image upload to an external service. For personal photographs, this can create privacy and data-retention concerns. A local system avoids that specific dependency, although local deployment still requires careful storage and access control.

High-fidelity generative VTON also has its own visual failure modes. A model may alter garment textures, distort logos or patterns, create incorrect sleeve geometry, or produce artifacts where clothing intersects with arms and hair. The supplied reference paper identifies misaligned garment transformations, loss of garment details, and unrealistic body-garment integration as major limitations of existing approaches. These observations motivate explicit treatment of geometry and compositing in the present project.

The proposed system addresses the resource problem through a different architecture. It uses pretrained models for specialized perception and classical processing for transformation and composition. The approach reduces the need for a large end-to-end generator and makes intermediate representations accessible. It also enables resolution fallback. If processing at 640 pixels is too demanding, the system can attempt 384 pixels and then 256 pixels, trading spatial detail for execution feasibility.

TABLE I. DESIGN COMPARISON. VITON emphasizes learned image-based try-on; CP-VTON explicitly incorporates geometric transformation; VITON-HD and HR-VITON emphasize higher-resolution and alignment-aware synthesis; the proposed project emphasizes CPU-oriented modular inference, local deployment, and classical geometric compositing. The proposed system therefore occupies a different engineering position rather than attempting to replace high-end generative approaches.

TABLE I. RESEARCH GAP ANALYSIS

System	Technology	Advantages	Limitations	Research Gap
VITON (Han et al., 2018)	GAN-based coarse-to-fine image synthesis	Learn person-garment relationships end-to-end	Require trained network and fashion-specific	No CPU-only deployment path

VITON-HD (Choi et al.)	Visual awareness, high-resolution GAN	Preserves garment detail at high resolution	High computational and memory cost	Not viable on low-resource hardware
HR-VITON	Joint appearance and flow segmentation GAN	Improve garment alignment and body representation	Needs acceleration for inference speed	Not viable on CPU-only hardware
CP-VTON / CP	TPS-based warping + GAN refinement	Compositional geometry with refinement	Refinement stage still GAN-based and resource-heavy	Only a step towards real-time
VTON+ Cloud-based API	Server-side deep generative models	High visual fidelity, minimal client compute	Requires internet connectivity, privacy concerns from image upload	No offline privacy-preserving option

The research gap is best described as a deployment and integration gap. There is value in a complete VTON prototype that can be studied without requiring a high-end GPU, that exposes intermediate stages for debugging, and that keeps the processing pipeline understandable to students. Such a system can also serve as a baseline for future work in which only selected components are replaced with more advanced models.

The limitation of this claim must also be stated clearly. Lower computational requirements do not automatically imply higher visual quality. A classical TPS-plus-blending pipeline cannot synthesize hidden garment surfaces or realistic cloth behavior as flexibly as a large generative model. The project's contribution is accessibility and modularity, not a state-of-the-art benchmark claim.

The limitations of existing systems can be divided into computational, deployment, privacy, and visual categories. Computational limitations include GPU dependence, high memory use, and expensive inference. Deployment limitations include framework dependencies, model-weight management, accelerator drivers, and cloud-service integration. Privacy limitations arise when personal photographs are transmitted to remote services. Visual limitations include geometric misalignment, texture alteration, boundary artifacts, and incorrect occlusion. The proposed system does not eliminate all of these categories, but it deliberately reduces several computational and deployment burdens.

A useful design comparison is therefore multidimensional. A generative VTON system may provide stronger synthesis of hidden information but require greater computational resources. A purely geometric system may run more easily but can only transform visible information. A cloud API can simplify local installation but introduces network and data-transfer dependencies. The proposed system selects local execution, pretrained specialist models, and classical compositing as a balanced configuration for a student-scale prototype.

The research gap should not be overstated. There are many

efficient computer-vision systems and many VTON systems in the literature. The specific gap addressed here is the integration of a complete VTON workflow around CPU feasibility, local operation, intermediate-stage visibility, and no training requirement during normal operation. This makes the project useful as a reproducible baseline from which more advanced methods can be evaluated.

The gap also has an educational dimension. Students can study how pose estimation, semantic segmentation, salient-object detection, non-rigid transformation, alpha compositing, REST APIs, and frontend interaction cooperate in one system. A large end-to-end generator may hide these relationships. The proposed architecture exposes them, allowing individual stages to be tested independently and improved incrementally.

The computational burden of generative VTON is not limited to the main model. High-resolution synthesis can require preprocessing, feature extraction, multiple intermediate tensors, and postprocessing. When several components are combined, the memory footprint can become larger than expected. The proposed architecture reduces this burden by keeping the AI stages specialized and by applying resolution control before expensive processing. The objective is not zero resource use; it is predictable resource use that can be managed on ordinary hardware.

Local execution also changes the privacy model. With a remote API, the user must trust an external provider to receive, process, and delete the photograph. With local inference, the photograph can remain on the user's computer, but the local application becomes responsible for temporary storage and access control. Therefore, the proposed system reduces one category of exposure without claiming complete privacy automatically. Good file lifecycle management remains necessary.

The visual limitations of the existing systems motivate the modular approach in another way. If a generative system changes a garment pattern or produces an artifact, it may be difficult to determine which internal feature caused the problem. In the proposed system, the source garment remains visible through the pipeline. Developers can compare the original mask, transformed mask, and final composite. This provides a more direct route to diagnosing geometry and blending errors.

The research gap can consequently be stated as a practical systems gap. The project brings together methods that are individually well established but organizes them around CPU feasibility, local deployment, and inspectable intermediate results. This is a meaningful engineering contribution for academic prototyping even though it does not claim to advance the state of the art in photorealistic VTON synthesis.

V. PROPOSED SYSTEM ARCHITECTURE

The proposed architecture is divided into presentation, API, orchestration, AI model, classical processing, and output layers. This layered organization reduces coupling and makes the system easier to maintain. The browser handles user interaction, the API manages communication, the orchestrator controls the sequence of operations, and the image-processing modules

perform the actual visual transformation.

The React/Vite frontend is responsible for image selection, input previews, request submission, progress or status information, and final result visualization. A clean frontend is important because VTON is inherently visual. Users should be able to distinguish the person image from the garment image and understand whether processing is running or completed. The frontend should also display meaningful error messages when input validation or backend processing fails.

FastAPI forms the service layer. It receives the uploaded files, validates them, creates or executes a processing task, and provides the resulting image to the client. The backend also provides health and task-related endpoints where required. Keeping the API independent from the frontend means that the same processing service can later be connected to another client without rewriting the computer-vision pipeline.

The pipeline orchestrator is the central control component. It selects the processing resolution, loads or reuses the required models, invokes pose estimation, semantic parsing, garment extraction, geometric transformation, mask processing, and blending, and records the final result. This controller is intentionally separate from individual model implementations so that each stage has a clear interface.

The AI model layer contains MediaPipe Pose, SegFormer, and U2NETP. MediaPipe Pose supplies body landmarks. SegFormer supplies semantic region information. U2NETP supplies the garment foreground mask. These components operate on different aspects of the input and their outputs are passed to the classical processing layer.

The classical processing layer contains resizing, morphology, TPS transformation, mask operations, alpha blending, and OpenCV utilities. The distinction between AI inference and classical processing is deliberate. The project does not require every stage to be a neural network. A computationally simple mathematical transformation can be preferable when the problem is fundamentally geometric.

Memory management is integrated into the architecture. Model initialization

is expensive, so lazy loading and model reuse are used to reduce repeated overhead. Image dimensions are controlled before inference. The resolution fallback mechanism provides a graceful response to resource pressure instead of treating a single resolution as mandatory. This is particularly relevant for CPU-oriented deployment.

The final layer handles local output storage and result delivery. Temporary data should be managed carefully and removed when no longer required. If persistent history is not part of the application requirement, keeping uploaded photographs only for the duration of the task reduces unnecessary exposure. The complete architecture therefore combines usability, modularity, resource awareness, and privacy considerations.

The architecture can be understood as a directed dataflow. The user first provides two images. The frontend packages the images into a backend request. FastAPI validates and manages the request. The orchestrator chooses a working resolution and invokes the AI model layer. The outputs of those models are converted into masks and geometric references for the classical image-processing layer. The final composite is stored locally and made available

to the frontend. This dataflow prevents UI code from becoming tightly coupled to model implementation details.

The architecture also supports fault isolation. If a garment mask is incomplete, developers can inspect the U2NETP output without changing the pose module. If the garment is correctly extracted but appears too high or too low, the TPS correspondences can be investigated. If the garment position is correct but the face or arms are covered, the semantic protection mask can be examined. This separation makes troubleshooting systematic instead of relying on repeated changes to the entire pipeline.

The output-storage layer should be treated as part of the security boundary. Generated images are user-related data and should not be retained indefinitely by default. A task identifier can be used to associate a request with its temporary result without exposing filesystem paths. Cleanup should occur after the result is no longer needed. For a local academic prototype, this approach keeps the demonstration simple while still following responsible data-handling principles.

The modular architecture also makes future replacement practical. MediaPipe Pose could be replaced by another pose estimator, SegFormer by a different human parser, or U2NETP by a garment-specific segmentation model. The TPS stage could later be supplemented with dense correspondence or a lightweight refinement network. Because the orchestrator defines the sequence, the rest of the application can remain stable while individual research components evolve.

The architecture can be implemented as a pipeline of explicit data contracts. The frontend submits image files; the API converts them into controlled temporary inputs; the orchestrator creates a working representation; model modules return landmarks, labels, or masks; geometric modules return transformed arrays; and the output layer encodes the final image. Each contract should specify expected dimensions and types. This reduces hidden assumptions between components and makes unit testing possible at module boundaries.

The pipeline orchestrator should also preserve stage ordering. Pose estimation and parsing operate on the target-person image. Garment extraction operates on the garment image. TPS requires information from both sides. Blending requires the transformed garment, transformed mask, person image, and protected regions. Running these stages in an incorrect order can produce invalid intermediate data even when every individual module is functioning correctly. A central orchestrator therefore provides an important correctness guarantee.

The architecture is suitable for incremental optimization. If profiling shows that garment extraction dominates processing time, that module can be optimized without modifying the frontend. If TPS is inexpensive but image encoding is slow, the output layer can be examined independently. This separation also supports alternative deployment modes, such as a local desktop service or a laboratory server, because the frontend communicates through the API rather than directly calling model libraries.

For academic demonstration, the architecture can be represented using a block diagram and a sequence diagram.

The block diagram explains component responsibilities, while the sequence diagram explains the order in which a request travels through the system. Both representations reinforce the project's central idea: a complete VTON result is produced by coordinating multiple specialized operations rather than by one opaque procedure.

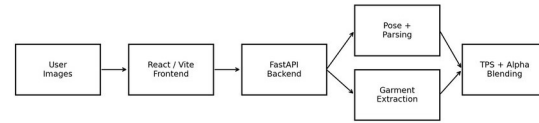


Fig.1. Proposed lightweight CPU-oriented architecture.

VI. DETAILED PROCESSING PIPELINE AND MATHEMATICAL MODEL

The first processing stage validates the uploaded files. The application should check that the files are readable images and that their dimensions are within an acceptable range. Images are then converted to a consistent internal representation. Resizing is performed when necessary to reduce memory use and establish a predictable processing scale. The selected scale is influenced by the resolution fallback policy.

The second stage performs pose estimation. MediaPipe Pose identifies body landmarks from the target-person image. The landmark set provides approximate positions for structural regions such as shoulders, elbows, and hips. These points can be normalized to the image coordinate system so that the geometric stage is less dependent on the absolute image dimensions.

The third stage performs semantic parsing. SegFormer assigns semantic labels to regions of the target image. The result is useful for identifying areas that should be protected during compositing. The exact labels available depend on the model configuration, but the architectural purpose remains the same: distinguish clothing-related regions from face, hair, arms, and background so that the final overlay is controlled rather than unrestricted.

The fourth stage extracts the garment foreground. U2NETP produces a salient-object mask for the garment image. The mask can be thresholded or refined according to the application's needs. Morphological opening or closing can reduce small isolated regions and fill minor holes. The RGB garment and its mask must retain identical spatial dimensions so that subsequent warping is consistent.

The fifth stage constructs geometric correspondences. Source control points are defined on meaningful positions of the garment image, while target points are derived from the target person's body geometry. For example, upper-left and upper-right garment regions can be associated with the approximate shoulder positions. Additional points can control lower garment extent. The accuracy of these correspondences has a direct influence on the quality of the deformation.

The Thin Plate Spline transformation can be described conceptually by $f(x,y) = a_1 + a_2x + a_3y + \sum w_i U(r_i)$, where $U(r) = r^2 \log(r^2)$ is the radial basis function and r represents distance from a control point. The affine component provides global movement while the radial

terms provides smooth local deformation. The transformation is applied to both the garment image and its mask.

The sixth stage performs mask-aware compositing. Let I denote the target person image, G the warped garment image, and M the normalized garment mask. A basic composite is $O = M \odot G + (1 - M) \odot I$. The operator $\odot$ represents element-wise multiplication. If the mask is refined using protected semantic regions, the final M is reduced or zeroed in regions that should remain visible, such as the face, hair, or uncovered arms.

The seventh stage performs boundary refinement and output generation. Minor mask smoothing can reduce hard edges, while excessive smoothing should be avoided because it can make garment boundaries transparent. The output is converted into a standard image format and stored locally before the API returns the result. The pipeline is deterministic for a fixed set of model outputs and transformation parameters, making it easier to debug than a stochastic generative workflow.

The preprocessing stages should preserve the relationship between the original image and the working image. If the image is resized, the scale factor should be known so that any coordinate produced by a model can be interpreted correctly. This is particularly important for TPS because control points are numerical inputs to the transformation. A small coordinate error can become a visible garment-placement error after deformation.

Control-point selection is one of the most influential parts of a TPS-based pipeline. Points should correspond to meaningful garment locations and body landmarks rather than arbitrary pixels. Upper garment corners, shoulder-related points, side boundaries, and lower anchors can provide a stable geometric structure. The number and placement of points should remain consistent enough that the transformation behaves predictably across similar inputs. Poor correspondences can cause local folding or stretching even when the TPS mathematics is correct.

Mask processing should be performed with the same care as RGB processing. A mask represents transparency and therefore directly determines which pixels reach the final output. A noisy mask can create isolated garment fragments; a mask with holes can remove texture; and a mask that extends into the background can introduce unwanted pixels. Morphological opening and closing are useful because they operate on local structure, but their parameters should be selected conservatively to avoid erasing thin garment details.

The alpha-compositing formulation also clarifies why protected regions are necessary. Without a protection mechanism, every nonzero garment-mask pixel can overwrite the person image. A semantic protection mask modifies the effective alpha and creates an explicit rule for preserving selected regions. This is a simple but interpretable alternative to asking a large generative model to learn all occlusion relationships implicitly.

The final output should be converted into a standard display format after all numerical operations are complete. During processing, floating-point or normalized mask representations may be convenient, but the browser generally expects an encoded image. The conversion stage should ensure that image dimensions, color channels, and alpha behavior are correct. A

technically successful algorithm can still appear incorrect if the final encoding is mishandled.

The mathematical formulation provides a useful way to reason about failure propagation. If the garment mask is wrong, the error enters the TPS stage because the mask is transformed along with the garment. If the control points are wrong, the warped

image is wrong even when the mask is perfect. If the protected-region mask is wrong, a correctly warped garment can still cover the wrong pixels. This chain demonstrates why intermediate validation is important: correcting the final image alone may hide the true source of the problem.

TPS also has a smoothness advantage over independent pixel displacement. Neighboring pixels remain related through the continuous deformation field. This helps the garment preserve its overall structure when control points are moved. However, smoothness is not equivalent to realism. The transformation does not know that a sleeve has a particular fabric thickness or that a collar should interact with a neck. The method should therefore be viewed as a controlled geometric approximation.

Mask-aware processing is equally important at the boundary. A binary mask can create a visibly hard edge if the garment is placed over a differently illuminated background. A soft alpha transition can reduce the abruptness, but excessive softness causes the garment to appear transparent. The implementation should therefore balance edge continuity and opacity. This is a visual parameter that can be tested independently from the TPS geometry.

The final mathematical operation is computationally simple compared with neural inference, which is one reason alpha blending is attractive for CPU deployment. Once the masks and warped image are available, the composite can be produced with element-wise operations. Efficient array processing through OpenCV or NumPy can therefore keep the final stage relatively lightweight.



Fig.2. Sequential VTON processing pipeline.

VII. SOFTWARE IMPLEMENTATION AND RESOURCE MANAGEMENT

The implementation is based on a web-service model. React and Vite provide the client-side interface, while FastAPI provides the Python service layer. Computer-vision libraries and pretrained models are loaded by the backend. This division keeps browser responsibilities lightweight and places computational operations in an environment where Python-based AI and image-processing libraries are readily available.

The frontend workflow begins when the user selects a person image and a garment image. The interface validates basic input conditions before submitting the files. During processing, the user receives a status indication. After completion, the generated image is displayed without requiring the user to understand the internal model pipeline. The interface can later be extended with comparison views, reset controls, a

ndresulthistory. FastAPI handles request validation and communication. Uploaded images should be given controlled temporary paths rather than arbitrary user-supplied filesystem locations. File names should be sanitized, file types checked, and maximum upload sizes enforced. The service should return structured status information so that frontend behavior is predictable.

Model lifecycle management is especially important for CPU-oriented deployment. Loading a model for every request wastes time and may create repeated memory allocation. Lazy initialization loads a model when its stage is first needed, while reuse keeps the initialized component available for subsequent tasks. In a production environment, the lifecycle strategy should be adjusted according to available memory and concurrency requirements.

The resolution fallback strategy is another important implementation feature. A high-resolution input provides more visual information but increases the number of pixels processed by every stage. If the 640-pixel path exceeds available resources, the pipeline can attempt 384 pixels and then 256 pixels. The fallback should be reported clearly so that users understand why output detail may vary between requests.

Resource monitoring should consider both memory and execution time. Useful engineering measurements include peak resident memory, CPU utilization, model loading time, per-stage inference time, TPS transformation time, blending time, and total end-to-end latency. These measurements can identify bottlenecks without requiring a new machine-learning benchmark.

Failure handling should be stage-aware. A pose failure is different from a garment extraction failure. A parser failure should not be reported as a generic server error if the system can identify the responsible stage. Similarly, an output-storage error should not be confused with a computer-vision error. Clear stage-level logging improves debugging and project demonstration.

Local execution provides an additional software-design advantage: the application can operate without an external VTON API. This reduces dependence on network availability and avoids sending photographs to a third-party inference service. However, local execution still requires responsible data management, especially if multiple users can access the machine or if generated images are stored for long periods.

The implementation therefore combines standard web development with specialized computer vision. The main engineering challenge is orchestration: ensuring that the output of one component is represented in the correct format and coordinate system for the next component. Consistent image dimensions, masks, data types, and error states are as important as the individual models.

The frontend and backend communicate through a clearly defined API contract. The request should specify the person image and garment image without exposing internal model paths. The response should provide a task state and, when complete, a result reference that the frontend can display. This separation also makes automated API testing possible because backend behavior

can be checked independently from the browser interface.

For CPU-oriented inference, concurrency should be considered carefully. Running several large inference operations simultaneously may increase memory consumption and cause CPU contention. A task-oriented backend can control how many processing jobs execute at once. For a student prototype, sequential execution is often easier to reason about and can prevent multiple requests from exhausting the available resources. Future deployment can introduce a queue when multi-user operation is required.

The model loaders should also distinguish model initialization from request-specific data. A reusable model object can remain in memory while image tensors, masks, and temporary arrays are released after each request. This reduces repeated model loading while limiting the lifetime of request-specific information. Explicit cleanup is particularly important when several stages produce large arrays at the same working resolution.

Logging should be useful but minimal. Stage names, task identifiers, processing durations, selected resolution, and error categories can help diagnose problems. Raw user photographs should not be written into ordinary logs. This approach supports troubleshooting while reducing unnecessary exposure of image content. The same principle applies to exception messages returned to the frontend: they should explain the operational problem without revealing internal filesystem details.

The software stack also provides a practical development path. React/Vite can be developed independently from FastAPI. The backend can be tested using API clients before the frontend is complete. Individual computer-vision modules can be tested using saved intermediate images. This incremental workflow reduces the risk of debugging several unrelated problems at the same time.

A practical deployment should separate configuration from source code. Model paths, working resolutions, upload limits, temporary-directory locations, and logging levels should be configurable rather than hard-coded. This allows the same application to run on different computers without modifying the core pipeline. It also makes future model upgrades easier because model identifiers can be changed through configuration.

The API should provide predictable response states. A request can move from accepted to processing and then to completed or failed. If the system uses task identifiers, the frontend does not need to keep a long-running HTTP connection open for the entire inference period. This is useful for CPU-oriented processing because inference duration can vary according to image complexity and selected resolution.

Resource management should include cleanup of intermediate arrays and temporary files after a request finishes. Python garbage collection alone may not immediately release every resource held by libraries, so explicit lifecycle design is valuable. Model reuse should be balanced against memory availability. If the machine is extremely constrained, unloading an infrequently used model may be preferable to keeping every component resident.

The development workflow should include unit tests for utilities, integration tests for the API, and visual tests for the image pipeline. A utility test can verify resizing or mask normalization. An API test can verify task states and error responses. A visual test

can compare intermediate masks on a fixed input set. Together these tests provide stronger evidence than checking only the final browser image.

VIII. EVALUATION METHODOLOGY AND EXPECTED BEHAVIOR

The project material does not contain a controlled numerical benchmark or experimentally measured quality table. Therefore, a formal evaluation should be designed rather than fabricated. The evaluation can be divided into functional correctness, geometric quality, segmentation quality, visual compositing quality, computational resource use, and robustness.

Functional evaluation should first confirm that the application accepts valid person and garment images, rejects invalid files, starts the processing pipeline, returns an output, and handles failures without crashing the service. API tests can verify request formats, status responses, output availability, and deletion or cleanup behavior where those functions are implemented.

Geometric evaluation should inspect shoulder alignment, sleeve direction, collar placement, garment width, garment length, and major contours. Test images should include straightforward frontal or near-frontal poses as well as moderate pose variation. Difficult poses should be included as stress cases rather than excluded, because they reveal the limits of sparse landmark-based alignment.

Segmentation evaluation should examine the garment mask separately from the final image. A good mask should retain the garment while suppressing the source background. Similarly, human parsing should provide sufficiently useful protected regions. Errors in either mask can propagate into the final composite, so intermediate masks are valuable evaluation artifacts.

Visual quality evaluation should examine garment texture retention, boundary continuity, color consistency, transparency, and accidental overlap with protected regions. The system is expected to preserve the original garment pixels more directly than a generative system because it uses an extracted garment image and geometric transformation. However, this also means that it cannot synthesize missing texture or physically accurate folds when those details are absent from the source image.

Computational evaluations should compare multiple working resolutions. The same input set can be processed at 640, 384, and 256 pixels while recording total processing time and peak memory. If a lower resolution succeeds after a higher resolution fails, the fallback mechanism can be considered functionally useful. The analysis should report both success rate and visual degradation rather than treating lower resolution as automatically better.

Robustness testing should vary image resolution, garment background complexity, garment aspect ratio, pose, lighting, body orientation, and clothing style. Results should be categorized by failure stage. This produces a useful engineering profile: for example, some cases may fail because pose landmarks are unreliable, while others may fail because the garment mask is incomplete or the TPS correspondences are inappropriate.

For future research, objective metrics such as structural similarity, perceptual distance, segmentation overlap, or user-based quality ratings can be introduced when a suitable ground-truth dataset is

available. A formal comparison with VTON benchmarks would require consistent datasets, preprocessing, evaluation protocols, and hardware reporting. Such experiments are outside the evidence available in the current project material.

The expected behavior of the present prototype is therefore qualitative:

it should produce a recognizable transferred garment when the person poses and a garment mask is sufficiently clear, while gracefully degrading or failing on cases outside its geometric assumptions. This expectation should be communicated clearly in demonstrations and project reviews.

A complete evaluation should use a fixed test protocol. The same set of person and garment images should be processed at each selected resolution so that changes can be attributed to the resolution rather than to different inputs. The hardware and software environment should be recorded. Model versions and preprocessing parameters should remain unchanged during a comparison unless the experiment is specifically testing those parameters.

Functional tests should include successful and unsuccessful paths. A valid pair should reach the output stage. Missing files should be rejected. Unsupported formats should not enter model inference. Oversized images should be constrained before memory-intensive operations. If a model cannot initialize, the API should return a controlled failure rather than leaving the task in an indefinite state. These tests evaluate software reliability independently of visual quality.

Intermediate-result evaluation is particularly important for this project. A final image can hide the cause of an error. Therefore, the evaluation should inspect pose landmarks, semantic parsing, garment mask, TPS result, effective alpha mask, and final composites separately. This allows a failure taxonomy to be created. For example, a garment may fail because it was not extracted, because it was extracted but incorrectly warped, or because it was correctly warped but incorrectly composited.

Visual evaluation can be structured around recognizable criteria. Garment fidelity asks whether the source garment's major colors, patterns, and shape remain identifiable. Geometric alignment asks whether shoulders, collar, sleeves, and lower boundaries correspond to the target pose. Boundary quality asks whether the transition between garment and person is smooth. Occlusion correctness asks whether protected regions remain visible. These criteria are more informative than a single subjective statement that an output 'looks good.'

Resource evaluation should include both peak memory and time. A pipeline that produces a visually acceptable image but consumes more memory than the target machine can provide is not successful for the stated CPU-oriented objective. Conversely, a very low-resolution result that completes quickly may sacrifice too much detail. The desired behavior is therefore a usable trade-off supported by the fallback mechanism rather than a single universally optimal resolution.

A useful evaluation report should separate capability from quality.

Capability asks whether the pipeline completes a request and returns an image. Quality asks whether the returned image is aligned, visually coherent, and faithful to the garment. Resource evaluation asks whether this

can be achieved within the target hardware limits. These dimensions should be reported separately because a system can be functional but visually limited, or visually good but computationally impractical.

Test cases should also include controlled variations. Starting from one person image and one garment image, the experiment can vary the working resolution while keeping all other inputs constant. Another experiment can vary the garment background while keeping the person image constant. A third can vary the target pose. Such controlled changes make it easier to attribute output differences to a specific factor.

A failure log is valuable for future development. Each failed case can record the input characteristics, selected resolution, stage at which the problem appeared, and visible artifact. Over time, these records can reveal recurring patterns, such as pose failure on side views or segmentation failure on visually similar backgrounds. This converts qualitative debugging into a structured engineering process.

If a formal benchmark is later introduced, the evaluation protocol should be frozen before results are collected. This prevents selecting only successful images or changing preprocessing after seeing the output. The current project can serve as the baseline implementation, while future methods can be compared under the same inputs and hardware conditions.

IX. RESULTS DISCUSSION, LIMITATIONS, PRIVACY AND SECURITY

The expected output of the proposed pipeline is a composited image in which the selected garment follows the approximate geometry of the target person's upper body. The strongest cases are expected when the person image has clear body landmarks, the garment image has a clean foreground, and the target pose does not require extreme deformation. In these conditions, TPS can produce a visually meaningful transformation while preserving the original garment pixels.

The modular pipeline provides an important form of interpretability. The system can expose the pose landmarks, human parsing output, garment mask, warped garment, and final composite as separate intermediate artifacts during development. These artifacts allow developers to identify whether an error originates before or after geometrical alignment. This is particularly valuable for a student project because it converts a visually complex application into a sequence of understandable computer-vision operations.

The first major limitation is garment deformation. TPS is a smooth geometric mapping, not a physical cloth simulator. It can stretch and bend pixels according to control points, but it does not know the physical properties of fabric. It cannot automatically infer how folds, thickness, wrinkles, or hidden surfaces should appear under a new pose. Extreme deformation may therefore produce stretched textures or unnatural boundaries.

The second limitation is occlusion. A person image can contain arms, hair, hands, accessories, or other objects that should appear in front of or behind the garment. Simple alpha blending cannot

always determine the correct depth ordering. Semantic parsing provides useful region information, but complete occlusion reasoning is a harder problem and remains an area for future enhancement.

The third limitation is segmentation quality. U2NETP can estimate a salient garment region, but source photographs can contain complex backgrounds, shadows, similar colors, or partially hidden clothing. Imperfect masks can introduce background pixels or remove legitimate garment pixels. Similarly, errors in human parsing can affect protected regions and lead to unwanted overlays.

The fourth limitation is visual realism. Because the proposed pipeline does not contain a dedicated generative refinement network, it does not explicitly synthesize realistic illumination changes, cloth folds, or unseen garment surfaces. The resulting images should therefore be described as a geometric image-based try-on rather than a complete physical simulation or guaranteed photorealistic rendering.

Privacy is a significant advantage of local processing when it is implemented correctly. Personal photographs can remain on the local machine rather than being uploaded to a remote VTON service. However, local processing does not itself guarantee privacy. Temporary images, generated results, application logs, and browser caches can still contain sensitive information. The application should retain files only as long as necessary and should provide appropriate cleanup.

Security becomes more important if the prototype is exposed as a network service. Upload size limits, file-type validation, safe temporary directories, sanitized file names, restricted filesystem permissions, and path-traversal protection should be applied. Authentication, authorization, HTTPS, rate limiting, and process isolation should be considered for multi-user or public deployment.

Overall, the system represents a deliberate trade-off. It prioritizes accessibility, modularity, and local execution while accepting reduced synthesis capability. The correct evaluation question is therefore not whether it completely replaces a high-end generative VTON system, but whether it provides a useful and reproducible VTON workflow at a lower computational and deployment cost.

The expected result should be interpreted according to the information available in the inputs. When the garment photograph clearly shows the complete clothing item and the person photograph provides visible upper-body landmarks, the pipeline has enough information to perform a meaningful geometric transfer. When information is missing, such as a heavily occluded sleeve, the system cannot reliably reconstruct the hidden portion through TPS alone. This is a fundamental difference between geometric transformation and generative synthesis.

Interpretability is one of the strongest practical properties of the proposed pipeline. Developers can save each intermediate result and compare it with the expected behavior of that stage. This makes it possible to distinguish model limitations from integration bugs. It also makes project demonstrations more informative because the audience can see how the final image is constructed rather than seeing only the final output.

Privacy should be considered throughout the complete data lifecycle.

The person photograph enters the browser, travels to the local API, is processed in memory, may be written temporarily to disk, and finally produces an output image. Each stage is a potential storage or exposure point. A privacy-aware implementation should minimize retention, use controlled temporary directories, avoid unnecessary copies, and remove temporary files after completion. If the application is extended to remote access, network encryption and authentication become additional requirements.

Security and resource protection are also connected. An attacker or accidental user could submit extremely large images or many simultaneous tasks and consume the available memory and CPU. Upload limits, concurrency controls, and timeouts reduce this risk. The application should also avoid accepting arbitrary filesystem paths or commands through request parameters. These controls are standard software safeguards and

become more important if the academic prototype is demonstrated on a shared network. The project's main trade-off can therefore be stated clearly: it reduces computational and deployment complexity by accepting limitations in synthesis quality and physical realism. This trade-off is appropriate for a CPU-oriented academic system, provided

the output is presented as an approximate visualization. The system should not be evaluated using claims that require capabilities it was not designed to provide. The absence of fabricated numerical results is important for scientific reporting. The project source describes the system design and intended advantages but does not provide a controlled table of measured accuracy, latency, or memory.

Therefore, the present journal should report expected behavior and evaluation methodology rather than inventing performance values. When actual experiments are completed, the numerical results can be inserted into the evaluation section without changing the architecture discussion. A qualitative result can still be analyzed systematically. For example, an output can be described as having correct overall placement but a weak sleeve boundary, or good garment fidelity but incorrect arm occlusion. Such descriptions identify the component that should be improved. A simple visual rating scale can be used in future testing, provided the criteria are defined before evaluation and applied consistently across test cases. Privacy and security are also part of result quality in a local application. A visually correct result that leaves personal photographs in temporary directories indefinitely is not a complete deployment solution. Cleanup behavior, access permissions, and retention policy should therefore be tested alongside the image-processing stages. If the system is demonstrated to multiple users, task isolation should ensure that one user cannot access another user's result by guessing a file path or task identifier. The system's limitations should be presented as engineering boundaries rather than failures of the entire VTON concept. Geometric methods are well suited to explicit deformation but limited in their ability to invent missing appearance information. Semantic parsing improves protection but depends on model accuracy. Local execution improves control but shifts resource responsibility to the host machine. These trade-offs provide clear directions for future development.

X. FUTURE ENHANCEMENTS AND CONCLUSION

The modular architecture creates several clear opportunities for future development. The first is improved occlusion handling. A depth-aware or layered compositing system could estimate whether an arm, hand, hair region, or accessory should appear in front of the garment. Instead of using one final mask, the system could maintain separate foreground layers and combine them according to estimated ordering.

The second enhancement is dense geometric alignment. Sparse pose and landmarks provide only a limited number of control points. Dense correspondence estimation or body-surface representation could provide more accurate deformation, particularly around sleeves, collars, and lower garment boundaries. A garment-type-specific control-point strategy could also improve results without requiring a large generative model.

The third enhancement is multi-garment support. The current design can serve as a basis for transferring different garment categories, but each category has different geometric and occlusion requirements. Shirts, jackets, dresses, trousers, and accessories could be represented by separate transformation rules or segmentation categories. Coordinating multiple garments would require consistent body geometry and layer ordering.

The fourth enhancement is lightweight learned refinement. A compact neural model could be introduced after TPS and before final output, with a narrow, well-defined role such as boundary correction, texture restoration, or illumination adjustment. This would preserve the project's modular CPU-oriented philosophy better than replacing the complete pipeline with a large generator.

The fifth enhancement is quantitative evaluation. A future version should use a suitable public VTON dataset and report image-quality, alignment, segmentation, and computational metrics under a fixed protocol. Hardware configuration should be recorded because CPU time and memory usage are central to the project's contribution. Human evaluation can additionally measure whether users perceive the garment as correctly aligned and visually plausible.

The sixth enhancement concerns the web application. The interface can provide side-by-side input and output views, processing progress, quality warnings, retry controls, and optional result comparison. A user could be informed when the pipeline has fallen back from 640 pixels to a lower resolution. Such feedback makes the system more transparent and reduces confusion when output quality varies.

In conclusion, this paper presents an AI Virtual Try-On system designed around a lightweight CPU-oriented architecture. The proposed workflow integrates MediaPipe Pose for body landmarks, SegFormer for semantic human parsing, U2NETP for garment extraction, ThinPlate Spline transformation for non-rigid alignment, and alpha blending for final image composition. React/Vite and FastAPI provide the web application framework, while lazy loading, model reuse, resizing, and resolution fallback address resource constraints.

The project demonstrates an alternative design philosophy for VTON. Rather than depending entirely on a large generative model, it combines specialized AI perception with classical geometric processing. This provides visible intermediate stages, easier debugging, local execution, and lower deployment complexity. The trade-off is

reduced ability to synthesize missing information, realistic cloth physics, complex occlusion, and high-fidelity appearance changes.

The proposed system is therefore best viewed as an accessible research and educational baseline. Its value lies in integrating multiple computer-vision techniques into a complete application while explicitly considering hardware constraints and privacy. Future work can progressively replace individual modules with stronger models, add quantitative benchmarks, improve occlusion handling, support more garment categories, and introduce selective learned refinement without abandoning the modular architecture.

Future improvements can be introduced incrementally. Occlusion-aware layering can be added without replacing the frontend. A denser correspondence module can replace or supplement the current TPS anchor strategy. A garment-specific segmentation model can improve foreground extraction. A compact refinement model can be inserted after compositing preparation. This incremental path is a direct benefit of the modular architecture and allows future work to target the most visible limitation first.

Another important enhancement is category awareness. Different garment types require different geometric assumptions. A shirt may be anchored primarily around the shoulders and torso, while trousers require lower-body landmarks and different protected regions. Dresses can span multiple body areas. A future classifier could identify garment category and select an appropriate transformation and mask strategy. This would improve the usefulness of the system without requiring one universal geometric rule.

A production-oriented version could also introduce explicit task cancellation and resource monitoring. If a user cancels a request, the backend should stop unnecessary processing and release request-specific memory. A progress indicator could show which stage is running. A resource warning could explain when the pipeline has moved from 640 to 384 or 256 pixels. These changes would make the system more transparent and predictable during longer CPU inference tasks.

The strongest long-term research direction is a hybrid system. Classical TPS can remain responsible for coarse geometric alignment because its behavior is interpretable and computationally manageable. A small learned module can then refine boundaries, restore texture continuity, and improve garment-body integration. This would preserve the project's CPU-oriented philosophy while addressing some limitations that cannot be solved by geometry alone. In summary, the proposed AI Virtual Try-On System demonstrates a complete workflow built from specialized pretrained inference models and classical image processing. MediaPipe Pose, SegFormer, and U2NETP provide perception; TPS provides non-rigid alignment; morphology and alpha blending provide controlled composition; React/Vite and FastAPI provide the application interface; and memory-aware execution makes the workflow suitable for ordinary hardware. The project therefore provides a practical academic baseline for future VTON research.

REFERENCES

[1] Google MediaPipe, "MediaPipe Pose Documentation," Google Research and MediaPipe Framework.

[2] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," *Advances in Neural Information Processing Systems*, 2021.

[3] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand, "U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection," *Pattern Recognition*, 2020.

[4] F. L. Bookstein, "Principal Warps: Thin-Plate Splines and the Decomposition of Deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[5] X. Han et al., "VITON: An Image-Based Virtual Try-On Network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] B. Wang et al., "Toward Characteristic-Preserving Image-based Virtual Try-On Network," *European Conference on Computer Vision*, 2018.

[7] S. Choi et al., "VITON-HD: High-Resolution Virtual Try-On via Misalignment-Aware Normalization," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.

[8] S. Lee et al., "HR-VITON: High-Resolution Virtual Try-On via Misalignment and Occlusion-Aware Normalization," *European Conference on Computer Vision*, 2022.

[9] PyTorch, "PyTorch Documentation: An Open Source Machine Learning Framework."

[10] OpenCV, "Open Source Computer Vision Library Documentation."

[11] FastAPI, "FastAPI Documentation: Modern, Fast Web Framework for Building APIs with Python."

[12] React, "React Documentation: The Library for Web and User Interfaces."

[13] Vite, "Vite Documentation: Next Generation Frontend Tooling."

[14] HuggingFace, "Transformers Documentation and Model Hub."

[15] NumPy Developers, "NumPy Documentation: Scientific Computing with Python."

[16] SciPy Community, "SciPy Documentation: Scientific Computing Tools for Python."

[17] H. Lee, D. Jeong, J. Park, S. Han, J. Jang, and J. Paik, "Dual-Stage Detail-Preserving Virtual Try-On Network With Geometric Refinement and Multi-Scale Feature Integration," *IEEE Access*, 2025.